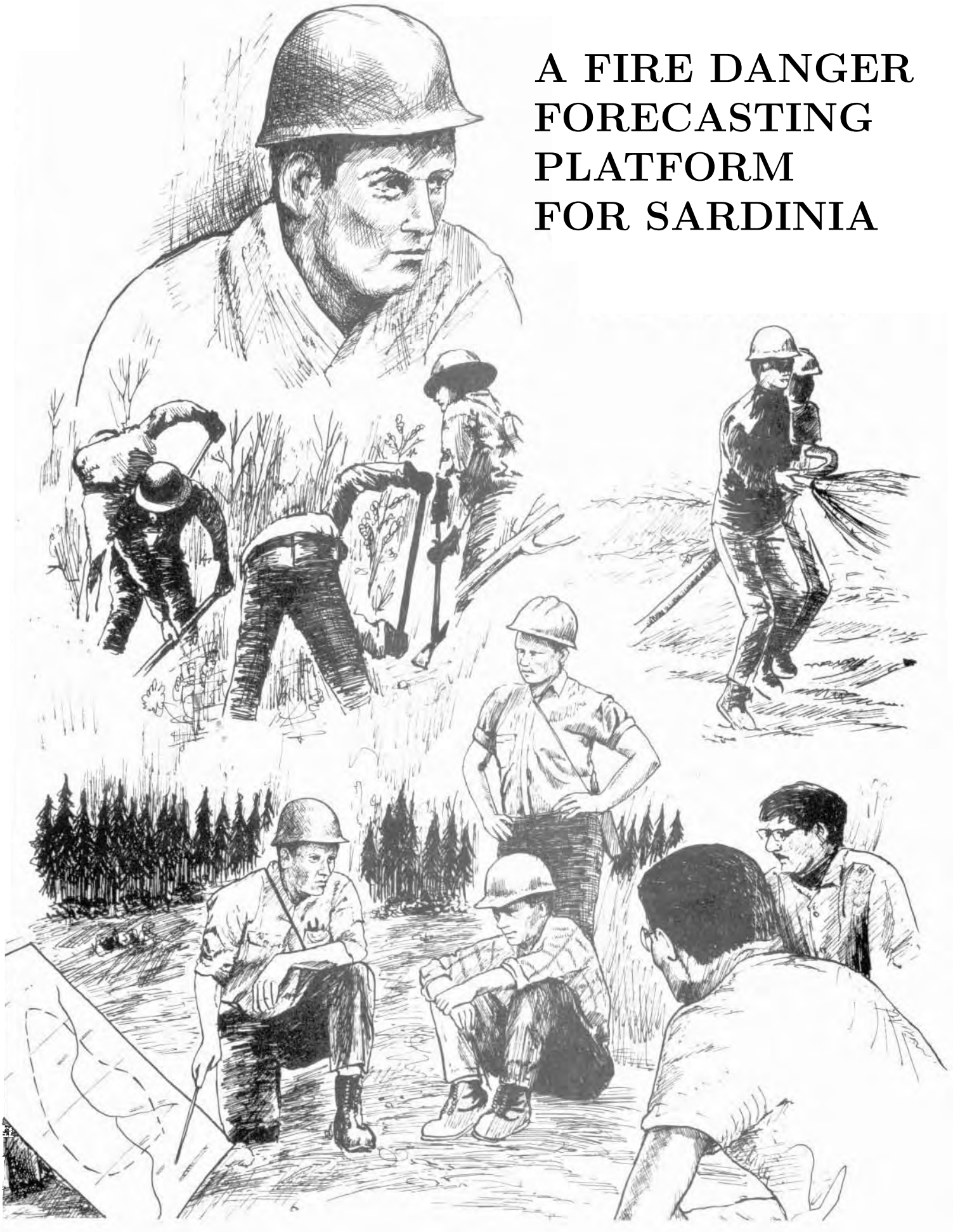
A FIRE DANGER FORECASTING PLATFORM FOR SARDINIA



Development of a web-based Fire Danger Forecasting System for Mediterranean Landscapes using Open Source Software and crowd-sourced Weather Data - the Isle of Sardinia (Italy) as an Example

Christian-Albrechts-Universität zu Kiel
Mathematisch-
Naturwissenschaftliche Fakultät

Dissertation zur Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät der
Christian-Albrechts-Universität zu Kiel

vorgelegt von
Dipl.-Geogr. Michael Nolde
Kiel, 2013

Erster Gutachter:	Prof. Dr. Rainer Duttmann
Zweiter Gutachter:	Prof. Dr. Athanasios Vafeidis
Tag der mündlichen Prüfung:	21. August 2013
Zum Druck genehmigt:	21. August 2013

gez.

Prof. Dr. Wolfgang J. Duschl, Dekan

Michael Nolde
Blocksberg 11b, 24103 Kiel
geboren am 9. Januar 1982 in Karlsruhe

Staatsangehörigkeit: deutsch
Studiengang: Diplom-Geographie
Nebenfächer: Informatik und Ozeanographie,
Politische Wissenschaft bis einschließlich Vordiplom
Abschluss: Diplom
Titel der Diplomarbeit: Development of an application for spatial and temporal analyses of wildfires using hot spot satellite data

Wissenschaftlicher Werdegang:

10/2002 - 04/2009	Studium der Geographie an der Christian-Albrechts - Universität zu Kiel (elf Studiensemester) *ab 01/2008:* Anstellung als studentische Hilfskraft am Lehrstuhl für Landschaftsökologie und Geoinformation
07/2006 - 06/2007	Studium an der Naturwissenschaftlich-Technischen Universität Trondheim, Norwegen (zwei Studiensemester)
seit 06/2009	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Landschaftsökologie und Geoinformation der CAU Kiel

Bibliographic Information

Faculty:	Faculty of Mathematics and Natural Sciences
Department:	Geography
Author:	Dipl.-Geogr. Michael Nolde
Degree programm:	Geography (Diploma)
Student register:	685205
Work description:	Doctoral thesis
Title:	Development of a web-based Fire Danger Forecasting System for Mediterranean Landscapes using Open Source Software and crowd sourced Weather Data - the Isle of Sardinia (Italy) as an Example
Amount:	160 pages, 35 figures, 25 tables, 25 scripts
Advisor:	Prof. Dr. Rainer Duttmann
Co-reviewer:	Prof. Dr. Athanasios Vafeidis
Closing date:	
Keywords:	Wildfire Forecasting, Sardinia, FWI

Minimal Abstract:
This thesis covers the preparation and realisation of setting up a fire danger forecasting platform for the Isle of Sardinia. The system created uses, firstly, crowd sourced weather data as input for calculation of fire danger predictions and, secondly, only free and open source software for the actual implementation. The platform is meant to provide regional, daily fire danger forecasting maps. Its configuration is based on a ten-year study of fire occurrence on Sardinia and Crete. The Fire Weather Index (FWI) with threshold values adapted to Sardinia is used for the forecasts.

meiner Familie,
meinen Freunden,
und Reidun im Besonderen

Acknowledgements

I would like to thank my doctoral thesis supervisor, Prof. Dr. Rainer Duttmann as well as my co-reviewer, Prof. Dr. Athanasios Vafeidis, for assistance and advice. Furthermore, many thanks to those who supported this work with additional suggestions and helpful information, including the staff of FIRMS, DWD, and EFFIS, as well as to my colleagues, friends and family. I express my deepest gratitude to Reidun for her constant motivational support and for the tremendous effort of proofreading this thesis. I also like to thank Lennart and Michael for final spell checking and feedback regarding statistics, respectively. Finally, I would like to acknowledge the software contributors in the geospatial Open Source community, especially the ones concerned with the development of GDAL/OGR, GRASS, and GMT.

Cover - Illustration:
Adapted from: Turner, J.A. 1973. A Fire Load Index for British Columbia. Environment Canada, Canadian Forestry Service, Forest Research Lab. Information Report BC-X-80.

"A little fire is quickly trodden out,
Which being suffer'd,
Rivers cannot quench."

Shakespeare, Henry the Sixth.
Part 3, Act 4, Scene 8 / Clarence

Abstract

This thesis aims at creating spatially and temporally differentiated wildfire danger forecasts for the Mediterranean area. It also intends to enhance the prediction quality of existing systems regarding reliability, spatial accuracy and prediction period.

Exclusively freely available weather data serves as a base for the predictions. This work further tries to illustrate the usability of this data as a replacement or supplement to proprietary data. The Isle of Sardinia is used as a region of study representative for Mediterranean conditions. Also, the Isle of Crete serves as a testing region to ensure the portability of the forecasting methodology to other regions in the Mediterranean.

This methodology is further integrated into a web-based, autonomous platform for prediction purposes, providing daily maps of wildfire danger for the region of study. The conception and realisation of this platform is explicitly described in this work.

A long-term study of the occurrence of wildfire on the islands of Sardinia and Crete between 2001 and 2010 serves as a means of calibration of the forecasting system. Additionally, historical weather data is used to retrospectively generate wildfire danger forecasting maps, which are then compared with the actual occurrence of fires. The Weather Underground network, which is supplied by crowd sourcing, serves as source of the required historical data. Information on wildfire occurrences are taken from MODIS hotspot data. Further datasets regarding land usage and burned areas are utilised to filter and validate the fire occurrence data.

The Canadian Fire Weather Index (FWI) serves as a means of estimating actual fire danger based on weather data. It is widely used as a general indicator of probability of fire occurrence. Two different classification schemes are utilised to classify the FWI outputs into classes of fire danger, ranging from 'very low' to 'extreme': Firstly, the scheme of the European Forest Fire Information Systems (EFFIS), and, secondly, a scheme developed within this thesis, which is based on the findings of the long-term study and especially adjusted for usage on Sardinia.

Both classifications are tested regarding the yielded results. Their ratio of false alarms as well as their recall ratio (type I and type II - error) serve as validation criteria. The testing results illustrate the superiority of the newly developed classification in respect to reduction of false alarms: While these occur in only 14.8% of cases using the new scheme, the percentage is more than twice as high - 34.2% - when using the scheme of EFFIS. However, the EFFIS methodology exhibits a better recall ratio: Only in 2% of cases, fires occurred in areas classified as 'not endangered'. Using the scheme developed in this thesis, these cases amount to 13.3%. This result is mainly due to inadequate fuel moisture modelling. Therefore, this modelling procedure should be optimised in order for the system to be capable of producing a more realistic picture of actual fire danger.

The well-working system was tested against the forecasting results published by EFFIS during summer 2012. While the results generally tend to be more moderate in terms of fire danger compared to EFFIS, the prediction of fire danger is closer to real conditions.

Daily forecasting results for the next ten days are available online via: http://www.sardinia-fireweather.org
The fire danger maps can also be consumed in form of a Web Map Service (WMS) or as a mobile application.

The implementation itself is free to download, use and modify under the terms of the GNU GPL. All scripts required for the operation of the forecasting platform are given in the appendix of this thesis.

Zusammenfassung

Die vorliegende Arbeit hat die Erstellung von räumlich und zeitlich differenzierten Waldbrand-Gefahrenvorhersagen im Mittelmeerraum zum Ziel. Dabei soll die Vorhersagequalität von derzeit bestehenden Vorhersagesystemen bezüglich Verlässlichkeit, räumlicher Genauigkeit und Vorhersagedauer verbessert werden.

Als Datengrundlage für die Vorhersagen werden ausschließlich frei verfügbare Wetterdaten verwendet. Es ist ebenfalls ein Ziel dieser Arbeit, die Verwendbarkeit dieser Daten als Ersatz oder Ergänzung zu proprietären Daten aufzuzeigen. Die Mittelmeerinsel Sardinien wird als repräsentatives Untersuchungsgebiet verwendet. Weiterhin dient die Insel Kreta als Testanwendungsgebiet, um die Übertragbarkeit der Vorhersagemethodik auf andere Mittelmeerregionen zu gewährleisten.

Diese Methodik wird im Weiteren in eine autonom funktionierende, web-basierte Vorhersageplattform implementiert, die tägliche Waldbrand-Gefahrenlandkarten für das Untersuchungsgebiet bereitstellt. Die Konzeption und Realisierung dieser Plattform wird in dieser Arbeit detailliert beschrieben.

Um das Vorhersagesystem zu kalibrieren, wird in einer Langzeit-Studie das Vorkommen von Waldbränden auf Sardinien und Kreta für den Zeitraum von 2001 bis 2010 untersucht. Parallel werden für jeden Tag in diesem Zeitraum aus archivierten Wetterdaten rückblickend Waldbrand-Gefahrenkarten erstellt, die dann mit den tatsächlich eingetretenen Brandereignissen verglichen werden. Als Datengrundlage für die historischen Wetterbedingungen dienen Daten aus dem *Weather Underground* - Netzwerk, die von Privatpersonen bereitgestellt werden. Als Informationsquelle für aufgetretene Brände werden MODIS

Hotspot - Daten verwendet. Diese wurden zuvor mit Hilfe weiterer Datensätze zu Landnutzung und verbrannten Gebieten validiert und gefiltert.

Die Abschätzung tatsächlicher Waldbrandgefahr auf der Basis von Wetterinformationen geschieht mit Hilfe des kanadischen *Fire Weather Index* (FWI), eines generellen Indikators für die Wahrscheinlichkeit des Auftretens von Bränden. Für die Einordnung der FWI - Ergebnisse in Gefahrenklassen - von '*very low*' bis '*extreme*' - werden zwei unterschiedliche Schemata verwendet: Zum einen das des *European Forest Fire Information Systems* (EFFIS) und zum anderen ein in dieser Arbeit entwickeltes Schema, das auf den Ergebnissen der Langzeit-Studie beruht und speziell für die Verwendung auf Sardinien angepasst ist.

Beide Klassifikationen werden hinsichtlich der erzielten Ergebnisse verglichen. Als Beurteilungskriterien dienen die Rate der Fehlalarme sowie die Trefferquote (Typ I und Typ II - Fehler).

Die Ergebnisse zeigen die Überlegenheit der hier erstellten Klassifikation hinsichtlich der Vermeidung von Fehlalarmen: Während es mit dem neu entwickelten Schema in nur 14.8% der Fälle zu einem falschen Alarm kommt, beträgt dieser Anteil bei Verwendung des EFFIS - Schemas mit 34,2% mehr als das Doppelte. Allerdings liefert die EFFIS - Methodik eine bessere Trefferquote: Nur in 2% der Fälle kam es zu Bränden in Gebieten, die als ungefährdet klassifiziert wurden. Bei der hier entwickelten Klassifikation liegt diese Fehlerquote bei 13,3%. Dieser Wert ist hauptsächlich der noch unzureichenden Modellierung von Feuchtigkeit in Brennmaterialien geschuldet. Diese sollte optimiert werden, damit das System ein zutreffenderes Bild der tatsächlichen Feuergefahr liefern kann.

Das funktionsfähige System wurde im Sommer 2012 gegen die von EFFIS veröffentlichten Vorhersageergebnisse getestet. Der Test zeigt eine im Vergleich zu EFFIS gemäßigtere, aber realistische Vorhersage der Feuergefahr.

Tägliche Vorhersageergebnisse des Systems für die nächsten zehn Tage sind online verfügbar unter:
http://www.sardinia-fireweather.org
Die Gefahrenkarten sind auch in Form eines Web Map Services (WMS) oder über eine mobile Anwendung nutzbar.

Die Implementierung selbst kann kostenlos heruntergeladen sowie gemäß den Bestimmungen der GNU GPL verändert und weitergegeben werden. Alle für den Betrieb der Vorhersageplattform benötigten Skripte finden sich zudem im Anhang dieser Arbeit.

Contents

Nomenclature

Wildfire / Weather

AFM	Accessory Fuel Moisture
BUI	Build Up Index
CFFDRS	Canadian Forest Fire Danger Rating System
DC	Drought Code
DMC	Duff Moisture Code
EMC	Equilibrium Moisture Content
FBP	Fire Behaviour Prediction
FDRS	Fire Danger Rating System
FFMC	Fine Fuel Moisture Code
FMC	Fuel Moisture Content
FOP	Fire Occurrence Prediction
FWI	Fire Weather Index
ISI	Initial Spread Index
OWS	Official Weather Station
PWS	Private Weather Station

ROS	Rate of Spread

Software / Licenses

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CC BY-NC-SA	Creative Commons Attrib.-Noncommercial-Share Alike License
CC BY-SA	Creative Commons License
CMS	Content Management System
FTP	File Transfer Protocol
GDAL	Geographic Data Abstraction Library
GML	Geography Markup Language
GMT	Generic Mapping Tools
GNU	GNU's Not Unix (recursive acronym)
GPL	General Public License
GRASS	Geographic Resources Analysis Support System
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LGPL	Lesser General Public License
MIT	Massachusetts Institute of Technology
ODbL	Open Database License
QCOW	QEMU Copy On Write
QEMU	Quick Emulator

QR	Quick Response (code)
RSS	Really Simple Syndication (originally: Rich Site Summary)
URI	Uniform Resource Identifier
XML	Extended Markup Language

GIS / Remote Sensing

AVHRR	Advanced Very High Resolution Radiometer
CORINE	Coordination of Information on the Environment
EOSDIS	Earth Observing System / Data and Information System
EPSG	European Petroleum Survey Group
FIRMS	(Maryland) Fire Information for Resource Management System
FOSS	Free and Open Source Software
GCM	General Circulation Model
GIS	Geographical Information System
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDW	Inverse Distance Weighting
INSPIRE	Infrastructure for Spatial Information in the European Community
MLR	Multiple Linear Regression
MODAPS	MODIS Adaptive Processing System
MODIS	Moderate-Resolution Imaging Spectroradiometer
NDVI	Normalized Difference Vegetation Index

NIR	Near Infrared
NOAA	National Oceanic and Atmospheric Administration
OGC	Open Geospatial Consortium
RS	Remote Sensing
SDI	Spatial Data Infrastructure
SRTM	Shuttle Radar Topography Mission
TIFF	Tagged Image File Format
UTM	Universal Transverse Mercator
WFS	Web Feature Service
WGS84	World Geodetic System (1984)
WMS	Web Map Service

Institutions

CFS	Canadian Forest Service
DG ENV	Directorate General for Environment of the European Commission
DPA	Deutsche Presseagentur (news agency of Germany)
DWD	Deutscher Wetterdienst (weather service of Germany)
EC	European Commission
EU	European Union
FAO	Food and Agriculture Organization of the United Nations
FAS	Foreign Agricultural Service
IPCC	International Panel on Climate Change

JRC	Joint Research Centre
NASA	National Aeronautics and Space Administration
WWF	World Wildlife Fond

Chapter 1

Introduction

1.1 Objectives of thesis

Wildland fire presents an enormous risk to both the living world and material goods. Every year, millions of hectares of wood-, grass-, bush- and shrublands are affected by burnings around the world. While the United States, Australia, and Brazil are the countries most severely affected by wildland burnings (Pyne, Andrews and Laven, 1996), it is the Mediterranean countries that form the critically endangered region in Europe. For this thesis, the Isle of Sardinia has been chosen exemplary as the region of study, because it exhibits a fire regime typical for Mediterranean areas, and because there is a sufficient number of weather stations contributing to the Weather Underground network available. Crete was chosen as a testing region because it features different conditions compared to Sardinia regarding climate, topography, and direction (East-West instead of North-South), while at the same time also being a Mediterranean island with a distinct fire regime.

The main objective of this thesis is the spatio-temporally differentiated prediction of wildfire danger in the Mediterranean area. A secondary aim is the improvement of prediction quality compared to what is currently available for the Mediterranean region, in terms of reliability, spatial accuracy and duration of forecasting period. Regarding reliability, a fire danger classification scheme especially

adapted to the fire regime on Sardinia is to be developed using k-means cluster analysis. As a base, a long-term study contrasting retrospectively computed fire danger conditions and actually occurred fires on this island is to be conducted. It will further be demonstrated that conventionally used proprietary weather data can be replaced by crowd sourced data in the computation of wildfire danger forecasting. The last objective is to show that the developed procedural method is portable to the whole Mediterranean area. It is for this purpose that a testing region is used.
To reach these objectives, fire danger maps have been calculated for Sardinia and Crete for every day between 2001 and 2010 using the Fire Weather Index (FWI) component of the Canadian Fire Weather Index System (CFFDRS) in combination with crowed sourced weather data from the Weather Underground network. The distribution of resulting index values has been analysed and used for a statistically based classification, complying with the fire danger situation on Sardinia. The classification has been tested regarding type I and type II errors, and implemented into an automated web platform, providing forecasting maps for Sardinia for ten days in the future. The results have then been evaluated in comparison to the fire danger warning website of the European Forest Fire Information System (EFFIS).

This thesis was written with the aim in mind that the work should be usable in the field of fire forecasting to local organisations in the Mediterranean area. While in many implementations of the FWI there are little or no adaptations to the actual region of operation, a fire danger classification especially suited for the conditions on Sardinia is developed in this case, serving as a base for the forecasts. This thesis differs from other works in this research field by its use of freely accessible, crowd-sourced input data instead of proprietary data from official weather service stations. Thus, in contrast to other systems developed in this field that require huge development costs, expensive input data and scientific knowledge for calibration and maintenance, the here proposed systems is designed to work autonomously at a minimal cost level. Tests confirm that the reliability of forecasting results are comparable to the outputs of EFFIS.
Next to Sardinia, the forecasting web platform is adaptable to other regions of

interest in the Mediterranean, and more importantly, is free to download, use and modify under the terms of the GNU General Public License (GPL). This is possible since only Free and Open Source Software (FOSS) has been used to develop the platform, including the operating system, and only freely accessible data was utilised. The complete system is available as a virtual machine, which can easily be integrated into an existing server infrastructure.
Governments worldwide increasingly promote Free and Open Source Software as a move towards transparency, which is interlinked with the freedom to run, copy, and improve the used applications (Hahn, 2009). The European Commission (EC) also supports the development of free software (Kroes, 2010). In addition to financial savings, free software is progressively rewarded in the science-policy interface due to the emerging paradigm of 'open science' (San-Miguel-Ayanz, 2013, following Stallman, 2005; Cai, Judd and Lontzek, 2012).
The development of a specially adapted classification scheme based on validated fire occurrence data over an extended timespan of ten years, together with the open approach allowing the adaptation and reusing of the complete online fire weather forecasting system implementation for own needs and regions of interest, is believed to be unique and hence, distinguishes this work from the works previously undertaken in this domain.

The thesis begins with a general overview about wildfire, including one section that focuses on the situation in Southern Europe.
Chapter 2 deals with the broad field of fire science, and the status of current research in fire danger rating using automated systems. A closer look is taken at the Canadian Forest Fire Weather Index System, which forms the basis of the computations performed in this thesis as well as of the EFFIS platform. EFFIS itself is portrayed in the following subsection.
Chapter 3 lists and describes the datasets used for this thesis, with special attention given to the Weather Underground data, which is the feeding source for the fire danger calculation in this thesis. Additional information is given on the applications and libraries used in this work, considering that only Open Source software is used which in part may not yet be popular in the field of Geography. Chapter 3 also covers the general approach in detail.

The realisation of the fire danger forecasting system, including the technical implementation on a Linux server are described in Chapter 4. It begins with the process of selection of the hotspots for validation purposes and the determination of a classification scheme for the FWI system adequate for Sardinia. This is followed by a description of the process for automated creation of the fire danger forecasts.
Chapter 5 demonstrates the results by presenting and describing figures and tables of the type I and type II - error tests. The two classifications used are compared, both in terms of the long-term study and the tests in the summer of 2012. In addition, average computed area sizes of fire danger levels are shown for both classification schemes.
Chapter 6 presents the conclusions regarding the achievements of this thesis.
The appendix lists the scripts of major importance written for the operation of the forecasting platform, implemented in `Python`, `Bash` and `R` source code, respectively.

1.2 Wildfire

1.2.1 The phenomenon of wildfire

Terminology

- **Fire** is a self-preserving reaction of oxidation, running at high temperatures and producing heat and energy. It mainly depends on the availability of fuel and oxygen (Rossotti, 1994).
- **Vegetation fire** is defined by the Food and Agriculture Organization of the United Nations (FAO) to address fires in forests, other wooded land, rangelands, grasslands, bushlands, agricultural lands and barren land (FAO, 2007). They have to be distinguished from fires intentionally applied in land management, such as the ones in prescribed burning or slash-and-burn agriculture (FAO, 2007).
- **Wildfire** is a term defined by the FAO as an unplanned and/or uncontrolled vegetation fire, regardless of ignition source, damage or benefit (FAO, 2010).

- **Prescribed burning / planned fire** are terms describing, contrary to a wildfire, a vegetation fire (regardless of ignition source) that burns according to management objectives and that requires limited or no suppression action (FAO, 2010).
- **Fire risk** determines the chance of fire ignition, as affected by the manner and occurrence of causative forces such as human activities (FAO, 2010, following Pyne, Andrews and Laven, 1996).
- **Fire behaviour** is a term addressing the rate of spread, intensity, magnitude, and direction of a fire (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011).
- **Fire hazard** enfolds the potential fire behaviour of a fuel complex, which is based on its physical and chemical characteristics (FAO, 2010, following Pyne, Andrews and Laven, 1996).
- **Fire regime** is a generalised description of the role fire plays in an ecosystem (Miller, 2003, following Agee, 1993). The term incorporates the fuel type, temporal nature, spatial pattern and the consequences of fire occurrence in a given location (Bowman et al., 2009).
  - **Fire frequency** addresses the regularity of fire occurrence. The frequency of fires has a strong influence on species composition, because selection pressure will favour organisms that can best adapt to the intervals of fire activity (Flannigan, Stocks and Weber, 2003).
  - **Fire size** means the extent of fire occurrence, which addresses the spatial dimension of the burned area (Flannigan, Stocks and Weber, 2003).
  - **Fire seasonality** specifies the predominant time of year when fires occur. The season affects fire intensity through differences in fuel amount and moisture content (Flannigan, Stocks and Weber, 2003).
  - **Fire intensity** represents the amount of energy or heat released per unit length of fire line. The intensity can vary considerably, being reliant on the fuel type (Flannigan, Stocks and Weber, 2003).

- **Fire severity** describes the depth of burn into the surface soil layers. Fire severity influences roots and soil seeds and thus the reproductive plant fitness (Flannigan, Stocks and Weber, 2003).
- **Fire type** addresses the assignment of the height level of usual fire occurrence, referring to ground, surface and crown fires (Flannigan, Stocks and Weber, 2003). Van Wagner (1983) defines the following five main kinds of fire types (with increasing amount of energy released, measured in kilowatts per unit of the front line; Wagner, 1983):
  - Smouldering fires in deep organic layers: less than 10 kW/m
  - Surface backfires (burning against the wind): 100-800 kW/m
  - Surface head fires (burning with the wind): 200-15,000 kW/m
  - Crown fires: 8,000-40,000 kW/m
  - High-intensity spotting fires: up to 150,000 kW/m

Wildfire is a worldwide phenomenon, which was detected in geological records soon after the appearance of terrestrial plants. It affects ecosystem patterns and processes, influencing regional and global biochemical cycles and controlling the distribution and structure of vegetation. It is thereby the dominant disturbance regime in both boreal and temperate forests as well as in tropical biota and oceanic shrublands (Flannigan, Stocks and Weber, 2003). Fire also impacts the climate through the release of carbon during burning (Bowman et al., 2009). The analysis of worldwide satellite data from the year 2000 by the Joint Research Center (JRC) of the European Commission revealed that 350 million hectares of land were affected by wildfires in that year, mostly located in sub-Saharan Africa and Australia (FAO, 2007). Wildfires occur in all vegetation zones with increasing trends in several regions regarding number of detections and area burned. However, it is assumed that most of the global fires are unmonitored and undocumented, so that their real numbers are probably considerably higher (Groot et al., 2006). Fire is causing major social and economic damages, including human lives lost. Among the victims are staff of fire fighting brigades as well as private persons. In 1998, for instance, 700 people were killed by fires solely in Brazil (FAO, 2007). In 2007, severe wildfires on the peninsula of the Peleponnesos resulted 80 life losses as well as 200,000 ha burned. The arisen damage costs have

been estimated to be approximately 5 billion Euro (Papadopoulos et al., 2013). Australia suffered from catastrophic wildfires in the area of Victoria in 2009, resulted in the highest losses from fire in Australian history with 173 people killed (Price and Bradstock, 2012). The occurrence of fire therefore combines physical, meteorological, biological, and social processes (Lin et al., 2000).

Three determinants are responsible for ignition and maintenance of a fire: fuels, oxygen, and one or several heat sources. The relationship of these have been named the Fire Triangle by Andrews (Pyne, Andrews and Laven, 1996). If fuels and oxygen are readily available to be consumed by a fire, then wind speed is the key factor for fire growth (Flannigan, Stocks and Weber, 2003). Regarding the natural causes, lightning is the main ignition factor (Goldammer and Furyaev, 1996).

Despite its destructive features, fire is an essential component in some ecosystems such as the pine forests in the Mediterranean or in Central America, maintaining biodiversity, productivity and ecosystem dynamics. It is also a widely used tool in land management. The heterogeneous nature of fire has to be considered by fire authorities, and a reasonable fire management should be supported for endangered regions rather than just suppression activities (FAO, 2007).

1.2.2 Causes of wildfire ignitions

Possible heat sources necessary for ignition are referred to as fire causes. A stroke of lightning or a volcanic breakout may trigger a huge wildfire, since the carbon-rich vegetation on the earth's surface together with seasonally dry climates and atmospheric oxygen render the landmasses highly flammable (Bowman et al., 2009). Within the realm of natural causes, lightning is by far the most frequent one, followed by volcanic eruptions (Goldammer and Furyaev, 1996). In Canada, lightning strokes account for 40% of wildfire occurrence. This percentage is responsible for 92% of the annual area burned (Leone et al., 2003), due to the difficulties in detection and suppression of fire occurring in remote areas. National and regional networks to sense strokes are in use in Canada and the USA using

magnetic direction finders, time of arrival techniques, or Very High Frequency (VHF) interferometry (Leone et al., 2003) to provide a near-realtime means of detection of potential new ignition areas (Allgöwer, Carlson and Wagtendonk, 2003).

In any other part of the world, however, lightning cannot be seen as a major wildfire cause. In the Mediterranean, lightning-caused wildfires account for only 0.6 to 4% of the annually burned areas (Naveh, 1975, following Susmel, 1973). Throughout Europe, the amount of lightning-caused wildfires oscillates between 1% and 2% according to national statistics (Leone et al., 2003).

As a fact, all natural causes together make up only for a small percentage in the numbers of wildfires worldwide. The main cause for fires in vegetation and agricultural areas are people by far. It was estimated that as much as 98% of wildfires originate from peoples action (Leone et al., 2003). The main reasons are land-clearing, negligence, and arson, followed by the extraction of non-wood forest products, industrial development, resettlement, and hunting (FAO, 2007). Accidental causes include electric arcs created by high voltage electric cables and sun's rays concentrated by broken glass fragments (Leone et al., 2003).

For fires started intentionally, the possible motivations include protest or vengeance against others or against the government as well as malignity. Noticeably, some arsonists are even employed at fire-fighting services. As much as 43 persons have been found guilty of arson in the Australian fire season 2002/03. However, criminal prosecution of arson is difficult to conduct; in many countries predominantly because of the lack of specific procedures (FAO, 2007).

1.2.3 Detection and suppression methods

Most countries regularly affected by fire still use watchtowers and patrols for early fire detection, yet satellite detection as well as aerial surveillance is becoming increasingly important (Pyne, Andrews and Laven, 1996). Remote sensing allows for wildfires to be detected near-realtime, and operational fire detection systems have been, amongst others, developed for use in Canada, Australia, and Finland (Allgöwer, Carlson and Wagtendonk, 2003).

In most cases, the fighting of fires is undertaken by ground-based suppression

forces. Attack is either executed in a direct manner, aiming at the extinguishing of the fire, or via an indirect approach. Two examples of the latter are backfiring - where vegetation being ahead of the flame front is burned so that the fire cannot advance any further in that direction - and the batting of firelines, i.e., the removing of fuel in a corridor that the fire cannot overleap (Pyne, Andrews and Laven, 1996). Ground-based forces are in many countries supported by airplanes and helicopters, the latter being also used for transporting fire crews to the site of operation (FAO, 2007). Aircraft costs usually represent the major part of the total fire management expenses, which in Canada sum up on average to US$450 million per year, but may reach up to twice that amount in extreme fire seasons.

1.2.4 Effects of wildfire burnings

The effects of fires, besides human and animal lives lost, include harm on health both short- and long-term, direct material losses and costs for fire suppression activities and evacuations, as well as damages to the environment such as soil degradation and CO_2 emissions (FAO, 2007).
There are also several secondary effects, e.g., soil erosion and thereby land- and mudslides, lowered biological diversity, as well as insect infestations. The latter often succeed fires (especially in the form of bark beetles in South Eastern Europe; FAO, 2007). In the Russian Federation, the financial losses due to wildfires are reported to be US$4.2 billion in 1998. In North Eastern Asia, the losses in timber are assumed to be US$0.5 - 1 billion per year (FAO, 2007). Summing up money spend by federal, state, and local agencies, it is estimated that costs for activities related to wildfire suppression in the United States are as high as two billion US$ per year (Brown, Hall and Westerling, 2004).

Fires burning in areas contaminated with radio-nuclides represent an exceptional and serious thread, since formerly bound radioactive particles can be reintroduced into the atmosphere. Almost every year, fires occur in regions polluted as a result of the Chernobyl nuclear power plant accident in 1986. Covering more than two million hectares in total, these contaminated regions are located in Belarus, in large parts of Ukraine and in Bryansk, a region of the Russian Federation.

Radioactive emissions from Central Asian wildfires were recorded in Canada in 2003 (FAO, 2007, following Wotawa et al., 2006). The situation is similar in Kazakhstan due to several hundred nuclear tests executed in that region between 1949 and 1989.

1.2.5 The role of Climate Change

Climate Change, according to the International Panel on Climate Change (IPCC), refers to an unnatural warming of the earth's atmosphere caused by the so-called greenhouse effect, a shift in the energy balance between incoming solar energy and the amount of energy radiated from earth into space. Trace gases such as carbon dioxide, methane and water vapour are able to absorb a small proportion of the outgoing radiation, which leads to a warmup of the atmosphere (IPCC, 2007b).
The occurrence of wildfires is very closely connected to prevailing weather conditions, especially temperature, precipitation, wind speed, and atmospheric moisture (Flannigan, Stocks and Weber, 2003). According to the IPCC, increased temperatures and dry periods will occur more frequently and more persistently, leading to an intensification and expansion of wildfire activity on a global scale (IPCC, 2007a). In 2008, Power et al. showed on the basis of sedimentary charcoal that climate and fire activity closely correlated in the past, in terms of cold intervals showing reduced fire activity and warm intervals showing increased numbers, irrespective of human presence (Bowman et al., 2009).

General Circulation Models (GCM) are used to simulate the response of the global climate system to increasing greenhouse gas concentrations. The release of CO_2 is known to contribute substantially to greenhouse gases and the associated global warming (IPCC, 2007b), which by itself increases the risk of extreme fire weather (Bowman et al., 2009). Biomass burning also releases black carbon aerosols, which by reason of their strong solar radiation absorption properties are likely to have the strongest effect on global warming, alongside with CO_2. The quantity of biomass burned globally each year was calculated to be 9.200 million tons (FAO, 2007, following Andreae, 2004), accounting for 3.431 million tons of

CO_2. 42 percent of the biomass burning worldwide occurs in Africa (FAO, 2007). Yet, only a marginal part of this amount remains in the atmosphere. In ecosystems with sustainable, fire-adapted regimes, the released emissions are taken up by regrowing vegetation and, subsequently, do not lead to atmospheric CO_2 increase or the greenhouse effect (Bowman et al., 2009).

Since there is no long-term, consistent and reliable data on occurrences and impacts of past wildfires, the determination of trends and the prediction of future wildfire scenarios is nearly impossible (FAO, 2007). Short-term trends of recent decades, which might not necessarily be connected with climate change, show a significant increase in the number of wildfires and in the area burned at least for the Balkans and Central and Northeast Asia (FAO, 2007).
Regarding the Mediterrenean area, Houghton et al. (1996) used a 2 x CO_2 scenario to determine climatic trends in the region and found an increase in air temperature and a reduction in summer rainfall (Pausas, 1999a, following Houghton et al., 1996), as did and Piol et al. (1998). Current predictions all suggest a growing water deficit, resulting in increased water stress and fuel conditions and consecutively, changes in the fire regime (Pausas, 1999b). Moriondo used the IPCC scenarios A2 and B2, which represent medium-high and medium-low greenhouse gas emissions, respectively, together with the Fire Weather Index to determine trends in fire danger in the Mediterranean. He determined a general increase in fire danger with both future scenarios over the whole region. More precisely, he detected an increase in the number of years with fire danger, in the length of season with fire danger, and in the number of extreme events. Moriondo calculated the increase in Fire Weather Index results between 16% and 25% in A2, and between 11% and 18% in the B2 szenario. It therefore has to be concluded that Mediterranean ecosystems will probably be affected by increases in area burned as well as in fire severity (Moriondo et al., 2006). This, in turn, will most likely result in the occurrence of floods, soil erosion, and consequently loss of fertility (Moriondo et al., 2006, following Court-Picon et al. 2004, Iliadis 2004). Mouillot (2002) calculated that, due to climate change, the time interval between two successive fires in the Mediterranean decreases from 20 to 16 years in shrubland vegetation, and from 72 to 62 years in forest stands. As he points

out, an increase in fire frequency and the consecutive domination of shrublands would further increase deep drainage and runoff, and through that decrease the yield of water (Mouillot, Rambal and Joffre, 2002).
However, in the study undertaken in this thesis covering a ten-year research period, such a long-term tendency could neither be detected for Sardinia nor for Crete.

Wildfire activity in connection to ENSO events

Global wildfire activity should also to be considered in relation to the El Niño phenomenon, an anomalously warm ocean current along the South American coastline, often associated with a much more extensive warming of the Pacific basin. Its counterpart, La Niña, coincides with a basinwide cooling of the tropical Pacific (Trenberth, 1997).
Dendrochronical analyses indicate a strong correlation between high fire activity and climate oscillations on interannual and decadal scales. A significant increase of ignitions occurs in tropical rainforests of South America during El Niño phases, whereas during times of La Niña the most severely affected regions are the southern United States and Patagonia in Argentina (Bowman et al., 2009). During the El Niño events in 1992/93, 1997/98, and 2002/03, the whole South American continent experienced severe droughts and widespread wildfire activity (FAO, 2007).

1.3 Wildfire situation in Southern Europe

Wildfires in Europe have caused enormous losses in terms of human life and environmental damage in recent decades (San-Miguel-Ayanz et al., 2002), 110 lives were lost during the 2007 wildfires in the Mediterranean area alone. Figure 1.1 on page 15 provides an overview on losses of area due to burning, as well as number of fires, both for Italy and Greece, from 2000 to 2011. Losses of human lives are additionally given in brackets for each year.
On average, 7,000 km^2 to 10,000 km^2 of forested Mediterranean area are burned per year (San-Miguel-Ayanz et al., 2003c). 5,000 km^2 of this amount are located

in the Southern European states of Portugal, Spain, France, Italy, and Greece (JRC, 2012), which are the five most affected European countries.

Natural fires have always been an integral part of Southern European ecosystems, especially in the Mediterranean area. A fairly new problem, however, is the general trend of a significantly increasing number of people migrating from rural areas to the cities (FAO, 2007). This applies in particular to young people. As a result, capable helpers lack in case of an occurring fire in rural areas, thus increasing the risk of the fire running out of control (FAO, 2007). Also, there is a shortage of workers in many pastoral areas for fire prevention activities. Preventive burning and controlled grazing are strategies that have successfully been used in the Mediterranean for a long time to reduce the fuel load and, therefore, the potential fire intensity (FAO, 2007).
Additionally to those fires initiated by natural causes, the number of human caused fires are increasing. This is due to high population density in suburban areas and the extensive use of forest regions as recreational zones, where fires often start as a result of negligence or accident. European forests are also often crossed by electricity cables, railways and road networks that further increase the risk of fires (San-Miguel-Ayanz et al., 2003c). In addition, a considerable number of fires are executed on purpose. In Italy, human caused fires accounted for 78.9% of the total number in the 2007 fire season. Fires due to arson have been 65.5%, whereas negligence only constitutes for 13.4% of fires induced by man. In total, the proportion of human caused fires in the Mediterranean might well be 95% (FAO, 2007). In this area in particular, the most frequent motivation for intentionally started fires is the search for profit, this includes renewing pastures, recovering land for crops, saving labour costs, and construction speculation (JRC, 2008). The renewing of pastures is a traditional and common practice of shepherds in the Mediterranean area, and is by far the major cause for deliberately set fires in that region (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011). Forest fire crimes are severely persecuted. 455 people were reported in Italy to the court of justice by the territorial garrison of Italian Forest Corps within the 2011 fire season, including nine persons taken under arrest for fire arson. These correspond to 7.9% of the number of deliberately caused fires (JRC, 2012).

During 2011, fires in the five most affected European countries burned a total area of 2,690.81 km^2, which is below the long-term average. The figures are over all declining since 2007, when Greece experienced the worst year of forest fires ever recorded. Huge wildfires at that time devastated the Greek peninsula of Peleponnesos, killing 80 people, 69 of them being civilians (JRC, 2008). Beside 2007, there were two more fire-intensive years during the last decade in Southern Europe: 2005 with over 6,000 km^2 of area burned, and 2003 with even more than 7,500 km^2. The latter being number three in area burned since the beginning of accurate fire related record keeping in Europe in 1980 (JRC, 2012).

The number of fires occurred in the most affected European countries (France, Greece, Italy, Portugal, and Spain) has been significantly increasing in recent years with over 55,000 detections in 2011. In the 1980s the numbers of fire detections were below 30,000 on average. This increase may however also be due to continuing improvements in the recording systems, allowing for more smaller fires to be detected. The total size of burned are might therefore represent a better indicator for trends in fire activity. Portugal and Spain have been the most severely affected European countries in the past decade regarding burned area, followed by Italy and Greece. The two latter, though, still exhibit problematic tendencies regarding wildfires. Italy shows about 80,000 ha burned area in 2011, as much as Portugal and only little less than Spain. For Greece, the average fire size in 2011, with 17.5 ha being seven times the size of an average fire in Portugal, is alarming and indicates insufficient fire suppression activities (JRC, 2012).

The wildfire occurrences of past years were accompanied by increasing public and political awareness on the issue of natural disasters, so that now a wide range of activities is undertaken to address the problem. Greece, for example, today maintains a fire brigade of about 8,900 persons operating 1,700 engines, with an additional seasonally hired personnel of about 5,400 persons just for forest fire suppression actions. 44 aircrafts and two helicopters support the fire fighting activities (JRC, 2012).

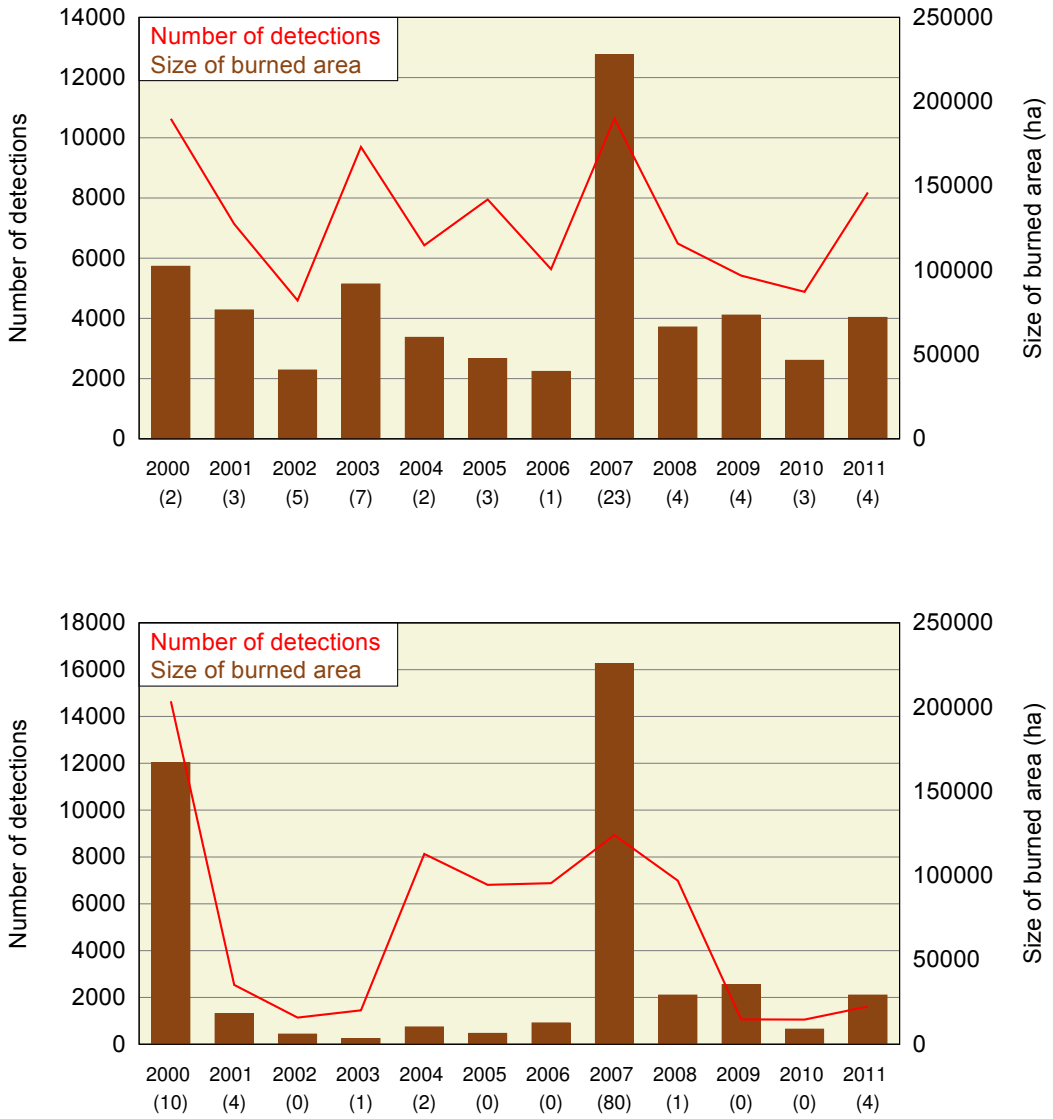


Figure 1.1: Wildfire situation in Italy (top) and Greece (bottom), 2000 - 2011

Losses of human lives for each year given in brackets

Sources:
JRC, 2001 / JRC, 2002 / JRC, 2003 / JRC, 2004 / JRC, 2005 / JRC, 2006 / JRC, 2007 / JRC, 2008 / JRC, 2009 / JRC, 2010 / JRC, 2011 / JRC, 2012

While fires that mean a hazard to people fall within the competence of national fires services, there is a strong collaboration between the Mediterranean countries in case of emergency. For coordination, the member states are organised in a networking arrangement named 'FAO Silva Mediterranea' (FAO, 2007).

In connection with the flammability of the vegetation and the socio-economic conditions, the regional climate yields favourable circumstances for large forest fires and days of great fire potential (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011, following Dimitrakopoulos et al., 2001). The fire frequency in Mediterranean ecosystems is typically high with large areas burned (Mouillot, Rambal and Joffre, 2002, following Moreno, 1998).
In the future, an increase in the frequency of severe weather conditions in the Mediterranean area is expected by local fire authorities in relation to climate change (FAO, 2007). Higher air temperatures and reduced summer rainfall are likely to further increase the risk of extreme fire weather during summer months.

1.3.1 Climatic conditions with respect to wildfire

Mediterranean fire bioclimates typically feature long, hot and dry summer seasons with maximum average daily temperatures of around 30 °C. Relative air humidities are generally between 50% and 60%. Heat waves in combination with strong, dry winds like 'Chamsin' in the eastern and 'Shirocco' in the western part of the Mediterranean basin occur frequently in the beginning and the end of the dry season, causing extended periods of droughts. Under these circumstances, maximum temperature values rise above 40 °C and relative humidities can drop to less than 30% (Naveh, 1975). The winters are comparatively mild, and rainfall is concentrated during winter months (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011, following McCutchan, 1977). Total annual rainfall ranges from less than 300 mm up to 900 mm. It is irregularly distributed within as well as between years (Pausas, 1999a). In most regions, moisture is the most important limiting factor for plant growth. Mediterranean regions feature a moderate marine-air influence all year.

1.3.2 Mediterranean vegetation ecology

Landscapes in the Mediterranean Basin differ in various aspects from those of the rest of Europe, mainly due to differences in climate, the role of fire, and long as well as intense human impact. Disturbances and changes in the prevailing disturbance regimes may even be more important for the shaping of the landscapes as climatic effects or the ones related to CO_2 (Pausas, 1999b). Since the 1950s, fire occurrence in the Mediterranean has increased significantly and has become the primary factor of disturbance (Marchetti, Ricotta and Volpe, 1995, following Ministero Agricoltura e Foreste, 1993). Fire strongly impacts the richness and distribution of plant species and therefore ecological systems (Marchetti, Ricotta and Volpe, 1995, following Robinson, 1991). In addition, areas having been affected by fires are vulnerable to erosion and hydrogeological events. This is of major importance since particularly in the Mediterranean region, the reestablishment of original plant formations is a slow and easily reversible process (Marchetti, Ricotta and Volpe, 1995). A large number of fire occur in the region annually, destroying between 700 x 10^3 and 1000 x 10^3 ha of forest on yearly average. This causes enormous economic and ecological losses (Moriondo et al., 2006, following Velez, 1997).

On the other hand, however, wildfire is also an important stress factor and selective force in ecosystems of the Mediterranean, contributing to biological diversity of genotypes as well as to composition, structure, productivity, and nutrient circulation of plant communities (Naveh, 1975). It must therefore be seen as an integral part of Mediterranean ecosystems and their evolution (Naveh, 1975).

Plant communities in the Mediterranean are characterised by high structural heterogeneity and complexity (Arca et al., 2007). Mediterranean vegetation includes forests and open woodlands, heathlands and dense shrublands as well as open scrublands. In wooded areas, pines and evergreen oaks are the most common species (Pausas, 1999a).

Since wildfires have always been an important factor in ecosystems of the Mediterranean, many plant species have developed strategies allowing them to survive periodic fires (Pausas, 1999a, following Naveh, 1975). The cork-oak (Quercus suber), which is common on Sardinia, is able to sprout from epicormic stem buds

after a fire, with the canopy recovering in a very short period. These resprouting species in general recruit quickly due to maintenance of alive below-ground biomass. Several species exhibit germination stimulated by fire, including a variety of leguminous species (such as Ulex parviflorus and Hippocrepis unisiliquosa, for example; Pausas, 1999a, following Doussi & Thanos 1994), and Cistaceae species (such as Cistus sp. pl. and Fumana sp. pl., for example; Pausas, 1999a, following Thanos et al., 1992). The stimulation process of these species is complex, but high temperatures are the key factor. Serotiny, the reservation of the seed in the canopy until the occurrence of a fire, is, however, rare in the Mediterranean basin (Pausas, 1999a). It is featured mainly in two pine species, Pinus halepensis Mill. and Pinus brutia Ten., which are found at low altitudes (Fyllas, Dimitrakopoulos and Troumbis, 2008). In contrast to regions prevailing similar conditions (such as Chile and California), no plants that strictly depend on fire for completing their life cycles are found in the region.

Post-fire weather conditions are of primary importance for the regeneration of vegetation after fire. Temperature and moisture conditions must be convenient for germination and resprouting, while excessive precipitation in burned areas might lead to intensified soil erosion and the loss of soil and seeds (Pausas, 1999a). This desertification process is, however, quite common, mainly in the southern part of the Mediterranean, due to both natural and human influence such as deforestation, overgrazing, urbanisation, and pollution (Detto et al., 2006).

While natural disturbances such as wildfires play a significant role in the development of Mediterranean vegetation patterns, the influence of anthropogenic activities is at least of the same importance. Burning, cutting, grazing on non-arable lands as well as clearing, terracing, and cultivating on the arable ones have historically formed a most human-influenced landscape in most of the Mediterranean basin. Today, abandonment of traditional rural uses is common in many Mediterranean vegetation areas, changing fire regimes through the increase in fuel loads (Chuvieco and Riva, 2009). Fuel buildup, possibly in connection with total fire suppression is known to have possibly fatal consequences (Allgöwer, Carlson and Wagtendonk, 2003). This fuel buildup then leads to fires of increased size and intensity and consecutively to an intensification of negative impacts on soils and vegetation resilience (Chuvieco and Riva, 2009). Pausas (1999) found a di-

rect connection of rural depopulations to increase in the annual number of fires and total surface burned. Prescribed fires, along with mechanical treatments, are often used to reduce fuel loads (Brown, Hall and Westerling, 2004). In Australia, changes in fire frequency have been found to have major impacts on the composition, the age-distribution and the biomass of ecosystems (Williams, Karoly and Tapper, 2001). Based on the principles of fire ecology, many fire endangered regions have adopted a more flexible approach to fire management (Chou, 1992). Fires are increasingly regarded as part of natural ecosystems, and are consecutively not suppressed if they burn within certain constraints (Allgöwer, Carlson and Wagtendonk, 2003) and do not pose threats to human lives or property.

1.3.3 The 2012 fire season

Figure 1.2: Clouds of smoke on Sardinia, caused by a wildfire broken out on July 15th 2012

Source: tagesschau.de, 2012 / dpa (2012/02/17)

In the summer of 2012, several regions in Southern Europe were affected by severe wildfires. Maximum temperatures of around 40 °C, extreme dryness from the

mid of May to the beginning of September and heavy winds led to fire conditions hardly controllable by the fire brigades. An area of 3,000 ha was destroyed on the Spanish island Tenerife. On La Palma, where 500 ha burned down, 200 people had to be evacuated. The Greek peninsula Peleponnesos had to declare state of emergency because of fires threatening the municipalities of Ano and Kato Kastritsi. In Montenegro, several fires had to be combated near the capital Podgorica (tagesschau.de, 2012 / dpa).

Italy was also affected; fires burned near Trapani on Sicily, in the region of Gargano, and in the area of Benevento. However, the most severe burnings happened on the island of Sardinia, where 600 ha of land were destroyed. 800 people had to leave houses and hotels to seek shelter (focus.de, 2012 / dpa). Authorities on Sardinia declared maximum alert (euronews.com, 2012). Firefighting planes and helicopters have been utilised to fight the fires, mainly burning at the northeastern coast of Sardinia at San Teodore, in the province of Olbia-Tempio. Five fire fighters were injured during the operations (focus.de, 2012 / dpa). The nature conservation organisation World Wildlife Fond (WWF) assumed arson to be the reason for the major part of the burnings, and accused the Italian mafia of being responsible, with clearing of land to be the motive (tagesschau.de, 2012 / dpa).

In the following days, 1,000 ha of macchia, a local expression describing shrubs degenerated through overburning, overgrazing, and soil erosion (Pyne, Andrews and Laven, 1996) burned in central Sardinia in the province of Nuoro between Ottana and Bolotona (sz-online.de, 2012 / dpa). Some minor fires occurred in southern Sardinia near the capital of Cagliari, threatening a military airport (zeit.de, 2012 / dpa).

In total, the Italian forestry agency encountered a surprising increase in wildfire occurrence of 76 percent in 2012, compared to the previous season (sz-online.de, 2012 / dpa).

1.4 Areas of Interest

1.4.1 Study area: Sardinia

Sardinia is the second largest island of the Mediterranean. It is located in the Western Mediterranean Sea between 38°52' and 41°15' North and 8°8' and 9°51' East, covering an area of 24,089 km^2. Its population amounts to roughly 1,640,000 inhabitants, a quarter of this number being situated in the metropolitan area of Cagliari, the islands capital. Sardinia features the typically Mediterranean climatic conditions with dry summers which are strongly influenced by the Azores Anticyclone regime (Delitala et al., 2000), and concentrated rainfall in autumn and winter. The climate is semi-arid, with average annual rainfall ranging from below 500 mm in the South to 900 mm in the inner, hilly areas (Vacca et al., 2002).

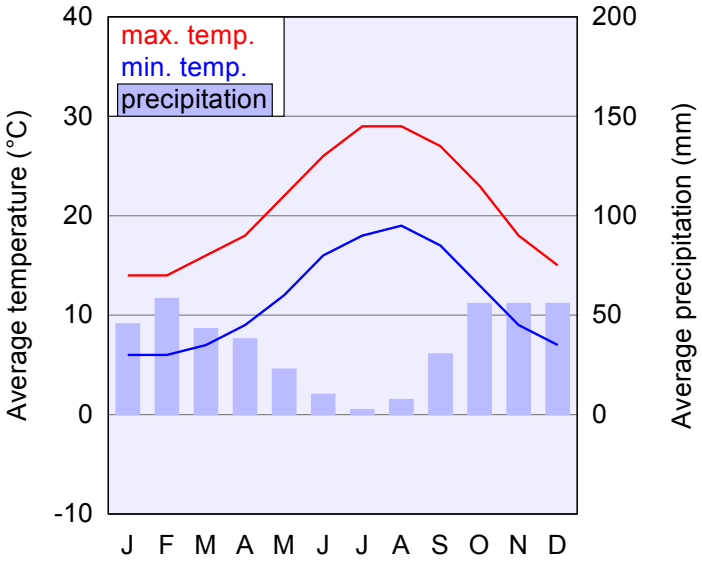


Figure 1.3: Monthly temperature and precipitation averages of Sardinia
Source: Own figure, data taken from The Weather Channel, LLC weather.com, 2012b

Precipitation is sporadic, typically peaking in November/December, and often occurring in combination with Alpine cyclogenesis (Delitala et al., 2000, following Tibaldi et al., 1999). The total amount of precipitation may vary greatly from year to year (Delitala et al., 2000). Figure 1.3 shows monthly temperature and

precipitation averages. Sardinia's orography is complex, exhibiting many mountains and valleys. The relief is due to Sardinia's geologically long-term tectonic activity, combined with irregular fluvial and aeolian erosion (Chessa and Delitala, 1997). The major ranges, of which two out of three are located in the eastern part of the island, belong to the Sardinian-Corse Mountain System, which extend from the North of Corsica to the South of Sardinia (Chessa, Cesari and Delitala, 1999). The highest peak is Punta La Marmora with 1,834 meters, which is part of the Gennargentu Range in East-Central Sardinia (Chessa and Delitala, 1997). Sardinia's topography is illustrated in figure 1.4. Mountains and hilly areas account for 86,4% of the islands surface. In consequence, soils exhibiting adequate farming conditions account for only 18% of the surface, with further 54% showing different degrees of suitability with respect to landscape morphology and vegetation cover (Vacca et al., 2002, following Aru et al., 1991). Sardinia suffers from broad desertification processes produced by fires and climate variations as well as by human influence, comprising deforestation, overgrazing, urbanization, pollution, and again fires (Detto et al., 2006).

The prevailing land cover types are grassland and grazing land, accounting for almost 40% of the area. 'Maccia', a degraded vegetation typical for the Mediterranean mostly used as pasture area, covers 20% of the surface. Further 10% are vegetated by hardwood forests (Vacca et al., 2002), of which Cork oak (Quercus suber L.) accounts for 37.5%, corresponding to 90,000 ha. Cork forms an economically important good of Sardinia (Vacca, 2000, following Sanfilippo and Vannelli, 1992). Tourism is developing into a major industries (Edelsward and Salzman, 1996). About 25,000 species of plants are found on the island, as well as 75% of the insects in Europe together with a high number of endemic species (Bodini and Cossu, 2010).

Wildfires represent an important problem on Sardinia, both economically and socially. The major part of Italian drought season wildfires occurs on this island (Sirca et al., 2007, following Bovio and Camia, 1997). Sardinia experienced a disastrous fire in 1989 in the Olbian region. Driven by strong southwestern winds, the fire ravaged for three days and burned over 8,000 ha. The burnings caused many fatalities, agripastoral areas and woodlands were devastated (Marchetti, Ricotta and Volpe, 1995). Apart from disasters such as this, Sardinia experiences

a high frequency of moderate wildfires as it is typical in the Mediterranean area (Mouillot, Rambal and Joffre, 2002), with burnings occurring every year to every second year.

The total vegetated area on Sardinia covers 12126.12 km^2, which accounts for 50.34% of the islands total area. This areas potentially endangered by wildfire comprise natural grasslands, sclerophyllous vegetation (mostly maccia), broad-leaved forest, coniferous forest, mixed forest as well as moors and heathland.

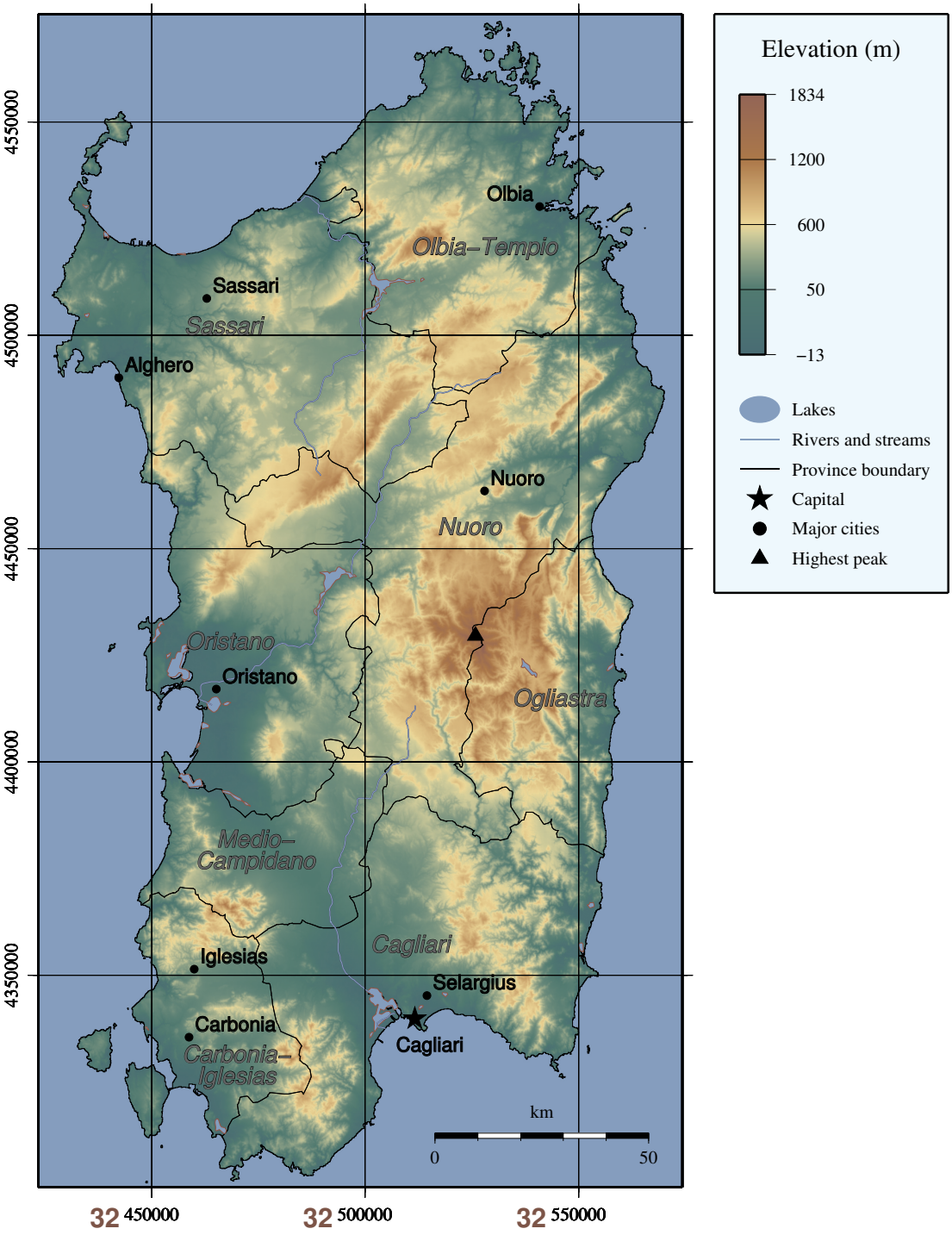


Figure 1.4: Studysite: Sardinia

Source: own map, based on:
Elevation: [SRTM41] Jarvis et al., 2008
Rivers/Lakes: [GSHHS] Wessel and Smith, 1996
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012
Cities: [geonames.org] geonames.org, 2012

1.4.2 Testing area: Crete

The Isle of Crete is located in the Southeast of the Mediterranean Sea, between 34°45' and 35°45' North and 23°30' and 26°20' East. Its surface covers an area of 8336 km^2. The population comprises 600,000 people, 30% of which live in Heraklion, the capital. The landscape is mostly mountainous, since the island is the continuation of the Greek mainland mountain ranges, and exhibits only a few plains in the coastal zones. This is where the most important agricultural sites are located (Chartzoulakis and Psarras, 2005). Figure 1.5 gives an impression of the topographic features of Crete. The highest peak is Mount Ida with a height of 2,456 meters (Columbia Encyclopedia, 2012). The climate is sub-humid Mediterranean, characterised through long, hot and dry summers contrasted to comparatively humid and cold winters. The island lies between the 18.5 and 19.0 °C isothermes, however, monthly mean temperatures vary significantly. Figure 1.6 shows the monthly mean temperature and precipitation. 70 to 80% of annual rainfall occurs during three to four winter months. Precipitation is highest in the Northwestern mountainous areas, where it can reach 2000 mm/year, and decreases towards the Southeastern part of the island, where the amount is between 300 and 700 mm. The mean annual precipitation is 927 mm, which corresponds to 7.69 x 10^9 m^3 (Chartzoulakis and Psarras, 2005). Olive cultivation is most important for the regional economy, 54% of the islands surface represent olive orchards (Chartzoulakis and Psarras, 2005, following Chartzoulakis, 2001). Crete is susceptible to land degradation, caused by climatic variations as well as by land-use activities (Croke et al., 2000).

The area vegetated and so possibly endangered by wildfire covers 3951.51 km^2, accounting for 47.45% of the islands total area. These areas comprise natural grasslands, sclerophyllous vegetation, broad-leaved forest, coniferous forest, mixed forest as well as moors and heathland. Generally, wildfire burnings on Crete occur less frequently as is the case on Sardinia, namely only every second to every third year.

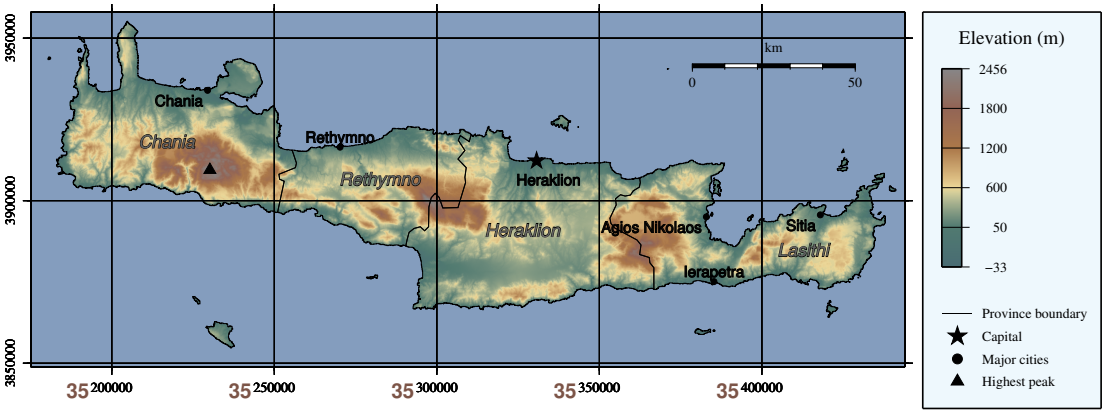


Figure 1.5: Testing Area: Crete

Sources: own mpa, based on:
Elevation: [SRTM41] Jarvis et al., 2008
Rivers/Lakes: [GSHHS] Wessel and Smith, 1996
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012
Cities: [geonames.org] geonames.org, 2012

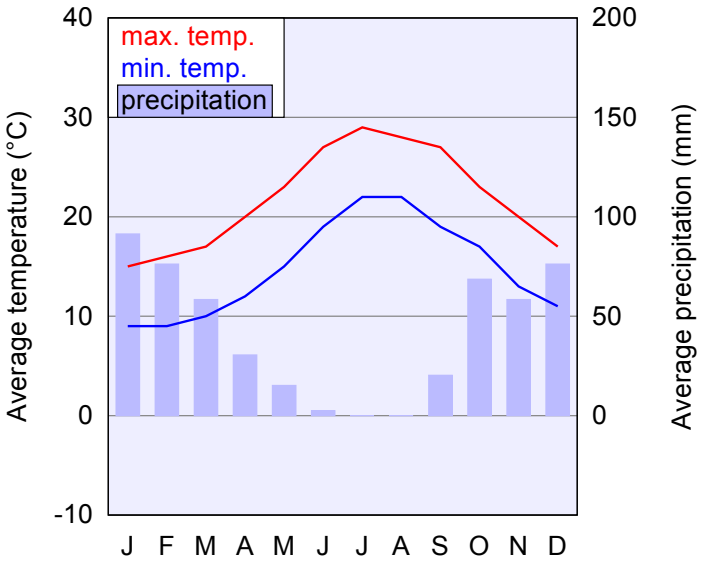


Figure 1.6: Monthly temperature and precipitation averages of Crete
Source: Own figure, data taken from The Weather Channel, LLC weather.com, 2012a

Chapter 2

Status of Current Research

2.1 Wilfire danger rating

Wildland fire research is a domain of considerable complexity. It comprises three main fields of interest, namely the phases before, during, and after fire activity. These phases are addressed by the terms 'fire occurrence', 'fire behaviour', and 'fire effects', respectively. This thesis is concerned with the occurrence of wildland fires, thus mainly dealing with the phase before fire activity. A schematic differentiation of the terms is given in 2.1, cited from Allgöwer, Carlson and Wagtendonk, 2003, with sectors outlined in this thesis highlighted in yellow and orange.

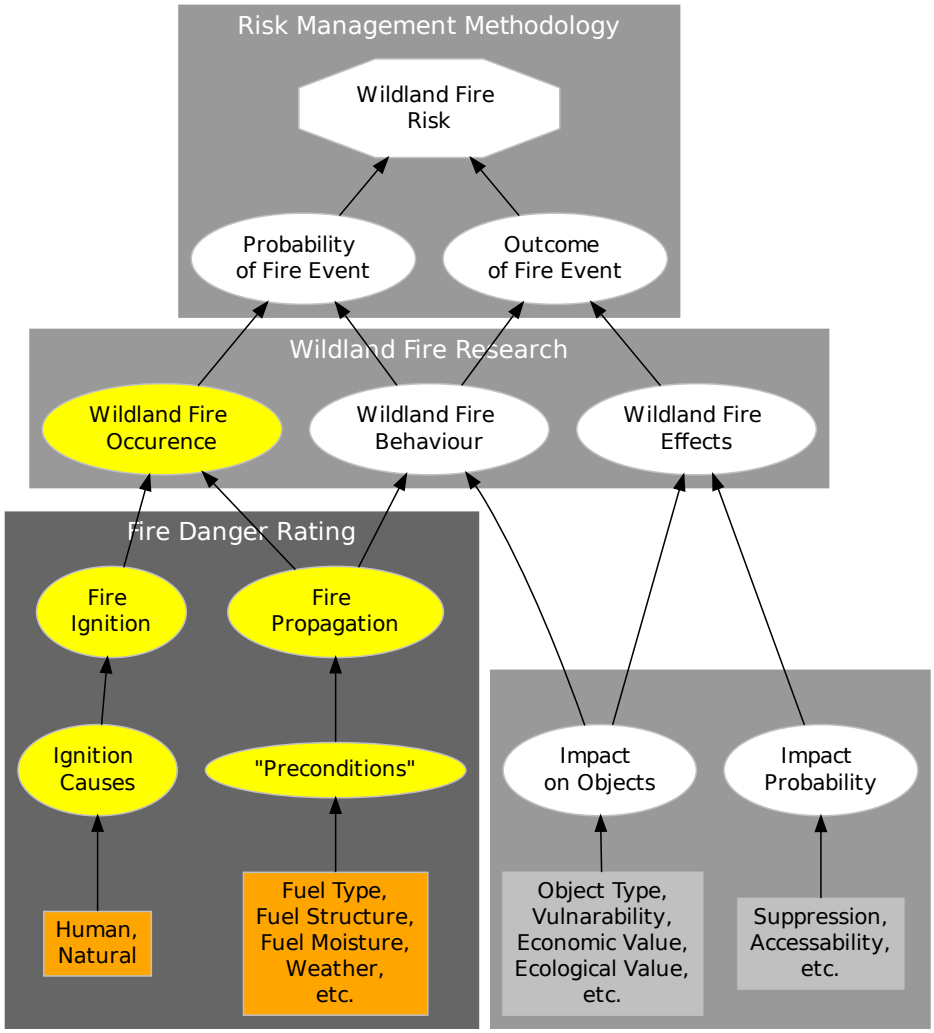


Figure 2.1: Overview of wildfire danger research

Sectors dealt with in this thesis are highlighted. Orange boxes are used for the base factors of fire danger rating: possible ignition sources as well as weather and fuel related input data, respectively.

Source: Allgöwer, Carlson and Wagtendonk, 2003

2.1.1 Background of research

Merrill and Alexander (1987) define forest fire danger as an estimation of fixed and variable factors of the fire environment that affect the ease of ignition, rate of spread, difficulty of control, and fire impact (Stocks et al., 1989).
The rating of fire danger can be described as the evaluation and integration of individual and combined factors influencing fire danger (Stocks et al., 1989). The aim of Fire Danger Rating is the minimisation of the uncertainty of fire danger prediction (Lin et al., 2000). It is essential for fire prevention and fighting preparedness and has been utilised for a long time in countries around the world for early warning purposes. Forest and land management agencies, as well as land owners and communities employ rating methods to determine possible periods of severe fire weather in advance. This allows fire managers to execute prevention, detection and pre-suppression plans before the actual occurrence of a fire (Groot et al., 2006).
The task of determining fire danger potential requires knowledge about biological, climatic, and physical domains, as well as understanding of fire behaviour. The issue has been addressed by using different approaches. Methods based on statistics, biophysics, systematic observation and social surveys as well as computer simulation methods integrated in Geographic Information Systems (GIS) have been used.
GIS is used for sophisticated spatial analysis and modelling. Several methods are available and they are helpful with regard to wildfire management and analysis, such as the calculation of spatial neighbourhood effects or propagation process constraints (Allgöwer, Carlson and Wagtendonk, 2003). Chuvieco integrated GIS and remote sensing techniques into synthetic risk indices. He argued that fire being a spatial and temporal process should be addressed both spatially and temporally, making the use of GIS obvious in this regard (Chuvieco and Riva, 2009). Keane (2001) used GIS and remote sensing techniques to produce fuel maps. Those maps serve as a basis for accessing spatial fire hazard and simulating fire growth and intensity across a landscape (Keane, Burgan and Wagtendonk, 2001). Several further studies regarding fire risk estimation and GIS have been conducted, such as the ones by Chou, 1992, Chuvieco and Congalton, 1989,

Jaiswal et al., 2002, Perry, Sparrow and Owens, 1999, and Schmidt et al., 2002. With the help of the findings of several decades of laboratory experiments and fieldwork, various models indicating fire danger potential have been developed. Those models are mostly based on physical factors. (Lin et al., 2000). The major factors are the ease of ignition of dead fuels and the potential number of ignition sources (Lin et al., 2000, following Deeming et al., 1972).
Fire danger rating systems use certain models to determine the effects of selected fire factors and combine them into one or more qualitative or numerical indices of fire potential (Lin et al., 2000, following Pyne, 1996), which provide helpful information for a variety of fire management activities (Stocks et al., 1989). These indices are distinguished into structural indices, which derived from factors that do not or barely change in time (such as terrain), and dynamic indices, which are based on factors that vary in short periods of time, such as vegetation status or meteorological conditions (Alves et al., 2010).
The index values are often classified into danger classes to simplify interpretation. Usually, four to six rating classes are used: [Very Low,] Low, Moderate, High, [Very High,] and Extreme (Wotton, 2009).

Many fire danger rating systems have been developed worldwide in recent decades (see Viegas et al., 2000, Lin et al., 2000, San-Miguel-Ayanz et al., 2003c). Research and establishment of danger rating methodologies has been carried out in Canada (see Stocks et al., 1989) and the United States (see Cohen and Deeming, 1985) since the 1930s as well as in Australia (by McArthur) since the 1940s (Groot et al., 2006, Lin et al., 2000). All systems in use aim to provide a relatively simple measure of fire danger for the comparison from day to day (Lin et al., 2000). They have proven valuable from regional to global scales (Groot et al., 2006).

2.1.2 Fire danger rating systems

Fire danger rating systems integrate meteorological data with biophysical characteristics of natural fuels, with the aim of forecasting their flammability and

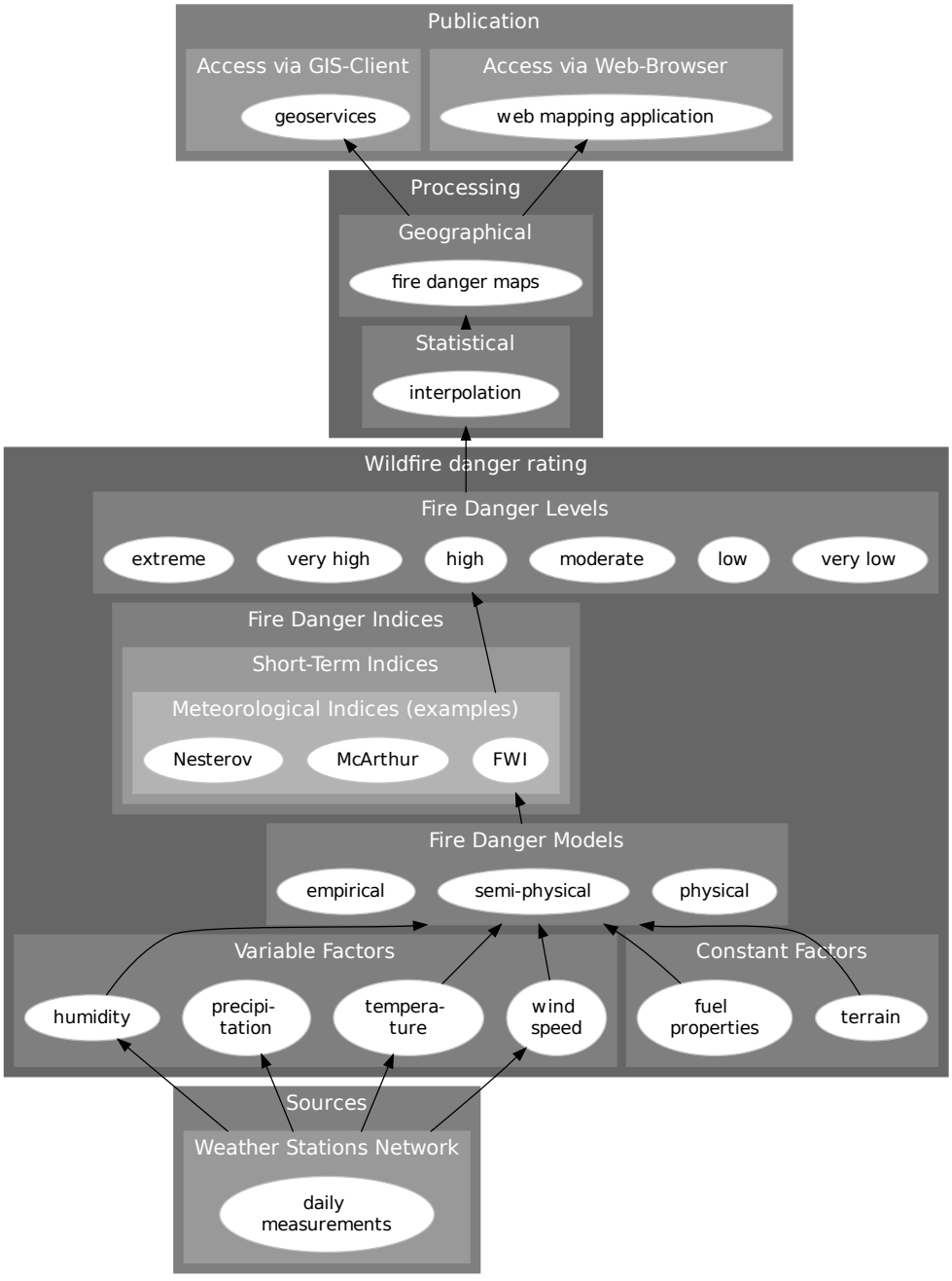


Figure 2.2: Structure of a system for wildfire danger rating and presentation

combustibility, to give an assessment of fire potential for a given area. Weather observations are used as input data (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011). However, forecasts are increasingly supported by other data like satellite imagery, for example high temperature signals and data on fuel conditions (FAO, 2007). By using predicted weather data, up to two weeks of early warning is possible (Groot et al., 2006), however, the forecast reliability decreases significantly with the length of the forecast period. The general structure of a fire danger rating system in visualised in figure 2.2.

Several studies have been conducted to compare danger rating methods regarding different geographic locations, scales, and vegetation types (Lin et al., 2000, Viegas et al., 2000). Existing methods are often reused in or adapted to other geographical, ecological, and cultural situations than they were originally intended for (Groot et al., 2006; Lin et al., 2000). This is often done for economic reasons, because otherwise considerable amounts of time and money had to be invested in research, development, and testing (Fogarty et al., 1998). However, the reusing of existing fire danger system elements is often reasonable due to the following two incidences: the fuel wetting and drying processes are universal, and the responses of fire to variations in fuel, weather and topography are comparable (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011).

In particular, the Canadian Forest Fire Danger Rating System (CFFDRS, see Wagner, 1987) has been adapted or even used in its original form in North America (Alaska, Florida/USA, Mexico) and South America (Venezuela, Chile, Argentina), Europe (Spain, Portugal), Eastern Asia (China, Malaysia, Indonesia, Fiji) and the South Pacific (New Zealand; see Taylor and Alexander, 2006, Fogarty et al., 1998, Taylor, 2001, Lin et al., 2000). The system has further been used for a range of vegetation types, including northern boreal and temperate forests (Taylor and Alexander, 2006), Mediterranean vegetation (Viegas et al., 2000), and tropical Southeast Asian forests (Field, Wang and Roswintiarti, 2004).

The CFFDRS is also used for a global application area in EWS-Fire, which is the forecasting component of the Wildland Fire Early Warning Portal of the Global Fire Monitoring Center (GFMC), operated by Freiburg University, Germany (Groot et al., 2006), and in the European Forest Fire Information System (EFFIS) for the complete European area (San-Miguel-Ayanz et al., 2002).

The complete Canadian Forest Fire Danger Rating System is described in detail in chapter 2.2. Further systems, which have gained international importance are described consecutively.

- ***McArthur Fire Danger Rating System (Australia):*** The system developed by McArthur in 1967 is intended for Australian forests and grasslands. It is based on the rate of spread of surface fires in a standard fuel type. The inputs needed are air temperature, relative humidity, wind, and rain duration. The Keith/Byram Drought Index (described below) has been implemented in the system. It was originally implemented via simple fire danger meters, which were converted to mathematical equations by Noble et al. in 1980 (Lin et al., 2000). As it is the case for the Canadian system, the McArthur models derived from statistical analyses of large amounts of field data. The McArthur index is interpreted as an estimation of the rate of fire spread under current climatic conditions. Spain has adopted this system.

- ***National Fire Danger Rating System (NFDRS):*** The NFDRS, developed by Deeming in 1972 is a hierarchical high level system using several mathematical models, the most important one being the fire spread model developed by Rothermel in 1972. The required inputs for the NFDRS are essentially weather data (air temperature, relative humidity, cloudiness, and fuel moisture), but it is also capable of taking human-caused fire risk into account. The output is the prediction of fire potential. The system has been revised several times since its original creation (Cohen and Deeming, 1985; Burgan, 1988). The main indices produced are the Spread Component (SC), the Energy Release Component (ERC), and the Ignition Component (IC). Fuel conditions and slope are considered to be constant over the area of interest. The System has been adopted by South Africa (Lin et al., 2000).

- ***Nesterov Empirical Drought Index:*** The Russian system, developed by Nesterov in 1949, is an empirical drought index used for estimations of fine fuel flammability. It is based on air temperature, relative humidity and daily precipitation (Mantzavelas, 2009). It was created as an empirical

function based on historical weather data (Mantzavelas, 2009, following Venevsky, 2002). Besides Russia, the Nesterov index is used in Austria and Portugal.

- ***Angstrøm Index:*** The Swedish index is a simple drought index. Only temperature and relative humidity are considered. The output is the probable number of ignitions on a given day (Mantzavelas, 2009). The index is also used in Norway and Denmark (Lin et al., 2000).

- ***Keetch-Byram Drought Index (KBDI):*** The KBDI, developed by Keetch and Byram in 1968, is an index for moisture deficiency, widely used by fire control managers for wildfire monitoring and prediction worldwide (Mantzavelas, 2009). It can be related to fire potential via a lookup table. Its required inputs are maximum daily temperature, total daily precipitation and the average annual precipitation. The index was thoroughly tested over northern Eurasia with good results for fire danger early warning (Groisman et al., 2007). In addition to being a stand-alone index, it was also incorporated in the National Fire Danger Rating System of the US since 1988 (Burgan, 1988), the Australian McArthur Fire Danger Index and the Canadian Fire Danger Rating System (San-Miguel-Ayanz et al., 2003b).

Table 2.1 is a compilation of indices produced by rating systems frequently cited in literature. However, several other countries have implemented these systems, with minor or no adaptation. All listed methods only require meteorological data as input, with the exception of the Ignition Component (IC) of the United States NFDRS system, which additionally processes information on possible human-caused fire danger (Lin et al., 2000).
The establishment of advanced communication technology in the 1980s and 1990s as well as the development of autonomous weather stations now allows for collection of weather data even in remote locations in near realtime (Taylor, 2001). Subsequently, automated fire danger rating systems are increasingly being used worldwide to forecast periods of potentially serious fire weather. Sophisticated fire management systems such as the Spatial Fire Management System (sFMS) by Englefield et al. (2000) have evolved to process and display these data in

concert with electronic fuels and topographic data using GIS engines (Englefield, Lee and Suddaby, 2000). A Fire Danger Rating System (FDRS) for Southeast Asia is in active use since 2000, Vietnam, Russia, Canada, South-Africa, India and Australia also use automated fire danger rating systems (Groot et al., 2006). Modern implementations are often realised in connection with a web server environment, so that users can access forecasting information through the internet. A web frontend often implements GIS-derived, high-resolution danger maps. (Groot et al., 2006, following Lee et al., 2002; Lin et al., 2000, following Burgan, 1998).

Fire danger rating systems have to be differentiated from Fire Behaviour Systems. While the first ones produce indices to indicate possible fire danger in the near future, the latter deal with the propagation of one or several fires currently burning. Possible model outputs include fire spread rate, circumference, flame length, and scorch height (Missoula Fire Sciences Laboratory, 2013). Fire Behaviour Systems are not dealt with in this thesis.
The following systems are in use today (Missoula Fire Sciences Laboratory, 2013):

- **FARSITE**, a fire area simulator by Finney, 1994
- **BehavePlus**, a fire modelling by Andrews, 2003
- **FlamMap**, a system for fire growth simulation by Finney, 2006

Table 2.1: Fire related indices used in different countries

(indices having emerged from others are listed in square brackets below them)

Index	Country	Author
Angstrøm Index	Sweden	–
Baumgartner Index	Germany	Baumgartner, 1967
Carrega Index	France	Carraga, 1985
CFFDRS: Build Up Index (BUI)	Canada	Van Wagner, 1970
CFFDRS: Drought Code (DC)	Canada	Van Wagner, 1970
CFFDRS: Duff Moisture Content (DC)	Canada	Van Wagner, 1970
CFFDRS: Fine Fuel Moist. (FFMC)	Canada	Van Wagner, 1970
[BEHAVE Dead Fine Fuel Moist.]	US	Andrews, 1986
[ICONA Method]	Spain	ICONA, 1993
[BehavePlus Dead Fine Fuel Moist.]	US	Andrews, 2003
CFFDRS: Fire Weather Index (FWI)	Canada	Van Wagner, 1970
[Met Office Fire Severity Index]	UK	Kitchen, 2006
CFFDRS: Initial Spread Index (ISI)	Canada	Van Wagner, 1970
Drouet-Sol Numerical Risk Index	France	Drouet/Sol, 1990
Fosberg Fire Weather Index (FFWI)	US	Fosberg, 1978
Fuel Moisture Index (FMI)	Australia	Sharples, 2009
IREPI Index	Italy	Bovio, 1994
Keetch-Byram Drought Index (KBDI)	US	Keetch/Byram, 1968
McArthur Fire Danger Index (FDI)	Australia	McArthur, 1967
[Italian Fire Danger Index]	Italy	Palmieri, 1983
Monte Alegre Formula (FMA)	Brazil	Soarez, 1972
[Mod. Monte Alegre Formula (FMA+)]	Brazil	Nunez, 2005
Nesterov Emp. Drought Index	Russia	Nesterov, 1967
[Mod. Nesterov Emp. Drought Index]	Russia	Nesterov, 1967
[Zhdanko Index]	Russia	Zhdanko, 1965
[Portuguese Index]	Portugal	Lourenço, 1988
NFDRS: Spread Component (SC)	US	Deeming, 1972
NFDRS: Energy Release Comp. (ERC)	US	Deeming, 1972
NFDRS: Ignition Component (IC)	US	Deeming, 1972
Orieux Index	France	Carrega, 1991
WBKZ - M68 Forest Danger Index	Germany	Käse, 1969

Sources:
Lin et al., 2000, Camia and Bovio, 2000, Mantzavelas, 2009

2.2 The Canadian Forest Fire Weather Index System

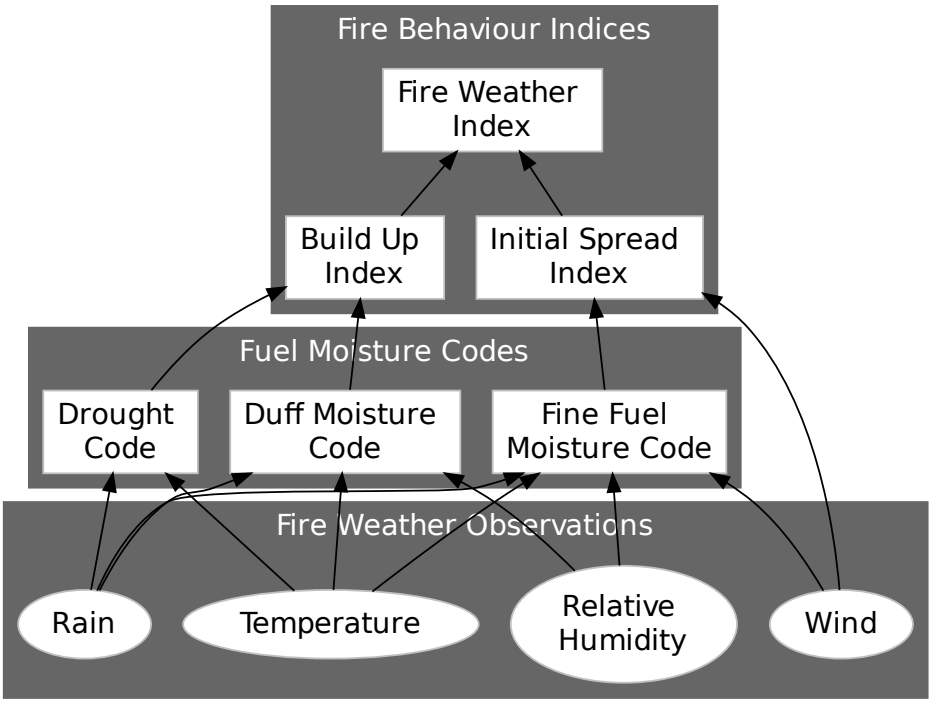


Figure 2.3: FWI structure
following Wagner, 1987

Forest fire danger rating research in Canada dates back to 1925, initiated by the federal government. J.G Wright, at that time, developed a practical research program to examine possible correlations between weather, fuel moisture and fire behaviour. Over the next decades, four different fire danger rating systems have been developed by Wright, H.W. Beall and others (Stocks et al., 1989) exhibiting increasing national applicability. An evolutionary approach was used in which components were maintained from system to system, sometimes with modifications (Wagner, 1987). Through the decades of development, the aim was

to simplify the required weather measurements as well as the method of calculation (Stocks et al., 1989). All approaches have been based on field experiments and empirical analysis (San-Miguel-Ayanz et al., 2003a), including test fire behaviour (Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011). The currently used system is called Canadian Forest Fire Danger Rating System (CFFDRS) and has been under development by Forestry Canada since 1968 (Stocks et al., 1989). The CFFDRS consists of two major subsystems, the Canadian Forest Fire Weather Index System (FWI) and the Canadian Forest Fire Behaviour Prediction System (FBP), as well as two minor subsystems, the Accessory Fuel Moisture System (AFM) and the Canadian Forest Fire Occurrence Prediction System (FOP). While AFM and FOP, the two smaller ones have not yet been developed for national use, the FBP was completed in 1992, and the FWI has been in active use across Canada since 1970, with minor revisions undertaken in 1976, 1978 and 1984 (San-Miguel-Ayanz et al., 2003a).

The development and structure of the FWI subsystem were originally described by Van Wagner in 1974. In 1985, Van Wagner and Pickett presented a FORTRAN 77 source code for calculation of the FWI, replacing the data tables, which have been used until then (Wagner and Pickett, 1985).

The subsystem itself comprises six components, each producing a relative numerical rating for a specific aspect of fire danger. These subindices can be grouped in fuel moisture codes and fire behaviour indices. The first group consists of the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), and the Drought Code (DC). The second contains the Initial Spread Index (ISI), the Build Up Index, and finally the Fire Weather Index (FWI) itself.

Collectively, they account for the effects of fuel moisture and wind on fire behaviour (Stocks et al., 1989). The calculation is based on four weather observations: Dry-bulb temperature, relative humidity, open wind speed, and accumulated rainfall of 24 hours, taken daily at noon (13:00h local time during wintertime, 12:00h local time in summer). The FWI system depends solely on weather data and does not consider differences in ignition risk or topography, nor does it notice differences in fuel types (San-Miguel-Ayanz et al., 2003a). The system was, however, developed to represent fire danger in a generalised, standard fuel type for vegetation typical across Canada, such as jack pine stands.

The FWI with its subsystems is a valuable indicator for different characteristics of fire activity. The main index itself is best used as a measuring indicator for fire danger in general. However, it should be examined together with the subcomponents for proper fire information (Stocks et al., 1989).
The FWI system can further be divided into fuel moisture codes as well as fire behaviour indices. While the three moisture codes, FFMC, DMC, and DC, represent actual measures for water content in fine surface litter, loosely compacted duff, and deep, organic matter, respectively, the remaining three indices address danger conditions directly related to fire activity. ISI gives information on the possible speed of the spreading of a fire. BUI addresses the amount of fuel that could possibly be combusted in case of a fire, and the FWI is the computation of the theoretical fire intensity of the flame front, based on the other five measures. As stated, it is also interpretable as fire danger in general.
The outputs of the FWI system have been tested against fire report information over many years across Canada. The results showed strong correlations between fire activity and fire weather severity as reflected by the FWI (Stocks et al., 1989). Strong relationships also existed between man caused fire occurrence and the FFMC, as well as between the burned area and the ISI (Stocks et al., 1989, following Turner, 1973; Stocks, 1974; and Kiil et al., 1977).
Within this thesis, the FWI system has been determined as the method of choice for fire danger determination. The computation of forecasts regarding the ten-year study as well as the daily forecasting on the platform is based on the FWI and its subindices ISI, BUI, FFMC, DMC, and DC. The following sections describe the indices in detail. It is noteworthy that the indices are not applicable if certain conditions are not met. In these cases, the calculations may fail or yield in erroneous results. These limiting conditions, such as the inapplicability of the FFMC calculations in dry weather conditions, are listed in the section called 'Restrictions' right after the formulas for each index.
The structure of the complete FWI system is shown in illustration 2.3a.

Elements of fire weather used as input values for the FWI subindices:

- $\boldsymbol{T}$**:** noon temperature (°C)
- $\boldsymbol{H}$**:** noon relative humidity (%)
- $\boldsymbol{W}$**:** noon wind speed (km/h)
- $\boldsymbol{r_o}$**:** daily rainfall, measured at noon (mm)
- $\boldsymbol{r_f}$**:** effective rainfall (mm), FFMC
- $\boldsymbol{r_e}$**:** effective rainfall (mm), DMC
- $\boldsymbol{r_d}$**:** effective rainfall (mm), DC

Effective rainfall addresses the amount of total rainfall less the actual evapotranspiration.

Table 2.2 shows input and output values for the FWI and its subindices, calculated with the formulas given in the following subsections. The timespan shown is July 2012, when the fire activity of the 2012 fire season was at its peak. The resulting values for the FWI as well as for the subindices subsequently have to be classified via a classification scheme to achieve interpretable results in respect of fire danger. These computations form the basis for the fire danger forecasting performed in this thesis.

Table 2.2: FWI input and output values

Weather station: ISARDEGN6 in Genneruxi, Sardinia in July 2012

day	T	H	W	r_o	FFMC	DMC	DC	ISI	BUI	FWI
09	30.7	59.4	15.5	0.0	89.4	182.7	599.7	8.5	207.4	35.2
10	32.6	45.8	23.4	0.0	90.2	187.0	609.3	14.3	211.6	49.1
11	32.7	45.5	13.9	0.4	90.3	191.3	618.9	9.0	215.8	36.6
12	33.9	35.1	19.8	0.3	92.4	196.6	628.7	16.3	220.7	53.5
13	31.0	39.9	24.0	0.0	92.4	201.2	638.0	20.4	225.0	61.5
14	31.9	36.6	23.0	0.0	92.5	206.1	647.4	19.5	229.5	59.9
15	34.5	46.4	10.1	0.0	92.1	210.6	657.3	9.6	233.9	38.4
16	30.9	58.1	18.8	0.0	89.6	213.7	666.6	10.4	237.3	40.5
17	27.4	86.6	19.7	0.0	83.0	214.6	675.3	4.4	239.2	22.5
18	29.3	77.2	18.0	0.0	83.7	216.2	684.2	4.4	241.6	22.6
19	32.3	66.9	9.4	0.0	86.0	218.8	693.8	3.9	244.7	20.8
20	31.4	79.8	10.8	0.0	85.2	220.4	703.1	3.7	247.1	20.2
21	32.9	56.2	10.4	0.0	88.0	223.9	712.7	5.5	250.8	26.6
22	26.2	61.2	27.3	1.5	83.6	226.4	721.2	6.9	253.7	31.0

2.2.1 Fine Fuel Moisture Code (FFMC)

The FFMC represents the water content of fine surface litter (Stocks et al., 1989). It is an indicator of the relative ease of ignition and flammability of these fuels (Camia and Bovio, 2000). Similar to the DMC and DC, the FFMC is a book-keeping system, meaning that the moisture value of the current day is dependent on the value of the previous day and the present weather. The system adds a moisture value after a rainfall and subtracts a certain amount for each day's drying (Stocks et al., 1989).

Symbols used in equations:

- m_o: fine fuel moisture content from previous day (%)
- m_r: fine fuel moisture content after rain (%)

- $\boldsymbol{m}$: water content in fine fuel moisture content after drying (%)
- $\boldsymbol{E_d}$: fine fuel EMC (Equilibrium Moisture Content) for drying (%)
- $\boldsymbol{E_w}$: fine fuel EMC (Equilibrium Moisture Content) for wetting (%)
- $\boldsymbol{k_o}$: intermediate step in calculation of k_d
- $\boldsymbol{k_d}$: logarithmic drying rate, FFMC ($\log_{10}$m/day)
- $\boldsymbol{k_l}$: intermediate step in calculation of of k_w
- $\boldsymbol{k_w}$: logarithmic wetting rate ($\log_{10}$m/day)
- $\boldsymbol{F_o}$: previous day's FFMC (-)
- $\boldsymbol{F}$: FFMC (-)

Equations (following Wagner and Pickett, 1985):

$$m_o = 147.2\frac{101 - F_o}{59.5 + F_o} \tag{2.1}$$

$$\text{if } r_o > 0.5 \text{ than } r_f = r_o - 0.5 \tag{2.2}$$

$$\text{if } m_o \leq 150 \text{ than } m_r = m_o + 42.5 r_f (e^{\frac{-100}{251 - m_o}})(1 - e^{\frac{-6.93}{r_f}}) \tag{2.3}$$

$$\begin{aligned}\text{if } m_o > 150 \text{ than } m_r = m_o + 42.5 r_f (e^{\frac{-100}{251 - m_o}})(1 - e^{\frac{-6.93}{r_f}}) \\ +0.0015(m_o - 150)^2 {r_f}^{0.5}\end{aligned} \tag{2.4}$$

$$E_d = 0.942 H^{0.679} + 11 e^{\frac{H-100}{10}} + 0.18(21.1 - T)(1 - e^{-0.115H}) \tag{2.5}$$

$$E_{\mathrm{w}} = 0.618H^{0.753} + 10e\frac{H-100}{10} + 0.18(21.1-T)(1-e^{\text{-}0.115H}) \tag{2.6}$$

$$k_{\mathrm{o}} = 0.424(1-(\frac{H}{100})^{1.7}) + 0.0694W^{0.5}(1-(\frac{H}{100})^{8}) \tag{2.7}$$

$$k_{\mathrm{d}} = k_{\mathrm{o}}\ 0.581e^{0.0365\,T} \tag{2.8}$$

$$k_{\mathrm{l}} = 0.424(1-(\frac{100-H}{100})^{1.7}) + 0.0694W^{0.5}(1-(\frac{100-H}{100})^{8}) \tag{2.9}$$

$$k_{\mathrm{w}} = k_{\mathrm{l}}\ 0.581e^{0.0365\,T} \tag{2.10}$$

$$m = E_{\mathrm{d}} + (m_{\mathrm{o}} - E_{\mathrm{d}})10^{\text{-}\mathrm{k_d}} \tag{2.11}$$

$$m = E_{\mathrm{w}} - (E_{\mathrm{w}} - m_{\mathrm{o}})\ 10^{\text{-}\mathrm{k_w}} \tag{2.12}$$

$$F = 59.5\frac{250-m}{147.2+m} \tag{2.13}$$

Calculation:

1. F_o is F of the previous day.
2. m_o is calculated from F_o by equation 2.1.
3. (a) if $r_o > 0.5$, r_f is calculated by equation 2.2.
   (b) m_r is calculated from r_f and m_o by equation 2.3 or 2.4.
      i. if $m_o \leq 150$, equation 2.3 is used.
      ii. if $m_o > 150$, equation 2.4 is used.
   (c) m_r takes the value of m_o.
4. E_d is calculated by equation 2.5.
5. (a) if $m_o > E_d$, k_d is calculated by equations 2.7 and 2.8.
   (b) m is calculated by equation 2.11.
6. if $m_o < E_d$, E_w is calculated by equation 2.6.
7. (a) if $m_o < E_w$, k_w is calculated by equations 2.9 and 2.10.
   (b) m is calculated by equation 2.12.
8. if $E_d \leq m_o \leq E_w$, $m = m_o$.
9. F is calculated from m by equation 2.13. This is the FFMC of the current day.

Restrictions:

1. Equations 2.3 and 2.4, respectively, must not be used when $r_o \leq 0.5$ (mm rainfall), meaning that in dry weather the rainfall routine cannot be performed.
2. m has an upper limit of 250. When equations 2.3 or 2.4 result in $m_r > 250$, m_r is to be set to 250.

2.2.2 Duff Moisture Code (DMC)

The DMC represents the amount of water contained in loosely compacted duff of moderate depth (Stocks et al., 1989). It is also used as an estimate of the amount of combustible fuel of medium size (Camia and Bovio, 2000). Similar to the other two fuel moisture codes, it is a bookkeeping system, in which the calculation for one day's value depends on the value of the day before.

Symbols used in equations:

- $\boldsymbol{M_o}$**:** duff moisture content from previous day (%)
- $\boldsymbol{M_r}$**:** duff moisture content after rain (%)
- $\boldsymbol{M}$**:** duff moisture content after drying (%)
- $\boldsymbol{K}$**:** logarithmic drying rate, DMC ($\log_{10}$m/day)
- $\boldsymbol{L_e}$**:** effective day length in DMC (hours)
- $\boldsymbol{b}$**:** slope variable in DMC rain effect (-)
- $\boldsymbol{P_o}$**:** previous day's DMC (-)
- $\boldsymbol{P_r}$**:** DMC after rain (-)
- $\boldsymbol{P}$**:** DMC (-)

Equations (following Wagner and Pickett, 1985):

$$\text{if } r_o > 1.5 \text{ than } r_e = 0.92 r_o - 1.27 \tag{2.14}$$

$$M_o = 20 + e^{\frac{5.6348 - P_o}{43.43}} \tag{2.15}$$

$$\text{if } P_{\text{o}} \leq 33 \text{ than } b = \frac{100}{0.5 + 0.3P_{\text{o}}} \tag{2.16}$$

$$\text{if } 33 < P_{\text{o}} \leq 65 \text{ than } b = 14 - 1.3lnP_{\text{o}} \tag{2.17}$$

$$\text{if } P_{\text{o}} > 65 \text{ than } b = 6.2 \text{ ln } P_{\text{o}} - 17.2 \tag{2.18}$$

$$M_{\text{r}} = M_0 + \frac{1000r_{\text{e}}}{48.77 + br_{\text{e}}} \tag{2.19}$$

$$P_{\text{r}} = 244.72 - 43.43ln(M_{\text{r}} - 20) \tag{2.20}$$

$$K = 1.894(T + 1.1)(100 - H)\ L_{\text{e}}\ 10^{-6} \tag{2.21}$$

$$P = P_{\text{o}} \text{ (or } P_{\text{r}}) \ + 100K \tag{2.22}$$

Table 2.3: Effective day lengths L_{e} for DMC

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
L_{e}	6.5	7.5	9.0	12.8	13.9	13.9	12.4	10.9	9.4	8.0	7.0	6.0

Calculation:

1. P_o is P of the previous day.
2. (a) if $r_o > 1.5$, r_e is calculated by equation 2.14.
   (b) M_o is calculated from P_o by equation 2.15.
   (c) b is calculated by the appropriate one of equations 2.16, 2.17, or 2.18.
   (d) M_r is calculated by equations 2.19, 2.17, or 2.18.
   (e) M_r is converted to P_r by equation 2.20. P_r takes the value of P_o.
3. L_e is taken from table 2.3.
4. K is calculated by equation 2.21.
5. P is calculated from P_o by equation 2.22. This is the DMC of the current day.

Restrictions:

1. Equations 2.14 to 2.20 cannot be used unless $r_o > 1.5$, meaning that in dry weather the rainfall routine cannot be performed.
2. P_r cannot be less than zero. Negative values resulting from step 2e must be set to zero.
3. Values of T (noon temperature) less than -1.1 must not be used in equation 2.22. If $T <$ -1.1, T has to be set to -1.1.

2.2.3 Drought Code (DC)

The DC is an estimation of the water content in deep, compact organic matter (Stocks et al., 1989). It also indicates seasonal drought effects on large size fuels

(Camia and Bovio, 2000). Similar to FFMC and DMC, the DC is a bookkeeping system.

Symbols used in equations:

- $\boldsymbol{Q}$: moisture equivalent of DC (%)
- $\boldsymbol{Q_o}$: moisture equivalent of previous day's DC (%)
- $\boldsymbol{Q_r}$: moisture equivalent after rain (%)
- $\boldsymbol{V}$: potential evapotranspiration (mm, units of 0.254 mm water/day)
- $\boldsymbol{L_f}$: effective day length in DC (hours)
- $\boldsymbol{D_o}$: previous day's DC (-)
- $\boldsymbol{D_r}$: DC after rain (-)
- $\boldsymbol{D}$: DC

Equations (following Wagner and Pickett, 1985):

$$\text{if } r_o > 2.8 \text{ than } r_d = 0.83 r_o - 1.27 \tag{2.23}$$

$$Q_o = 800 e^{\frac{-D_o}{400}} \tag{2.24}$$

$$Q_r = Q_o + 3.937 r_d \tag{2.25}$$

$$D_r = 400 ln \frac{800}{Q_r} \tag{2.26}$$

$$V = 0.36(T + 2.8) + L_{\mathrm{f}} \tag{2.27}$$

$$D = D_{\mathrm{o}} + 0.5V \tag{2.28}$$

Table 2.4: Day length factors (L_{f}) for DC

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
L_{f}	-1.6	-1.6	-1.6	0.9	3.8	5.8	6.4	5.0	2.4	0.4	-1.6	-1.6

Calculation:

1. D_{o} is D of the previous day.
2. (a) if r_{o} ¿ 2.8, r_{d} is calculated by equation 2.23.
   (b) Q_{o} is calculated from D_{o} by equation 2.24.
   (c) Q_{r} is calculated by equation 2.25.
   (d) Q_{r} is converted to D_{r} by equation 2.26. D_{r} takes the value of D_{o}.
3. L_{f} is taken from table 2.3.
4. V is calculated by equation 2.27
5. D is calculated from D_0 by equation 2.28. This is the DC of the current day.

Restrictions:

1. Equations 2.23 to 2.26 cannot be used unless $r_o > 2.8$, meaning that in dry weather the rainfall routine cannot be performed.
2. D_r cannot be less than zero. Negative values resulting from step 2d must be set to zero.
3. Values of T less than -2.8 are not allowed in equation 2.27. If $T <$ -2.8, T has to be set to -2.8.
4. V cannot be negative. If the result of equation 2.27 is negative, V must be set to 0.

2.2.4 Initial Spread Index (ISI)

The ISI provides a rating of fire spread, meaning the expected speed of propagation of the flame front. It is generated through the combination of FFMC and wind speed. The variability of fuels is not considered (Stocks et al., 1989, Camia and Bovio, 2000).

Symbols used in equations:

- $\boldsymbol{f(W)}$**:** wind factor
- $\boldsymbol{f(F)}$**:** fine fuel humidity factor
- $\boldsymbol{m}$**:** water content in fine fuel moisture content after drying
- $\boldsymbol{R}$**:** ISI

Equations (following Wagner and Pickett, 1985):

$$f(W) = e^{0.05039\,W} \tag{2.29}$$

$$f(F) = 91.9e^{-0.1386m}(1 + \frac{m^{5.31}}{4.93\ 10^7}) \tag{2.30}$$

$$R = 0.208f(W)f(F) \tag{2.31}$$

Calculation:

1. *f(W)* is calculated by equation 2.29.
2. *f(F)* is calculated by equation 2.30.
3. R is calculated by equation 2.31. This is the ISI of the current day.

2.2.5 Build Up Index (BUI)

The BUI, which is the combination of DMC and DC, is a numerical rating of the total amount of combustible fuel (Stocks et al., 1989).

Symbols used in equations:

- ***P*:** DMC
- ***D*:** DC
- ***U*:** BUI

Equations (following Wagner and Pickett, 1985):

$$\text{if } P \leq 0.4D \text{ than } U = \frac{0.8PD}{P + 0.4D} \tag{2.32}$$

$$\text{if } P > 0.4D \text{ than } U = P - (1 - \frac{0.8D}{P + 0.4D})(0.92 + (0.0114P)^{1.7}) \tag{2.33}$$

Calculation:

1. U is calculated by equation 2.32 if $P \leq 0.4D$, or by equation 2.33 if $P > 0.4D$. This is the BUI of the current day.

2.2.6 Fire Weather Index (FWI)

The FWI, generated through the combination of ISI and BUI, is a measure of the potential intensity of a propagating flame front in a standard fuel type on even terrain. (Stocks et al., 1989, Camia and Bovio, 2000).

Symbols used in equations:

- $\boldsymbol{f}$**(D):** duff moisture function
- $\boldsymbol{R}$**:** ISI
- $\boldsymbol{U}$**:** BUI
- $\boldsymbol{B}$**:** FWI (intermediate form)
- $\boldsymbol{S}$**:** FWI (final form)

Equations (following Wagner and Pickett, 1985):

$$\text{if } U \leq 80 \text{ than } f(D) = 0.626U^{0.809} + 2 \tag{2.34}$$

$$\text{if } U > 80 \text{ than } f(D) = \frac{1000}{(25 + 108.64e^{-0.023\mathrm{U}})} \tag{2.35}$$

$$B = 0.1R\ f(D) \tag{2.36}$$

$$\text{if } B > 1 \text{ than } lnS = 2.72(0.434\ lnB)^{0.647} \tag{2.37}$$

$$\text{if } B \leq 1 \text{ than } S = B \tag{2.38}$$

Calculation:

1. *f(D)* is calculated by equation 2.34 for values of $U \leq 80$. If $U > 80$, 2.35 is used.
2. B is calculated by equation 2.36.
3. if $B > 1$, S is calculated by its logarithm, given by equation 2.37. If $B \leq 1$, S has to be set to B according to equation 2.38. S is the FWI of the current day.

2.3 The European Forest Fire Information System (EFFIS)

2.3.1 Background

EFFIS is an information system purposed for European forest fire danger rating, developed and managed by a research group working specifically on the design and realisation of methods for the evaluation of forest fire danger. This group was originally set up by the Joint Research Centre (JRC) of the European Commission (EC) in 1998, with the coordination taken over by Directorate General for Environment of the European Commission (DG ENV; San-Miguel-Ayanz et al., 2002). The system could be commissioned in the year 2000. While the research work and the operation of the EFFIS web platform is done at JRC, the assistance of the European member states regarding in-depth background information is crucial (San-Miguel-Ayanz et al., 2003c). Meetings of the network of forest fire experts from the participating 24 European Union (EU) countries are held regularly twice a year (JRC, 2013a). These countries are Bulgaria, Croatia, Cyprus, Czech Republic, Estonia, Finland, France, Former Yugoslavian Republic of Macedonia, Germany, Greece, Hungary, Italy, Latvia, Lithuania, Montenegro, Poland, Portugal, Romania, Slovakia, Spain, Sweden, Switzerland, Turkey, and the United Kingdom. The network's activities are coordinated by the unit for Agriculture, Forest & Soils of the DG ENV and the unit for Land Management and Natural Hazards of JRC (JRC, 2013f). The network surrounding EFFIS is open to be joined by every state within the EU, as well as all non-European Mediterranean Countries (JRC, 2013c).

The goal of EFFIS is to support the services in charge of the protection of forests against fires in the EU countries (JRC, 2013e) and to provide homogeneous information on forest fires at the European level (San-Miguel-Ayanz et al., 2002). The preparation of up-to-date information is possible by using remote sensing data and GIS processing (San-Miguel-Ayanz et al., 2003c).
The prepared products comprise daily meteorological fire danger maps and forecasts up to six days, daily satellite images as well as maps of the latest hot spots

(JRC, 2013e). The EFFIS platform is used by the European member states to exchange information about fire prevention, fire fighting, restoration practices and other activities related to fire management (JRC, 2012). It is important to note, however, that the institutions involved take no responsibility for any damages or losses occurring through fire prevention activities undertaken or not undertaken because of the forecasts made by the system (JRC, 2013d). Besides the online system, the scientists concerned with EFFIS also maintain a European Union fire database, which contains the forest fire information compiled by the member states. The members are bound to deliver a minimum set of data for each fire occurring (JRC, 2013g).

EFFIS is developed to support EC initiatives and regulations. Since 2003, it is part of the Regulation (EC) No 2152/2003 (Forest Focus) of the European Council and Parliament on monitoring forests and environmental interactions. This regulation succeeded the former directives on forest fire prevention, EEC No 2158/1992 and EEC No 804/94, in which the initialisation of harmonised fire information in the EU member states was appointed, as well as the setting up of a common core database on forest fires. In improving cooperation in forest fire preparedness and fighting, EFFIS supports the Community Action Plan on Civil Protection of 2002, and the Civil Protection Action Plan. Regarding the regional development on areas affected by hazards, EFFIS is in concordance with the European Spatial Development Perspective (ESDP) / ESPON (JRC, 2013i).

2.3.2 Technical details

EFFIS uses weather forecast data to produce daily maps of fire danger for the next six days to come during the main fire season, from beginning of March until the end of October. Modelling results of the COSMO-EU and GME weather forecasting models are used as input data, which is gathered from German meteorological service (Deutscher Wetterdienst, DWD). The model results are transferred to geographical maps with a spatial resolution of 36 km. Several fire danger rating methods have been tested during the first five years of operating time of

EFFIS, and in 2007, the Canadian Fire Weather Index (FWI) was finally adopted as the best suiting method for identifying fire danger in the European Countries (San-Miguel-Ayanz et al., 2002). The indices tested are described in detail in Camia and Bovio, 2000. The fire danger rating is realised by using six fire danger classes (expanded from five classes in summer 2012), ranging from 'Very low' to 'Extreme'. These classes are generated in the same way for all subregions to gain a homogeneous picture of fire danger throughout Europe (JRC, 2013h). The forecasting maps are distributed via a web-based interface and additionally sent by email to the services in charge of fire protection in the specific countries (San-Miguel-Ayanz et al., 2003c).

The web interface also displays active fire hotspot data gained from the Moderate-Resolution Imaging Spectroradiometer (MODIS) Aqua / Terra satellites (the dataset is described in detail in section 3.5.1, as well as the MODIS sensor), and can inform the user about the main land cover category affected, as well as transfer news to the user regarding specific fire by analysing several Really Simple Syndication (RSS) feeds (JRC, 2013b). EFFIS may also be used for rapid damage assessment by the member states. For this purpose, MODIS daily true colour images with 250 m resolution as well as MODIS daily burned area images can be displayed via the web interface (JRC, 2013j).

Chapter 3

Conceptual Design

This chapter describes the general approach of realising a fire danger rating system, as well as the applications and datasets relevant for this thesis. An overview of applications is given in table 3.1 (p. 68), whereas table 3.2 (p. 73) shows the data used.
In the frame of this thesis, only free and Open Source software applications and libraries have been used, as well as only freely accessible data. Since the aim of this work is a self updating fire danger forecasting platform, which requires a maximum level of automation, only command line applications and programming scripts have been utilised to build the processing chain for the prediction of fire danger areas, as well as for the preparation of source data, and the validation process. This methodological approach has also been maintained throughout the complete preparation of this thesis. All maps, figures, and schematic diagrams have been created using scripts. The specific scripts created for controlling the fire danger platform can be found in the appendix.

3.1 Selection and adaptation of the FWI as fire danger rating system

According to Taylor, 2001, the engineering of a new fire danger rating system is a costly and long-term process, which mainly consists of three parts:

- An investigation of relationships between weather, fuels and topography regarding fire occurrence has to be conducted by means of long-term studies and field experiments. What additionally needs to be developed are moisture models representing fuels that are important to fire behaviour, regarding relationships with ignition, spread, difficulty of control and impact.
- The technical infrastructure to gather data related to fire occurrence has to be set up, as well as the equipment to process and analyse the data and publish the results as soon as possible.
- One should build and maintain relationships with agencies in charge, especially in terms of information transfer and training.

Since creating a completely new fire danger index was beyond the scope of this thesis, an appropriate index that already existed was selected and adapted to the region on interest.
One advantage of adopting an existing Fire Danger Rating System is that it minimises the costs that are usually needed for research and development. Yet, on the other hand, the application of this system can cause significant errors if the given fire environment distinctly differs from the one the system was originally designed for (Fogarty et al., 1998). Still, the drying and wetting of fuels follows the same physical processes all over the world, as does the response of fires in changes of these influences (Taylor, 2001). Thus, reuse of an existing system is generally possible, but this system needs to be chosen with care regarding similarity of intended area of application. Furthermore, it has to be adapted to the specific conditions of the target region.
After having taken into account several fire danger rating indices - such as the Nesterov Index and the Ignition Component (IC) of the United States National Fire Danger Rating System (NFDRS) -, the Fire Weather Index (FWI) of the Canadian Forest Fire Danger Rating System (CFFDRS) was determined to be the index of choice. The reason of this preference was that the FWI can be computed by readily available weather conditions data, which is, according to Fogarty, an essential criterion for successful application of a fire danger rating system (Fogarty et al., 1998). Another aspect is the greater simplicity of the

calculation procedures, compared to systems such as the NFDRS (Moriondo et al., 2006). It was also designed with modularity, adaptability and flexibility in mind (Fogarty et al., 1998).

The FWI system is furthermore based on sound scientific principles, which have been found over a 70-year research period, and it has already been successfully applied to model fire potential in different parts of the world and in a broad range of fuel types (Moriondo et al., 2006, following van Wagner, 1975; Fogarty et al., 1998), whereby it should be noted that the applications in maritime climates (for example British Columbia / Canada; Fogarty et al., 1998) have been the decisive ones. Regions, where the FWI found application include Mexico, south-east Asia, Florida, and Argentina (Moriondo et al., 2006). Because of its adaptability and the studies already conducted the Mediterranean area in this regard, the FWI was found to be the index best suited for application in the study region.

Viegas et al. (2000) conducted a comparative study of different methods of fire danger evaluation in southern Europe, and they were of the opinion that the FWI system components yielded the best overall correlation with fire occurrence in southern Portugal, Spain, France and Italy. The system performed best in forested areas, but was also found applicable to the dry Mediterranean shrubland vegetation (Viegas et al., 2000). Sol (1999) gathered similar results for southeastern France (Moriondo et al., 2006).

Aguado (2003) successfully used the Drought Code (DC) component of the FWI system to model fuel moisture in Andalucía (Spain), validating the results in relation to National Oceanic and Atmospheric Administration (NOAA) Advanced Very High Resolution Radiometer (AVHRR) satellite imagery (Aguado et al., 2003). Viegas (2001) also used the DC to simulate moisture content of several fine fuels in central Portugal and Catalunya (Spain), checking the results against field measurements. He proved that the slow response of live fine fuel moisture content (FMC) to meteorological conditions (mainly precipitation) was well described by the DC (Viegas et al., 2001).

Dimitrakopoulus et al. (2011) evaluated the FWI against Mediterranean conditions in Crete, Greece. The study revealed a high correlation between the new classification for FWI, DMC, DC, and BUI and actual fire occurrence as well as between the amount of measured litter and the Fine Fuel Moisture Code (FFMC).

As a result, Dimitrakopoulus showed that the FWI successfully reflects fire danger and is therefore qualified to serve as fire danger rating index in Mediterranean regions.

Since Viegas first study of FWI performance in southern European regions in 1994, the use of this system is spreading within operational research institutions throughout Europe (Viegas et al., 2001). Today, the Joint Research Center is using the FWI system at European Union level for production of daily fire danger maps in the European Forest Fire Information System (EFFIS).

The adaptation of the FWI mainly takes place by considering the prevailing fuel type of the target region. One approach of adaptation is to calibrate the fuel drying rates used in the models against study results of local field fuel moisture (Taylor, 2001), whereby local day length influence on wetting and drying rates should be taken into account. The second option is to change interpretation of resulting FWI values based on long-term experience. Since the FWI was developed on the basis of a reference fuel type representing vegetation conditions in Canada, the relative numerical outputs point to different grades of fire potential in different fuel types, and do not represent absolute measures (Fogarty et al., 1998). According to McAlpine et al., the relative numerical outputs of the system have to be compared with relevant fire environment characteristics for fires burning under different conditions (McAlpine et al., 1990) in order to calibrate the system in a given local fire environment. Fogarty proposes the usage of fire danger outputs taken from historical weather records that were checked against measures of actually occurred fires (Fogarty et al., 1998, following Kiil et al., 1997; Kruse et al., 1993).

Projected on implementation of the FWI system, threshold values have to be determined depending on historic local fire occurrence to classify system outputs into meaningful levels of danger.

Dimitrakopoulus et al. (2011) found the existing threshold values used for FWI classification to be inadequate for the dry and fire prone eastern Mediterranean climate of Crete. Using logistic regression, he defined reasonable threshold limits appropriate for the Mediterranean area (see table 4.6). He stated, however, that long-time studies still have to be conducted to examine the precise thresholds of fire danger classes according to fire occurrence (Dimitrakopoulos, Bemmerzouk

and Mitsopoulos, 2011).

It is exactly this type of study that has been undertaken in the frame of this thesis. The approach is to counteract the above mentioned deficiency of the original fire danger classification used in the Mediterranean area by using retrospectively calculated fire danger conditions for Sardinia and Crete for the period from 2001 to 2010. Historical weather data from the Weather Underground database is used to create fire danger maps for each day in the mentioned timespan. Two steps are required for the production of these maps: Firstly, the input values for the FWI of each weather station need to be interpolated over the area of Sardinia (and Crete, respectively) for each day, to assign values to stations possibly not having delivered measuring results for the specific date. Inverse Distance Weighting (IDW) is used for the interpolation process, as it is one of the methods proposed by Lawson and Armitage, 2008 for this purpose. Temperature values need to be adjusted in terms of height. Secondly, FWI values (and therefore also values of the subindices ISI, BUI, FFMC, DMC, and DC) are computed for the positions of the used weather stations, because there the input values are most reliable. The index outputs are also interpolated over the islands area and classified using a specific classification scheme (see below) to represent one of six levels of fire danger. The result forms the base for the final fire danger map.

In order to secure the validation of the forecasts, a list of occurred fires is prepared, listing wildfires with date of burning and geographic coordinates, while excluding prescribed burnings by farmers and false detections. The original wildfire hotspot data was therefore filtered, in a way that only hotspots in vegetated areas are selected and that these hotspots are sure to actually represent occurred wildfires. The former is to be done using land cover data, the latter using burned area data. Every hotspot location is then to be tested against the forecasting map of the specific date. It must be checked which of the danger levels was calculated for the location of the hotspot to determine if the computation of the danger level was correct. The FWI is typically classified in six danger levels, from 'Very Low' to 'Extreme'. The idea is to count every hotspot, being situated in an area marked with 'High', 'Very High', or 'Extreme' fire danger, as a successful prediction. The hotspots, however, which appeared in areas with predicted 'Very Low', 'Low' or 'Moderate' danger and therefore representing wildfires occurred in areas marked

as low risk, are counted as a failure.
This process is repeated for different classifications, one used by the European Forest Fire Information System, EFFIS, and another developed in this thesis adapted to conditions on the Isle of Sardinia and the Mediterranean in general. Cluster analysis, specifically the k-means method (developed by Lloyd, 1957), is used for that purpose. The determination of threshold values for the classification is done by partitioning the entirety of resulting index values from the long-term study of the FWI (and ISI, BUI, FFMC, DMC, and DC, respectively) into six groups. The value limits separating two groups will then be used as the thresholds. The results can be found in table 4.5 and 4.6, respectively, listed next to the values used by EFFIS. The latter were retrieved on request from EFFIS staff.

The general aim of a fire danger classification scheme is to give a numerical rating of the severity of daily fire danger correlating with fire potential to be used as guideline for preparedness planning purposes. Various different classification schemes are used for that purpose around the world, partly because there exists no common definition for measuring fire danger. Most of the systems in use have not been quantitatively defined, but have evolved historically by observation and interpretation, making it difficult to compare them from an objective point of view (Taylor, 2001). Turner, in 1973, defined thresholds for fire classes so that each class appeared with the same frequency throughout the year (Turner, 1973). Van Wagner proposed a classification based on an arbitrary setting of average days per year to be be rated as days of extreme fire danger (Wagner, 1987). The classification developed in this thesis pursues the aim of being as objective and realistic as possible. This is achieved by exclusively using statistical results that derived from historically occurred index outputs.
With regard to the FWI system, classifications of four to six classes are in use, depending on the usage of the classes 'Very Low' and 'Very High'. In 1994, Alexander suggested to introduce an additional distinction between 'Very High' and 'Extreme'. His 'Very High'-class would address situations in which fires could still be controlled by conventional techniques, whereas the 'Extreme'-class describes conditions where fires are nearly or completely uncontrollable.

The results yielded by the thesis and the EFFIS classification applied in the long-term study are compared by means of forecasting reliability, to determine which one is the best suited for fire danger forecasting on Sardinia and Crete.
As proposed by Fogarty, the results are validated in order to ensure that the system outputs correlate with actual fire occurrence in the given area of interest (Fogarty et al., 1998).
For validation of a fire danger classification, it must be tested against two kinds of errors (Taylor, 2001):

- **Type I error:** This error occurs if a null hypothesis is rejected although it is true. It is also referred to as a false positive. In the context of wildfire prediction, the error addresses the situation where the system forecasts a severe danger situation, but in fact no fire occurs. The danger is thus over-estimated. These tests were performed for the study area and the testing area, compare figures 5.3, 5.4, 5.8, and 5.9 as well as tables 5.2, and 5.7 of the results for Sardinia and Crete, respectively.

- **Type II error:** In this case, the null hypothesis is false, but fails to be rejected. It is a false negative. In wildfire prediction, this refers to the situation where a serious fire occurs without the system having warned the agencies in charge. Hence, the danger is underestimated. This may lead to critical situations, such as fire management agencies not being sufficiently prepared, which increases the risk of failing the initial fire-fighting. The thesis' fire danger classification has also been tested against the type II error, the results of which can be seen in figures 5.1, 5.2, 5.6, and 5.7, as well as in tables 5.1, and 5.6.

In remote sensing, the terms 'error of commission' and 'error of omission' are utilised synonymously to 'type 1 error' and 'type 2 error', respectively. They are used in the process of 'ground truthing', to address the validity of feature classification. As the type II error is the more significant one among these two errors, it is regularly dealt with first in this thesis.
To get a visual impression of the differences of the two classification schemes, the computed area sizes of each danger class for each of the six indices were also

plotted against the corresponding results of EFFIS. See 5.10, and 5.12 for the results of FWI and FFMC for plottings of the area of Sardinia. For Crete, see 5.11 and 5.13.
The found classification scheme is later integrated into a self-updating web server environment, which uses crowd sourced weather data and open source software to produce daily, regional fire danger forecasting maps for the Isle of Sardinia.

3.2 Using hotspots for validation purposes

Among the different means of wildfire information collection, conventional ground-based in situ observation is still frequently used. Further approaches are air-borne human and photographic surveillance, as well as space-borne remote sensing. While ground based and air based observations are at variance with each other in terms of quality, density, and frequency among each other, remote sensing provides possibilities to acquire uniform, high-resolution fire information on a repetitive basis (Li et al., 2000).
Currently, satellite based approaches to detection of large-area burnings can be differentiated into two categories: Firstly the detection of active fires, called hotspots, where fire thermal energy is used for the determination of burnings. Secondly, the detection of areas having recently been burned (Fraser, Li and Cihlar, 2000).
Since hotspots only represent a series of snapshots of burn activity, a considerable amount of wildfires might be missed due to cloud cover, which makes detection impossible (Fraser, Li and Cihlar, 2000). This circumstance poses an intrinsic limitation for optical remote sensing applications (Li et al., 2000). Burned area detection possesses the advantage, that the indication of fire remains detectable for a longer period. Usually, the comparison of Normalized Differenced Vegetation Index (NDVI) information for a region of interest for two points of time, called differencing, is used in this context. Automated post-fire burn interpretation can, however, be complicated. Pixels might erroneously be classified due to different reasons for NDVI decrease, such as drought, seasonal vegetation senescence, and timber harvesting (Fraser, Li and Cihlar, 2000).
The potential of detecting actively burning fires was observed by Dozier (1981),

using scenes from the Advanced Very High Resolution Radiometer (AVHRR). Automatic algorithms for fire pixel detection have been a subject of research since that time (Nakayama et al., 1999). Contextual algorithms, which include the pixel values of the neighbours of a potential fire pixel in the decision process, were first described by Justice and Dowty (1994; Nakayama et al., 1999).
Today, as it is the case for Giglios enhanced algorithm developed for MODIS MOD14 scenes (Giglio et al., 2003), multi-spectral thresholds based on the local surrounding are used for automated fire pixel classification (Nakayama et al., 1999).
Hotspots have regularly been used in recent decades to access the occurrence of wildfires, most recently among others by Feltman et al., 2012 and Koubarakis, Kontoes and Manegold, 2012.

3.3 Open Source software and licenses

Open Source software is computer software with the source code made publicly available and distributed under an Open Source license. Development mostly takes place in a collaborative manner open to the public. Several free platforms exist for the developers to exchange source code and information. Their efforts are being supported by the non-profit Free Software Foundation (FSF), through preparation of legally binding licenses. Several conditions must be met for software to be regarded as Open Source, according to the Open Source Initiative (OSI, see OSI, 2013e):

- **Free redistribution:** The license must neither require a fee to be paid nor must it restrict a party from selling the software.
- **Availability of source code:** The software must include the source code and, if it is not packaged directly, it has to be made available as a free web download.
- **Possibility of derived works:** Modifications and secondary works based on the original one must be allowed, as well as the distribution of these works under the same conditions as the original software.

Furthermore, Open Source software may not discriminate against persons or groups, and may not impose restrictions upon other software. It has to be distinguished from freeware, which is distributed free of charge but without source code and without the right to apply changes. Several Open Source licenses are in use, differing by their degree of restrictiveness.

- `BSD` and `MIT` are permissive licenses, allowing anyone to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the software without restrictions (OSI, 2013d, OSI, 2013b).
- `GPL`, the GNU General Public license provided by FSF, restricts usage in a way that derived works can only be published under the same license. Proprietary products must not include GPL software (OSI, 2013a).
- `LGPL` is the abbreviation for the GNU Lesser General Public License created by FSF. It softens the GPL in that software under this license is allowed to be distributed in package together with proprietary products, as long as the free software is technically clearly separated from the proprietary parts (OSI, 2013c).
- `CC` refers to the Creative Commons license, which allows the distribution of copyrighted works if certain conditions set by the author are met (for example the exclusion of commercial use; CC, 2013).
- `ODBl` is the Open Database license used by the OpenStreetMap project, among others. It is a license created especially for databases, to allow users to freely share, use, and modify data without infringing copyrights of others (ODC, 2013).

Data distributed as 'Public Domain' can be used by anyone without restrictions of any kind.
In geographic context, the Open Source Geospatial Foundation (OSGeo) is of major importance for spatial software development. OSGeo is a non-profit and non-governmental organisation purposed to support the development of free geospatial software. This is done by provision of financial, organisational and legal help.

It also acts as an advocacy institution for the complete Open Source geospatial community and provides a forum and infrastructure for collaboration. OSGeo is also responsible for planning and realising the Free and Open Source Software Conference for Geospatial (FOSS4G), which is held annually in varying locations. Projects developed under the patronage of OSGeo include `GRASS`, `Quantum GIS`, `gvSIG`, `GDAL/OGR`, `FDO`, `GEOS`, `MapGuide Open Source`, `GeoServer`, `GeoTools`, `PostGIS`, and `GeoNetwork` (OSGeo, 2013).

3.4 Applications and libraries

This section lists the software programs used in the creation of this document, as well as for the operation of the fire danger forecasting platform. See table 3.1 for an overview.

Table 3.1: Applications and Libraries used in the frame of this thesis

(additional packages and modules shown in square brackets)

Category	Name	License	Purpose	Version
Spatial / Fire-related	GDAL/OGR	MIT/X	Geofile Conversion	1.7.3
	GMT	GNU GPL	Map creation	4.5.2
	GRASS GIS	GNU GPL	Spatial analysis	6.4.1
	MapServer	X/MIT	WMS	5.6.5
	OpenLayers	BSD	Web-Mapping	2.12
	PROJ.4	MIT	Coord. transformat.	4.7.0
	pyfwi	new BSD	FWI calculation	1.0
Various	Apache	Apache 2.0	Webserver	2.2.16
	Debian	GNU GPL	Operating System	6.0.5
	Gnuplot	Gnuplot Lic.	Plots	4.4.0
	Graphviz	Eclipse	Schematic diagrams	2.26.3
	ImageMagick	Apache 2.0	Image processing	6.6.0-4
	KVM	GNU GPL	Virtualisation	0.12.5
	Midnight Com.	GNU GPL	File Management	4.7.0.9
	OpenSSH	BSD	Secure File Transfer	5.5
	Python	Python Lic.	Scripting	2.6.6
	[gdal]	MIT	Geofile import	1.7.3
	[PIL]	PIL lic.	Image import	1.1.7
	[shapely]	BSD	Geoprocessing	1.2.1
	R	GNU GPL	Statistics	2.11.1
	[gstat]	GNU GPL 2	Interpolation	0.6-24
	[rgdal]	GNU GPL 2	Geofile export	0.7-22
	[sp]	GNU GPL 2	Interpolation	0.9-99
	Rsync	GNU GPL 3	Backups	3.0.7
	TexLive	LPPL	Setting of thesis	2009-11
	CUED PhD	GNU GPL	LATEX template	1.1
	Vim	GNU GPL	Writing of thesis	7.2.445

3.4.1 Python

Python is a general purpose, multi-paradigm programming language, distributed under a GPL compatible license and supporting several operating systems. The comprehensive standard library is extended by third-party software, called packages. A variety of packages exists especially for scientific purposes, for example, linear algebra (NumPy), signal and image processing (SciPy, PIL), and interactive 2D/3D plotting (matplotlib; Holhs, 2013). The fields of GIS programming, mapping, image processing and analysis are also addressed by several additions, for example, by the Python Spatial Analysis Library (PySAL), Rtree for spatial indexing features, psycopg for accessing a PostgreSQL/PostGIS database, and Shapely for planar objects manipulation. With GeoDjango, a readily deployable geospatial web framework is available based on the Python language. Furthermore, interfaces have been developed for communication between Python and several stand-alone software products. Regarding GIS, Python bindings exist for the R statistics software, PROJ.4 and GDAL/OGR, amongst others. The other way around, GIS software suites have themselves been enhanced by implementing Python interpreters to give users extended scripting capabilities. The Open Source products GRASS, QGIS, and SAGA use Python as a scripting language, as is the case with the commercial GIS software ArcGIS. Due to its strong support for spatial computing, Python has rapidly become a favoured language GIS programming (Gillies, 2013a).

3.4.2 GDAL/OGR

The Geospatial Data Abstraction Library (GDAL) is a conversion library for gridded geospatial data. It contains methods for importing and exporting a wide variety of geographic file formats. Bindings exist for Python, R, Java, Perl, C#.NET, and Ruby (Warmerdam, 2013a). Besides its main function, several command line utilities exist for performing projection of gridded data (gdalwarp), aggregation of multiple files (gdal_merge), or vectorisation of gridded data (gdal_polygonize; Warmerdam, 2013b). The GDAL library has been implemented in a variety of both free and commercial GIS environments, like GeoDjango, GeoServer, GMT, GoogleEarth, GRASS, gvSIG, MapGuide,

mapnik, MapServer, Opticks, Orfeo Toolbox, OSSIM, Quantum GIS, PostGIS, R, and SAGA GIS, as well as ArcGIS and the Feature Manipulation Engine (FME; Warmerdam, 2013e).

The OGR Simple Features Library is integrated into the source code of GDAL. It is capable of reading and writing a considerable range of vectorial data file formats and of accessing a variety of spatial databases. Moreover, it supports the projection of vectorial data between coordinate systems, as well as the extraction of features via SQL queries and clipping of data by a bounding box (Warmerdam, 2013c). Both GDAL and OGR have been profoundly used in the production of this thesis, in particular for conversion between file formats (Shape to GML and vice versa), clipping of areas, projecting of source data to different coordinate systems (WGS84/UTM for calculations and printed maps, and WGS84/Pseudo Mercator for display in the web mapping client), and for importing data for further analysis in Python.

3.4.3 Further applications

- **R:** R is both a programming language and a software environment for statistical computing and visualisation. The statistical techniques comprise, among others, linear and non-linear modelling, time-series analysis, classification and clustering. It is easily extensible via packages, and runs on a variety of platforms (R Foundation, 2013). In the frame of this thesis, R together with three packages related to spatial data, gstat, rgdal, and sp, has been utilised for Inverse Distance Interpolation (IDW) of weather conditions data and fire danger index outputs, as well as for k-means clustering in fire danger level classification.

- **PROJ.4:** PROJ.4 is the cartographic projections library internally used by GDAL/OGR. It was originally developed by Gerald Evenden of USGS, and is now maintained in the frame of the GDAL/OGR libraries. Bindings exist for Python, Visual Basic, MySQL, Ruby. The software is intended to project lists of coordinates between coordinate systems, and is distributed under the MIT license (Warmerdam, 2013d). During this thesis, it has mainly been used in the context of data preparation, in particular for pro-

jection of geographic coordinates listed in tabular raw data to WGS84/UTM coordinates.

- **GMT:** The `Generic Mapping Tools (GMT)` are a collection of software programmes for manipulation and visualisation of georegistered datasets. Data processing capabilities include filtering, gridding, and projecting. The map creation supports data plots of two and three dimensions, producing `Encapsulated PostScript (EPS)` or `PostScript (PS)` vectorial files (Wessel, 2013). All map representations in this document have been created using `GMT`.

- **GRASS:** The `Geographic Resources Analysis Support System (GRASS)` is a free and widely used Geographic Information System and focuses on geospatial data management and analysis as well as image processing and spatial modelling. It is scriptable via `Perl` and `Python` and can be completely controlled by using the operating system shell. Furthermore, it is free software and runs on a variety of platforms. During this thesis, it was used for a wide variety of GIS purposes, especially in the context of data preparation.

- **MapServer:** `MapServer` is software for publishing spatial data to the web. It is capable of producing Open Geospatial Consortium (OGC) conformal web services, such as `Web Map Service (WMS)` and `Web Feature Service (WFS)`. These can then be consumed by Desktop GIS software or a web-based client like `Open Layers`. It is used on the platform developed in this thesis for producing daily fire danger forecasting WMS (University of Minnesota, 2013).

- **OpenLayers:** `OpenLayers` is a software library intended to produce dynamic maps in a web browser. It is implemented in JavaScript, an interpreted programming language aimed at web applications, and supports a variety of sources, including local `Geography Markup Language (GML)`, `Keyhole Markup Language (KML)` and `GeoJSON` files as well as `Web Map Services` and `Web Feature Services` from a remote server (OpenLayers

Dev Team, 2013). OpenLayers is the graphical front-end used in the fire danger platform to display forecasting maps of fire danger.

- **Sencha Touch:** `Sencha Touch` is a software framework purposed for development of mobile web applications, using `JavaScript` and fifth generation `Hypertext Markup Language (HTML)` technology, being the main markup language for web site creation. It supports several types of mobile devices, including Android, iOS, Blackberry, and Kindle (Sencha Inc. 2013). This thesis applies SechaTouch in connection with `OpenLayers` to build a mobile web application that displays fire danger levels for the current day in the area of Sardinia. Additionally, it also shows the geographic position of the user.

- **GML:** The `Geography Markup Language Encoding Standard (GML)` of the Open Geospatial Consortium (OGC) is an Extended Markup Language (XML) grammar for description of geographical features. It is an open interchange format primarily developed for geographic transactions on the internet. The `OGC Web Feature Service` Standard is built upon `GML`. Many GIS software products implement `GML` because of its usability as geographic data exchange format (OGC, 2013b).

  Given that `GML` is easily created by a script and simply editable in a text editor, this format was chosen to be the standard throughout the creation of this thesis. Also, this format was extensively applied for exchange purposes between GIS software as well as for simple manipulation, collection, and extraction of data.

3.5 Datasets

Table 3.2: Data used in the frame of this thesis

Category	Name	License	Purpose	Years
Fire related	FIRMS active fire	Public Domain	Validation	2001-12
	FIRMS burned areas	CC BY-SA 3.0	Validation	2001-10
	wunderground.com	free (non com.)	Forecasts	2001-12
Various	CORINE LC (v15)	free	Validation	2000/06
	GADM (v2.0)	CC BY-NC-SA 3.0	Maps	2012
	geonames.org	CC BY-SA	Maps	2012
	GSHHS (v2.2.0)	GPL	Maps	2011
	OpenStreetMap	ODbL	Web	2012
	SRTM (v4.1)	free (non com.)	Maps	2008

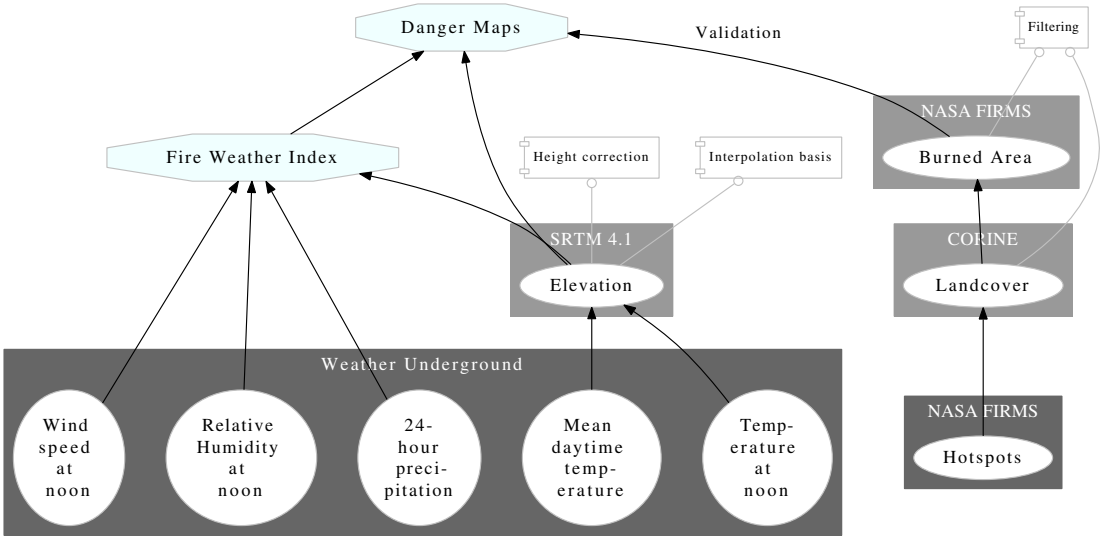


Figure 3.1: Flow of data processing

The figure shows the connection of data sources in the process of the fire danger map preparation. Data sets of primary importance are underlaid with dark-grey boxes, while the sources used for filtering or enhancement are shown within a light-grey background. FIRMS and CORINE data, shown on the right, is used for validation purposes.

3.5.1 FIRMS Active Fire and MODIS MOD14/MYD14

FIRMS (Fire Information for Resource Management System) Active Fire Data is a near real-time collection of worldwide fire locations. It derived from the MODIS MOD14/MYD14 Fire and Thermal Anomalies products and is processed by LANCE. The latter is a group of four near real-time data systems operated by the NASA / GSFC Earth Science Data Information System (ESDIS), purposed to supporting the land and atmosphere science community (NASA, 2012b, NASA, 2012a).

MODIS (Moderate-Resolution Imaging Spectroradiometer) is a sensor on board the NASA Terra/Aqua satellites, which were launched in 1999 and 2002, respectively. The sensors are capable of capturing data in 36 spectral bands, with wave lengths ranging from 0.4 µm to 14.4 µm (NASA, 2012f). Two bands are pictured at a spatial resolution of 250 m, five of 500 m, and the remainder at 1000 m. A +/-55° scanning pattern at the 705 km - orbit of Aqua/Terra allows for global coverage every one to two days (NASA, 2012e). The two satellites orbit the earth in opposite directions: Terra crosses the equator from north to south, while Aqua crosses it from south to north (NASA, 2012d). MODIS data is provided free of charge. MODIS fire products not only provide information about location and timing of currently burning fires, but also about the smouldering ratio and their instantaneous radiative power (Giglio et al., 2003).

The determination of pixels potentially containing active fires is done by an algorithm originally created by Kaufmann et al. in 1998 and significantly enhanced by Giglio (Giglio et al., 2003). The program marks areas which significantly differ from their surroundings in respect of heat release, with a spatial extent of one km^2. This algorithm uses the MODIS 4 µm and 11 µm channels for active fire detection (named T_4 and T_{11} hereafter), with the channel at 12.0 µm (T_{12}) for cloud masking.
The 250 m resolution red and near infrared channels at 0.65 µm, and 0.86 µm are also used for cloud masking and suppressing of false alarms, and have been addressed as ρ0.65 and ρ0.86 in Giglio's publication.

The 2.1 µm band is utilised to reject water induced false alarms, its reflectance is denoted by ρ2.1. An overview of the MODIS channels used by the detection algorithm is given in table 3.3.

Table 3.3: MODIS channels used in detection algorithm

as of Giglio et al., 2003, edited

Channel number	**Resolution**	**Central wave length**	**Purpose**
1	250 m	0.65 µm	Sun glint and coastal false alarm rejection; cloud masking
2	250 m	0.86 µm	Bright surface, sun glint, and coastal false alarm rejection; cloud masking
7	500 m	2.1 µm	Sun glint and coastal false alarm rejection
21	1000 m	4.0 µm	High-range channel for active fire detection
22	1000 m	4.0 µm	Low-range channel for active fire detection
31	1000 m	11.0 µm	Active fire detection, cloud masking
32	1000 m	12.0 µm	Cloud masking

The proceeding of the identification of potential fire pixels conducted by NASA using Giglios methodology is described in detail hereafter. In a first preparative step, obvious non-fire pixels are excluded. Potential fire pixels are then determined by the following equation.

$$T_4 > 310K,\ \Delta T > 10K,\ \rho 0.86 < 0.3,\ \text{where } \Delta T = T_4 - T_{11} \tag{3.1}$$

Those pixels that remain are checked in six subsequent tests to ensure they do contain one or several active fires (as of Giglio et al., 2003):

The absolute threshold test:

$$T_4 > 360K \text{ (320 } K \text{ at night)} \tag{3.2}$$

Contextual tests:

$$\Delta T > \overline{\Delta T} + 3.5\delta_{\Delta T} \tag{3.3}$$

$$\Delta T > \overline{\Delta T} + 6K \tag{3.4}$$

$$T_4 > \overline{T_4} + 3\delta_4 \tag{3.5}$$

$$T_{11} > \overline{T_{11}} + \delta_{11} - 4K \tag{3.6}$$

$$\delta_4{}^{'} > 5K \tag{3.7}$$

The gained fire locations are made publicly available through FIRMS as both tabular and point-shapefile download for the last 24h, 48h and seven days, where every fire location (called 'hotspot' hereafter) represents the center of a 1 km grid cell in the original data, which has been flagged as containing one or several fires by the algorithm. Keyhole Markup Language (KML) files can be downloaded for the last 24h and 48h. Additionally, Web Map Services (WMS) are published by FIRMS, which can be integrated in a Desktop- or WebGIS client.

Beside the near real-time data, FIRMS also offers archived historical data with a time lag of approximately two months after the present date (NASA, 2012a).

While the production of the former focuses on quick delivery to the user and is therefore processed with less accurate ancillary data, the historical data is gained by considerably more processing of the raw data. The main difference is the accuracy of the hotspot locations: Definitive orbit information of the satellites is used for historical data, while in the case of near real-time data, only predictive orbiting information is used. This historical data is processed by MODAPS (MODIS Adaptive Processing System) and is labelled 'science quality' by FIRMS (NASA, 2012c). Only this hotspot data has been used in the frame of this thesis.
The website and data is hosted by NASA's Earth Observing System / Data and Information System (EOSDIS). It is available from November 2000 until today and can be requested online via a web form. Beside the latitude and longitude of every fire detected, the data also includes the exact date and time of detection, the measured brightness value, the satellite carrying the detecting sensor (Aqua / Terra), a confidence value, and the version of the algorithm used to determine if a pixel contains an active fire.

The complete attributes of FIRMS historical hotspot data are as follows (with exemplary values; NASA, 2013):

- **latitude:** 36.49 (Vertical centre of 1 km fire pixel)
- **longitude:** 22.501 (Horizontal centre of 1 km fire pixel)
- **brightness:** 317.9 (Brightness temperature of fire pixel in Kelvin)
- **scan:** 1.1 (Along Scan pixel size)
- **track:** 1 (Along Track pixel size)
- **acq_date:** 2012-08-14 (Date of acquisition by MODIS)
- **acq_time:** 2040 (Time of acquisition by MODIS)
- **satellite:** T (A/B for Aqua and Terra, respectively)
- **confidence:** 95 (Indication of quality of detection, 0-100%)
- **version:** 5.0 (MODIS Collection and source data reference)

- **bright_t31:** 298.2 (Channel 31 brightness temperature in Kelvin)
- **frp:** 20.4 (Fire radiative power of pixel in MegaWatts)

Hotspot data is commonly used for fire danger calibration purposes because of its frequent global coverage and rapid availability (Groot et al., 2006, following Csiszar et al., 2005 and Dymond et al., 2006).

Processing in the frame of this thesis:
The tabular data has been sorted (ascending by date) and the relevant regions extracted via bounding box coordinates of Sardinia and Crete, respectively. The coordinates of the resulting tables were projected from geographic coordinates to WGS84 / UTM zone 32N for Sardinia and to WGS84 / UTM zone 35N for Crete, respectively, by use of the software `PROJ.4`.

3.5.2 FIRMS burned area / MODIS MCD45A1

The MODIS burned Area product (MCD45) is a worldwide, monthly collection of burned areas and is part of the 'MODIS Land collection 5 product suite' since year 2000. It is available as HDF-EOS format or as monthly GeoTIFFs provided by the University of Maryland / FIRMS and hosted on NASA's data and information system EOSDIS. While the original data is available in sinusoidal projection, the FIRMS data has been reprojected to Plate-Carrée projection and offered in form of several sub-continental windows (Boschetti, Roy and Hoffmann, 2009). Window '08' covers Europe (with bounding box coordinates: longitude -11 to 35, latitude 33 to 70) and is the burned area data subset used in this thesis.
While the original HDF data includes eight layers, only two are available in GeoTIFF format. These are the ones storing information on dates at which burnings took place and confidence of the detection: In the layer containing the dates of burnings, every pixel either has the value 0, if the area represented by the pixel is unburned, or a value from 1 to 366, representing the approximate Julian day of burning from eight days before the beginning of the respective month to eight days after the end of the month. The Julian day of the first of the relevant month is used as part of this month's file name. Additional values represent

snow, water, or lack of data (Boschetti, Roy and Hoffmann, 2009). The pixels can obtain several further values indicating snow, water or lack of data, as shown in the following listing.

- **0:** unburned
- **1-366:** approximate Julian day of burning within the month
- **900:** snow or high aerosol
- **9998:** water bodies (internal)
- **9999:** water bodies (seas and oceans)
- **10000:** not enough data to perform inversion throughout the period
- **-32768:** pixel not covered by any MODIS tile

Pixels in the confidence layer can adopt four values, whereby the first two are determined by passing tests depending on changes in time between satellite overpasses:

- **1:** most confidently detected pixels
- **2:** pixels where several overpasses predict the same change
- **3:** pixels selected in the first stage of the contextual analysis
- **4:** pixels selected in the second stage of the contextual analysis

The algorithm developed to determine burned areas from MODIS satellite imagery is described in detail by Roy (Roy et al., 2005).
The original MODIS burned area product can be ordered through the Land Processes Distributed Active Archive Center (LP-DAAC). In addition, HDF as well as GeoTIFF files are available from the University of Maryland via a File Transfer Protocol (FTP) server. An online registration has to be performed in order to obtain the files.

Processing in the frame of this thesis:
The GeoTIFF files for each month between January 2000 and December 2010 have been vectorised and saved as GML files, with pixel values (Julian day of burning) transferred as attribute to the resulting polygons. All of the vector files were then clipped with the bounding boxes for the study site of Sardinia and the testing site of Crete. The resulting extracts have been projected from Plate-Carrée to WGS84 / UTM zone 32N for Sardinia and for WGS84 / UTM zone 35N for Crete, respectively. The features inside the resulting vectorial files have then been split up depending on their date of burn via the `Python` script 'mcd45split.py' (A.1.2), to receive single GML files containing all the burned area polygons for the specific day between 2001 and 2010. While in the GeoTIFFs, the Julian day of the first of the month is used as part of the file name, this has been transferred to Gregorian dates as part of the filenames of the resulting files, to make comparisons with other datasets easier.

3.5.3 Weather Underground

Weather Underground has been the world's first online weather service providing comprehensive real-time global weather information (WUI, 2013c). It is run by Weather Underground, Inc., a commercial entity evolved from the University of Michigan in 1995, based in San Francisco (WUI, 2013a). It features global real-time weather conditions together with charts and graphs as well as historic data, and detailed local forecasts. The site furthermore offers warning and advisories regarding critical weather situations, a tornado tracking product, and marine weather information. Weather Underground permits free access to the data and allows to make copies of it for non-commercial use (WUI, 2013f). The information provided can be used via a web browser, a mobile device or queried directly via an application programming interface (API).
Weather Underground is a global, community based project. Local observations are contributed by users via Personal Weather Stations (PWS). With over 36,000 PWS, Weather Underground provides the world's largest network of privately run weather stations (WUI, 2013a). While the majority of contributing stations

is located in the US, several thousands can be found in Europe (1,103 in Italy, and 160 in Greece as of January 2013; WUI, 2013d).
The contributors range from amateur meteorologists to scientists at the U.S. National Weather Service. Collecting of local weather data through the use of personal weather stations is of importance for a variety of industries (WUI, 2013g). The main motivation for submitting the data to Weather Underground is probably the gaining of localised weather forecasting information for the own region of interest.
A PWS is an outdoor instrument measuring weather conditions. In its basic form for use in the Weather Underground network, it includes a thermometer, barometer, anemometer, hygrometer, and wind vanes, but can also be equipped with sensors measuring UV index, leaf wetness, soil moisture and temperature, and water temperature (WUI, 2013g). The quality and number of possible measurements depends on the type of the used station as well as on the correct installation. The pricing ranges from $100 for low-end hardware to $1,000 for a sophisticated technical solution (WUI, 2013g).
To assure the quality of the measured weather data, several instructions are given by the website regarding the correct positioning of a personal weather station. Among other constraints, it has to be set up 15 m from the nearest paved surface as well as from the nearest tree or body of water, 10 m above the ground and 2 m above surrounding obstructions for the thermometer, hygrometer, rain collector, and anemometer to work correctly.
The measurements of the station are then transmitted to a PC and uploaded to Weather Underground meteorological conditions database. Several free and Open Source applications are available to perform this task automatically at a given time interval. Direct transmitting to the database without having to use a PC is also possible with sophisticated weather station hardware. In addition to PWS, data of several official weather stations (mostly located at airports) can be gathered via the Weather Underground website and database.
Besides viewing the data in a web browser or mobile device, Weather Underground also offers access to the database by use of an application programming interface (API). In this way, developers can implement weather data in own applications by requesting them from the Weather Underground server via Hypertext

Transfer Protocol (HTTP). The response is formatted in JavaScript Object Notation (JSON) or Extensible Markup Language (XML). Unlike viewing the data via a web browser, it is necessary to create a user account for using the API. Below is an exemplary response in JSON format:


```
{
        "history": {
                "observations": [
                {
                "date": {
                "year": "2012",
                "mon": "10",
                "mday": "22",
                "hour": "23",
                "min": "46",
                "tzname": "Europe/Rome"
                },
                "tempm":"19.7",
                "dewptm":"17.2",
                "hum":"85",
                "pressurem":"1016.5" }
                ],
        }
}
```


The advantage of the data being well structured in JSON or XML notation is that it can easily be parsed by a programming language. Interfaces exist for several languages, in particular `PHP`, `Ruby`, `Python`, `ColdFusion`, and `JavaScript` (WUI, 2013b). In this thesis, `Python` is used for accessing the Weather Underground database and extracting the relevant data. The exact output of a weather station depends on its technical capabilities. Listed below are the attributes and values returned by querying the personal weather station 'ISARDEGN6' in Genneruxi, Sardinia for historical data of October, the 22th, at noon time in 2012. The syntax for the query as well as its returned output are as follows:

`http://api.wunderground.com/api/[user_id]/history_20101022/q/pws:isardegn6.json`

- **year:** 2012 (year of observation)
- **mon:** 10 (month of observation)
- **mday:** 22 (day of month of observation)
- **hour:** 12 (hour of observation)
- **min:** 00 (minute of observation)
- **tzname:** Europe/Rome (timezone)
- **tempm:** 25.5 (temperature in °C)
- **tempi:** 77.9 (temperature in °F)
- **dewptm:** 20.0 (dewpoint in °C)
- **dewpti:** 68.0 (dewpoint in °F)
- **hum:** 71 (humidity in percent)
- **wspdm:** 4.8 (wind speed in km/h)
- **wspdi:** 3.0 (wind speed in m/h)
- **wgustm:** 11.3 (speed of gusts in km/h)
- **wgusti:** 7.0 (speed of gusts in m/h)
- **wdird:** 301 (wind direction in degrees)
- **wdire:** WNW (wind direction)
- **pressurem:** 1014.4 (air pressure in hPa)
- **pressurei:** 29.96 (air pressure in inches of mercury)
- **precip_ratem:** 0.0 (hourly rate of precipitation in mm)
- **precip_ratei:** 0.00 (hourly rate of precipitation in inch)
- **precip_totalm:** 0.5 (accumulated rain in mm)

- **precip_totali:** 0.02 (accumulated rain in inch)
- **solarradiation:** 258 (solar radiation in Watts/m^2)
- **softwaretype:** weatherlink.com 1.10 (uploading software)

For historical data, also a summary for the specific day is included, giving daily average values for the attributes listed above as well as minima and maxima values for temperature, humidity, dew point, air pressure, and wind speed.
The ten-day forecast gives information for the next ten days to come. The query syntax for station 'ISARDEGN6' in Genneruxi, Sardinia is as follows:

```
http://api.wunderground.com/api/[user_id]/forecast10day/q/
  pws:isardegn6.json
```

The following attributes are returned:

- **day:** 22 (day of month of forecast)
- **month:** 10 (month of forecast)
- **year:** 2012 (year of forecast)
- **yday:** 295 (day of year of forecast)
- **hour:** 23 (hour of forecast)
- **min:** 00 (lowest forecasted values in °C)
- **tz_long:** Europe/Rome (timezone)
- **high**
  - **fahrenheit:** 77 (highest forecasted values in °F)
  - **celsius:** 25 (highest forecasted values in °C)
- **low**
  - **fahrenheit:** 61 (lowest forecasted values in °F)
  - **celsius:** 16 (lowest forecasted values in °C)

- **pop:** 90 (probability of precipitation)
- **qpf_allday**
  - **in:** 0.57 (quantitative precipitation forecast during 24h in inch)
  - **mm:** 14.5 (quantitative precipitation forecast during 24h in mm)
- **qpf_day**
  - **in:** 0.50 (quant. precipitation forecast during daytime in inch)
  - **mm:** 12.7 (quant. precipitation forecast during daytime in mm)
- **qpf_night**
  - **in:** 0.06 (quant. precipitation forecast during nighttime in inch)
  - **mm:** 1.5 (quant. precipitation forecast during nighttime in mm)
- **snow_allday**
  - **in:** 0 (predicted height of snow cover during 24h in inch)
  - **cm:** 0 (predicted height of snow cover during 24h in mm)
- **snow_day**
  - **in:** 0 (predicted height of snowcover during daytime in inch)
  - **cm:** 0 (predicted height of snowcover during daytime in mm)
- **snow_night**
  - **in:** 0 (predicted height of snowcover during nighttime in inch)
  - **cm:** 0 (predicted height of snow cover during nighttime in mm)
- **maxwind**
  - **mph:** 6 (maximum speed of predicted wind in m/h)
  - **kph:** 10 (maximum speed of predicted wind in km/h)
  - **dir:** WNW (wind direction of predicted maximum wind speed)

  - **degrees:** 303 (dir. of predicted max. wind speed in degrees)

- **avewind**
  - **mph:** 3 (average speed of predicted wind in m/h)
  - **kph:** 5 (average speed of predicted wind in km/h)
  - **dir:** West (wind direction of predicted average wind speed)
  - **degrees:** 266 (direc. of predicted avg. wind speed in degrees)
- **avehumidity:** 76 (average forecasted rel. humidity in%)
- **maxhumidity:** 98 (maximum forecasted rel. humidity in%)
- **minhumidity:** 69 (minimum forecasted rel. humidity in%)

It is to be noted, however, that working with the Weather Underground API happens within certain access limits: The free developer account is allowed 500 calls to the database per day, with ten calls being the limit per minute. Access to historical data is also granted within these constraints. Extended access has to be paid for, with three grades of access offering increasing possibilities. At the highest grade, which features a hourly ten-day forecast, yesterday's weather summary, dynamic animated satellite images, and current tropical storms, the costs for 1,000,000 calls per day (the highest amount possible) account for US$750 per month. If historical data is wished to be incorporated in the contract, the costs sum up to US$5,750. However, for the operation of the fire weather forecasting platform for Sardinia, developed in this thesis, the free developer account was found to be sufficient (WUI, 2013e).

3.5.4 Further datasets

- **CORINE Land Cover seamless vector data:** The dataset covers land usage in the area of Europe, divided into 44 classes. The data is an updated version of the database originally created in the 1990s as part of the European Commission programme to **Coo**rdinate **In**formation on the

Environment (CORINE). It is now distributed by the European Environment Agency (EEA) providing updated data based on the status both of 2000 and 2006. Both sets are revised regularly; version 15 being the one used in this thesis (EEA, 2012a; EEA, 2012b). As the area of Greece is not covered in the dataset of 2006, this data is gathered from the CLC version of 2000.

CLC is available at a spatial resolution of 1:100.000. Only features possessing a minimum size of 25 ha with a minimum width of 100 m are mapped (Büttner and Kosztra, 2007). SPOT-4 as well as IRS LISS III satellite scenes with a geometric accuracy of below 25 m are used as source data. The thematic accuracy is expressed to be above 85% in the technical guideline. This has been validated for the year 2000 dataset (Büttner and Kosztra, 2007, following Büttner and Maucha, 2006).

- **OpenStreetMap:** The community-driven `OpenStreetMap` project aims at creating a free, editable map of the whole world. The contributions are done by capturing geographical features using a Global Positioning System (GPS) device or digitising free aerial photography and subsequently uploading the created data to the `OpenStreetMap` database. An Extended Markup Language (XML) formatted file representing the current state of the complete database is available for download and further use (OpenStreetMap Community, 2013). An `OpenStreetMap` Web Map Service is used in the frame of this thesis as the default background layer for both the main forecasting platform and the mobile application.

Chapter 4

Realisation

4.1 Preparation

4.1.1 Selection of weather stations

37 weather stations on Sardinia and ten on Crete were available through the Weather Underground network at the time of writing. But since some stations have started operating just recently, they could not be used for the long-term analysis in this thesis. The entirety of stations available on Sardinia and Crete is listed in tables 4.1 and 4.2, with stations used for the long-term study being highlighted. Official weather stations are preceded with the abbreviation 'OWS', the abbreviation 'PWS' marks personal weather stations. The locations of these applied stations are shown in maps 4.1 and 4.2.

Table 4.1: Weather stations on Sardinia (Weather Underground)

(stations used are highlighted in grey)

Type	Place name	Station name	Start	Lat.	Lon.	Height
OWS	Alghero	LIEA	1996	40.63°	8.89°	23 m
	Cagliari / Elmas	LIEE	1996	39.24°	9.06°	4 m
	Capo Bellavista	LIEB	1996	39.93°	9.71°	150 m
	Capo Caccia	LIEH	1996	40.56°	8.16°	204 m
	Capo Carbonara	LIEC	2006	39.10°	9.51°	118 m
	Capo Frasca	LIEF	1996	39.74°	8.46°	95 m
	Capo S. Lorenzo	LIEL	–	39.50°	9.63°	5 m
	Decimomannu	LIED	1996	39.35°	8.97°	28 m
	Fonni	LIEN	–	40.12°	9.25°	1029 m
	Guardiavecchia	LIEG	–	41.22°	9.40°	159 m
	Olbia	LIEO	1996	40.90°	9.51°	13 m
	Oristano / Fenosu	LIER	1996	39.90°	8.64°	11 m
	Perdasdefogu	LIEP	–	39.67°	9.44°	606 m
	Tortoli	LIET	1996	39.92°	9.68°	7 m
PWS	Asuni	IORISTAN5	2010	39.87°	8.95°	233 m
	Borutta	ISARDEGN15	2009	40.52°	8.74°	488 m
	Genneruxi	ISARDEGN6	2008	39.23°	9.13°	13 m
	Carbonia	ISARDEGN17	2009	39.17°	8.52°	100 m
	Castadas	ISARDEGN38	2011	39.21°	9.54°	49 m
	Desulo	ISARDEGN22	2010	40.01°	9.22°	869 m
	Guspini	ISARDEGN10	2008	39.55°	8.65°	100 m
	Iglesias (1)	ISARDEGN35	2011	39.32°	8.52°	294 m
	Iglesias (2)	ISARDEGN32	2011	39.31°	8.52°	226 m
	Oristano	IORISTAN2	2008	39.91°	8.59°	4 m
	Orosei	ISARDEGN39	2011	40.38°	9.70°	15 m
	Ozieri	ISARDEGN27	2010	40.58°	9.00°	403 m
	Perdasdefogu	IOGLIAST2	2011	39.68°	9.44°	611 m
	Porto Torres	ISSPORTO1	2007	40.84°	8.40°	3 m
	Quartu Sant'Elena	ISARDEGN8	2008	39.24°	9.19°	27 m
	San Giov. Suergiu	ICISANGI2	2010	39.13°	8.49°	10 m
	San Sperate	ISARDEGN40	2011	39.36°	9.00°	42 m
	San Vito	ISARDEGN9	2008	39.44°	9.54°	15 m
	Sassari	IITALIAS2	2011	40.75°	8.54°	142 m
	Sestu	ISARDEGN5	2008	39.30°	9.08°	56 m
	Set. San Pietro	ICAGLIAR7	2011	39.29°	9.18°	76 m
	Siamaggiore	IORISTAN4	2009	39.97°	8.63°	29 m
	Sinnai	ICAGLIAR6	2011	39.31°	9.20°	152 m

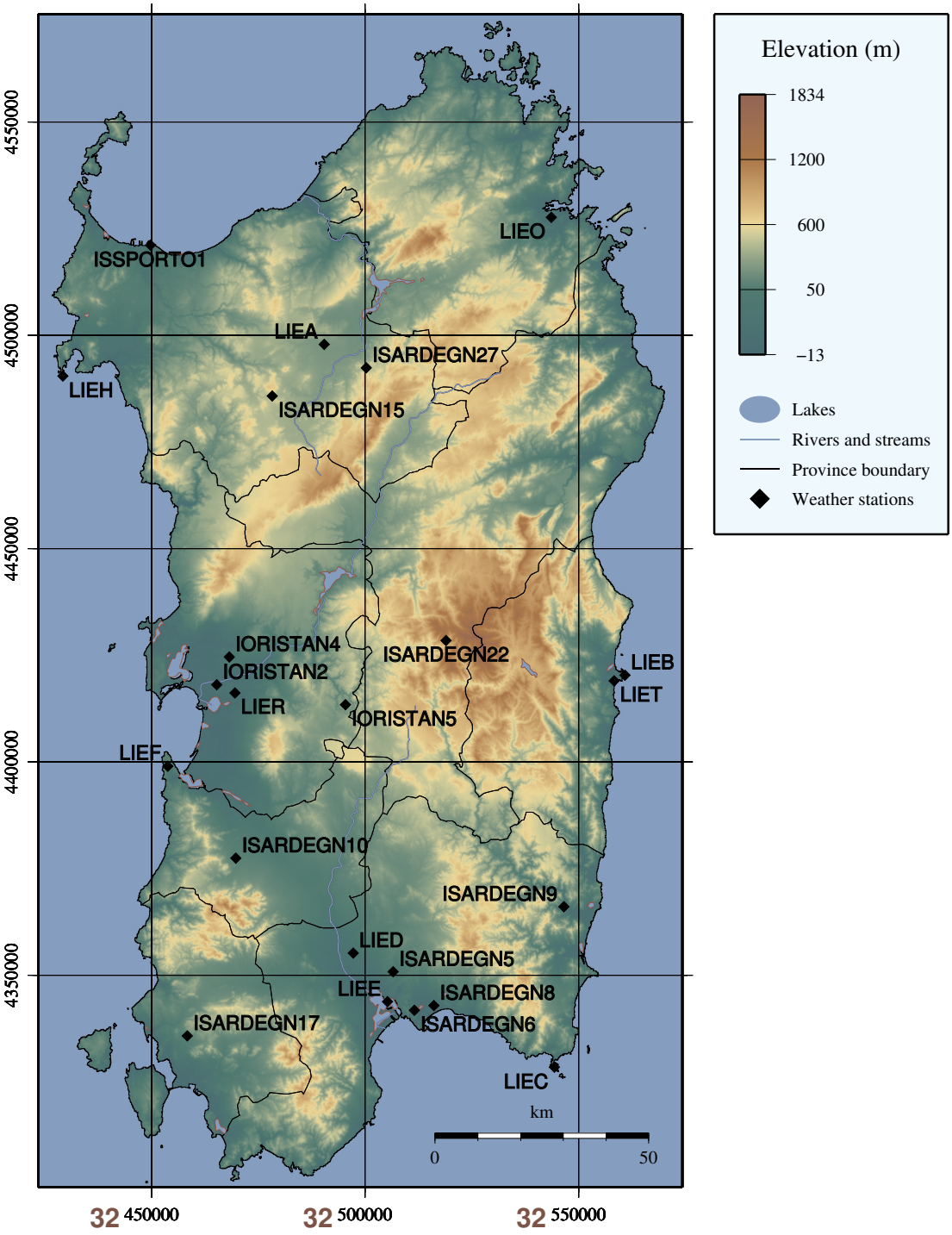


Figure 4.1: Weather stations on Sardinia

Sources:
Elevation: [SRTM41] Jarvis et al., 2008
Rivers/Lakes: [GSHHS] Wessel and Smith, 1996
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012
Weather stations: [wunderground.com] WUI, 2013d

Table 4.2: Weather stations on Greece (Weather Underground)

(stations used are highlighted in grey)

Type	Place name	Station name	Start	Lat.	Lon.	Height
OWS	Heraklion	LGIR	1996	35.34°	25.18°	39 m
	Kasteli	LGTL	1996	35.19°	25.33°	336 m
	Sitia	LGST	1996	35.22°	26.10°	28 m
	Souda	LGSA	1996	35.48°	24.12°	151 m
PWS	A. Mar. Tsalikaki	IHERAKLI4	2010	35.33°	25.09°	20 m
	Heraklion	IHERAKLI1	2005	35.34°	25.15°	30 m
	Plakias	IRETHYMN4	2011	35.19°	24.41°	61 m
	Makrigialos	ILASITHI2	2008	35.04°	25.98°	0 m
	Chania	ICHANIAC2	2007	35.53°	24.07°	122 m
	Treis Vagies	IHERAKLI3	2010	35.32°	25.10°	11 m

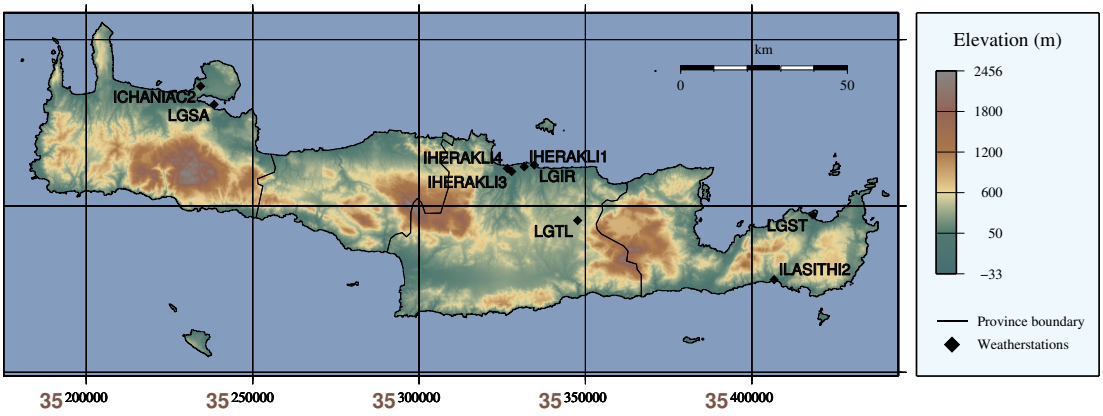


Figure 4.2: Weather stations on Crete

Sources:
Elevation: [SRTM41] Jarvis et al., 2008
Rivers/Lakes: [GSHHS] Wessel and Smith, 1996
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012
Weather stations: [wunderground.com] WUI, 2013d

4.1.2 Selection of hotspots for validation

The MODIS hotspot detection algorithm classifies areas significantly warmer than their surroundings. Although the algorithm is finely calibrated, the detected locations do not necessarily indicate wildfires, but they could be industrial fires or false detections caused by sea water reflections as well (Giglio et al., 2003). Furthermore, the detections could result from farmers burning land for clearance reasons, which is the majority of reasons in the Mediterranean area.
As the fire danger platform is intended to forecast areas showing high ignition danger due to weather conditions, intentionally started fires for land clearing reasons should not be included in the hotspot data set. This data is intended to be used as a means of validation to determine the quality of fire danger forecasting, and it is therefore of the upmost importance that the dataset is as reliable as possible.
It so becomes evident that special care had to be taken of the selection of valid hotspots that were taken into account, and that a considerable proportion of hotspot data had to be excluded in order to get a picture representing true conditions on Sardinia and Crete.
After the clipping of the relevant timespan from 2001 to 2010 of the FIRMS hotspot dataset, two steps were necessary to ensure that the used hotspot locations actually represent detected wildfires. It is to be accepted that a considerable proportion of the original data is excluded during this process. Firstly, the locations of hotspots are checked against the land usage they occurred on, using the CORINE land cover dataset. Table 4.3 for Sardinia and table 4.4 for Crete are set in relation to the number of hotspots and the area size of the affected land usage class, listed in descending order by hotspots per km^2. In the case of Sardinia, dump sites followed by commercial or industrial units form the most affected land usage classes. Fires on these areas are sure to have nothing to do with the wildfires which are under investigation in this thesis. It is therefore reasonable to use only hotspot locations that coincide with natural vegetation, consisting of natural grasslands, sclerophyllous vegetation, coniferous forest, mixed forest and broad-leaved forests, which are highlighted in tables 4.3 and 4.4. The checking of hotspot locations against CORINE land use polygons is done by the script 'pipogrextr.py' (see A.1.1).

The results are shown in maps of extraction via land cover. By this proceeding, it is assured that the detected fire actually occurred in a vegetated area. The second step makes sure that the detection was in fact caused by a fire, and was not the result of a false alarm. Using the FIRMS burned areas dataset, it is checked for every hotspot if an area comprising the hotspot location was marked as having burned within the next two weeks after the hotspot detection date. The extended length of the chosen timespan is due to possible cloud coverage.
The final product is a set of 152 hotspot locations for Sardinia regarding the period of 2001 to 2010, and a collection of only seven hotspots for Crete. The original amount of hotspots for Sardinia and Crete has been 4261 and 575, respectively. The computation is done in the script 'pipburnedarea_extr.py', see A.1.3.
The hotspots are used during the following procedure to test the validity of predictions in form of a hit ratio. A hit ratio in this context is a number indicating the percentage of correct prediction results for a set of testing points. The ratio is used to determine the success of the prediction methodology. It is named 'recall ratio' hereafter.

Table 4.3: Hotspot exclusion via CORINE land cover data on Sardinia

(sorted via hotspots per area, areas used are highlighted in grey)

Corine-Category	**Hotspots**	**Area (km²)**	**Hotspots/km²**
Dump sites	1	0.9065	1.103098
Indust. or commercial units	30	91.4575	0.328021
Inland marshes	2	6.7118	0.297984
Burnt areas	2	7.2479	0.275943
Fruit trees and berry plant.	23	96.5260	0.238278
Olive groves	81	341.9639	0.236867
Sport and leisure facilities	3	13.8861	0.216044
Non-irrigated arable land	1092	5244.0786	0.208235
Salt marshes	9	47.1042	0.191066
Airports	2	11.9059	0.167984
Annual crops	4	24.8060	0.161251
Beaches, dunes, sands	2	12.6291	0.158364
Complex cultivation patterns	140	907.8599	0.154209
Natural grasslands	232	1620.4706	0.143168
Discontinuous urban fabric	58	452.4221	0.128199
Agriculture, with natural veg.	253	2038.9910	0.124081
Mineral extraction sites	5	53.2415	0.093912
Sparsely vegetated areas	27	291.8003	0.092529
Salines	2	21.7276	0.092049
Sclerophyllous vegetation	584	6448.3952	0.090565
Transitional woodland-shrub	10	114.8998	0.087032
Agro-forestry areas	147	1723.4722	0.085293
Bare rocks	6	80.9870	0.074086
Coniferous forest	49	671.3756	0.072984
Sea and ocean	1151	16036.9332	0.071772
Vineyards	4	58.6232	0.068232
Mixed forest	10	160.9956	0.062113
Water bodies	5	95.6217	0.052289
Broad-leaved forest	154	3185.9038	0.048338
Continuous urban fabric	1	38.6162	0.025896
Coastal lagoons	1	112.6981	0.088733
Road and rail networks	0	4.4135	0
Port areas	0	12.3928	0
Construction sites	0	0.9346	0
Green urban areas	0	4.7005	0
Pastures	0	0.3700	0
Moors and heathland	0	38.9801	0
Water courses	0	1.0540	0
Estuaries	0	0.6853	0

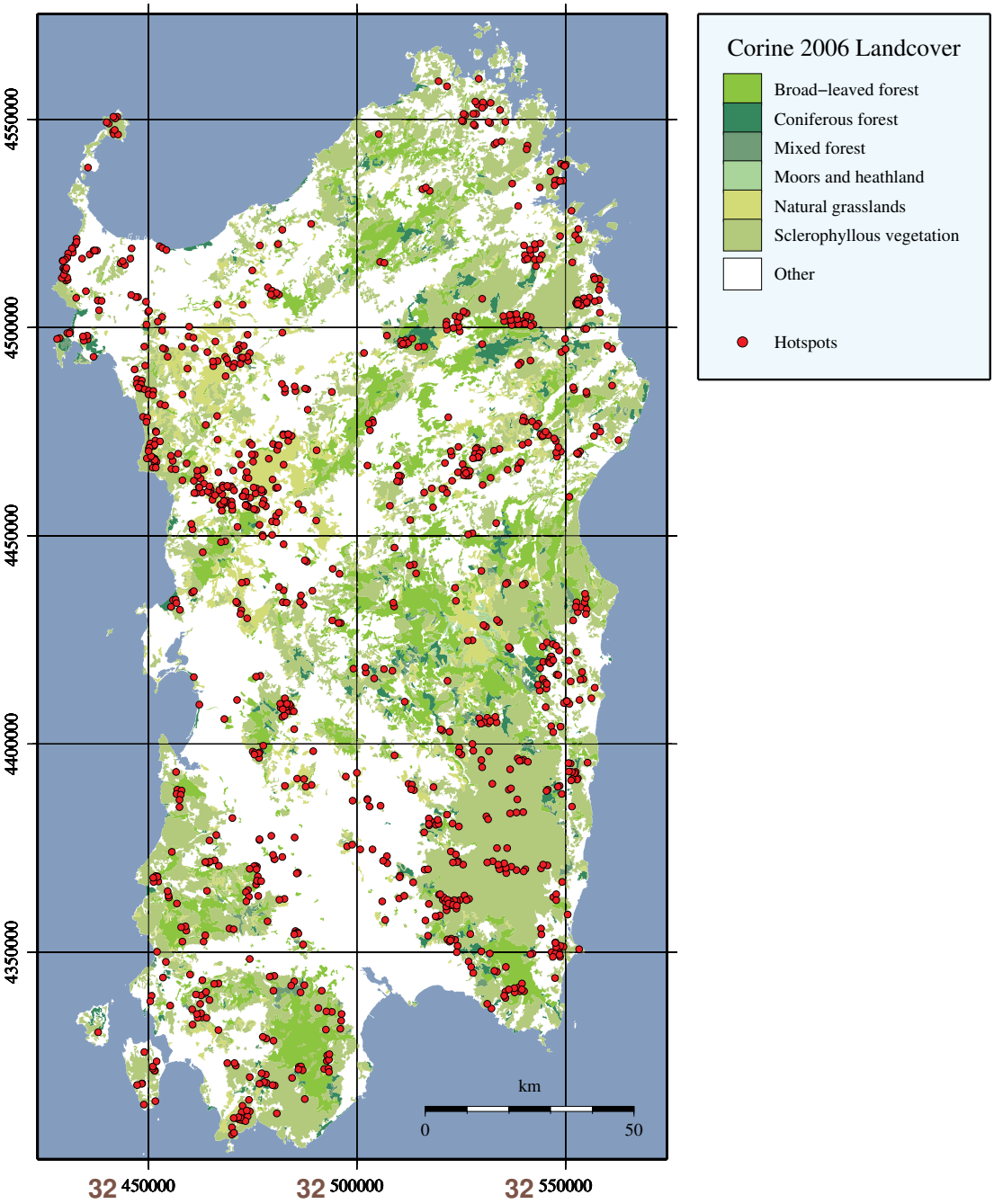


Figure 4.3: Sardinia's hotspots in areas of natural vegetation 2010-2010

Sources:
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012

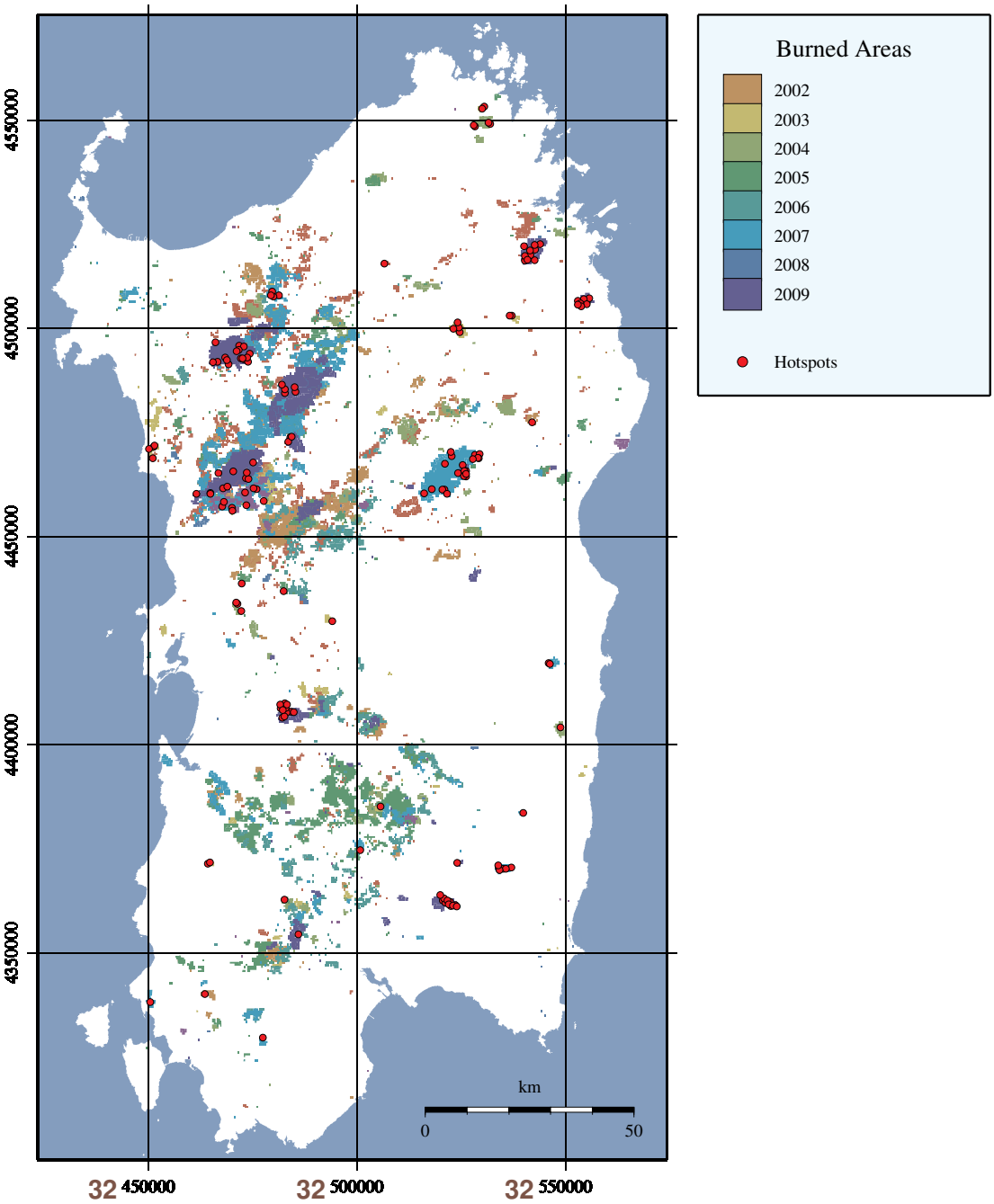


Figure 4.4: Sardinia's hotspots in areas detected as having been burned, 2001-2010

Sources:
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012

Table 4.4: Hotspot exclusion via CORINE land cover data on Crete

(sorted via hotspots per area, areas used are highlighted grey)

Corine-Category	**Hotspots**	**Area (km²)**	**Hotspots/km²**
Non-irrigated arable land	9	36.0174	0.249879
Pastures	3	16.3158	0.183871
Natural grasslands	163	1672.7745	0.097443
Agriculture, with natural veg.	76	821.1213	0.092556
Sclerophyllous vegetation	170	1978.1210	0.085940
Indust. or commercial units	1	13.0317	0.076736
Broad-leaved forest	4	62.9304	0.063562
Complex cultivation patterns	24	471.6457	0.050886
Olive groves	95	1938.6379	0.049003
Transitional woodland-shrub	13	312.5686	0.041591
Fruit trees and berry plantat.	2	58.1159	0.034414
Vineyards	6	214.9977	0.027907
Sparsely vegetated areas	5	297.5254	0.016805
Coniferous forest	3	214.8890	0.013961
Discontinuous urban fabric	1	75.7709	0.013198
Continuous urban fabric	0	3.5344	0
Road and rail networks	0	3.8211	0
Port areas	0	1.9650	0
Airports	0	10.9060	0
Mineral extraction sites	0	4.4180	0
Construction sites	0	4.6088	0
Sport and leisure facilities	0	10.7949	0
Permanently irrigated land	0	16.5255	0
Mixed forest	0	5.6463	0
Moors and heathland	0	17.1519	0
Beaches, dunes, sands	0	7.5327	0
Bare rocks	0	54.1865	0
Water bodies	0	1.3863	0
Sea and ocean	0	17609.5244	0

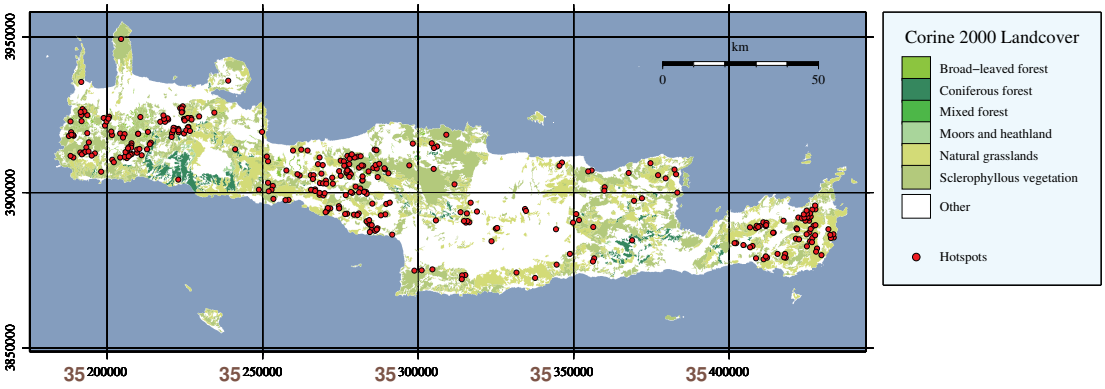


Figure 4.5: Crete's hotspots in areas of natural vegetation 2001-2010

Sources:
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012

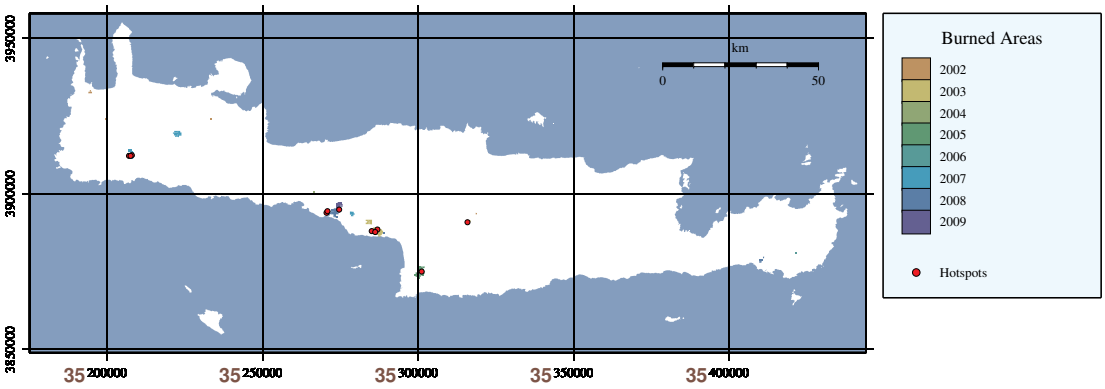


Figure 4.6: Crete's hotspots in areas detected as having been burned, 2001-2010

Sources:
Shoreline/Province Boundaries: [GADM] University of Berkeley, 2012

4.1.3 Determination of thresholds

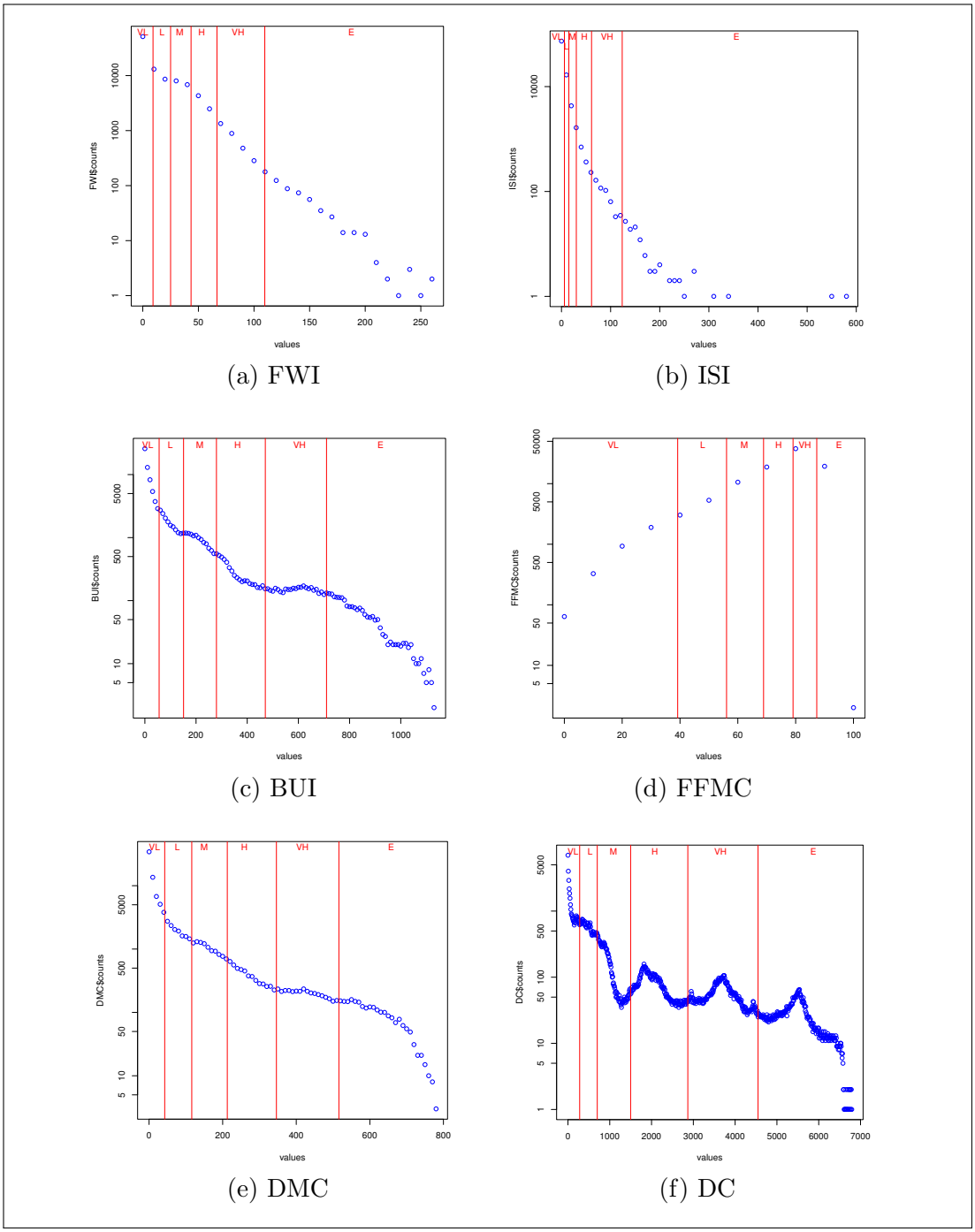
Thresholds, which are boundary values separating different states of affairs, are needed for the classification of fire danger. In this thesis, they are defined in a statistical way, using the k-means clustering algorithm. The distribution of index results computed in the long-term study was analysed for each of the six indices, and six severity classes were determined for each index. The resulting threshold

values for the FWI and its subindices are given in table 4.6 and 4.5, respectively (the EFFIS threshold values have been generously provided by JRC). Figure 4.7 shows a representation of the cluster analysis results. The thresholds found in this thesis for Sardinia differ considerably from the ones used by EFFIS for the region of Europe and the ones calculated by van Wagner for Canada. This is reasonable since the relative numerical outputs of the FWI indicate different grades of fire potential in different fuel types, and do not represent absolute measures. If the index is applied in areas not featuring the reference fuel type, the outputs have to be interpreted according to long-term experience. Dimitrakopoulos (2011) evaluated the FWI on the Isle of Crete, which has a flora comparable to the one of Sardinia. It is therefore comprehensible that his classification is similar to the one found in this thesis for Sardinia.

In order to compare the different classification schemes more efficiently, the thesis focuses on the boundary separating the three low fire danger classes ('Very Low' to 'Moderate') from the three high danger classes ('High' to 'Extreme'). This is regarded to be the most significant value, since it represents a 'turning point' in fire danger prediction. This implies that if a classification has a fairly high value for this turning point, the three low fire danger classes are more predominantly represented than the three higher ones.

Dimitrakopoulos (2011) calculated an FWI result of 48.2 for this turning point. A very similar value of 43.4 was found in this thesis.

EFFIS uses a more intemperate classification, which exhibits a value of 21.3 for this turning point. By this, EFFIS can also address fire danger conditions in the even more endangered countries of northern Africa, such as Algeria, Libya, Tunisia, and Egypt. In contrast, a threshold value of only 9 was proposed by van Wagner for Canadian fire danger conditions (Wagner, 1987). The higher sensitivity of this scheme compared to the ones mentioned so far is due to the Canadian vegetation being dominated by forests, so that the intensity of a propagating flame front, the actual predication of the FWI index, is much higher in case of a fire compared to bush- and shrublands in Mediterranean ecosystems.



(a) FWI (b) ISI

(c) BUI (d) FFMC

(e) DMC (f) DC

Figure 4.7: Results of k-means cluster analysis

The cluster limits resulting from the k-means algorithm are shown as red, vertical lines, together with the underlying distribution of index values from the long-term study. These are drawn as blue circles, with the Y-axis showing the number of occurrences of a specific value, using a logarithmic scale. The X-axis represents the range of index results (labelling: VL = Very Low, L = Low, M = Moderate, H = High, VH = Very High, E = Extreme).

Table 4.5: Thresholds of FWI subindices

Index	**Class**	**Thesis**	**EFFIS**
Fine Fuel	Very Low	< 39.2	< 82.7
Moisture Content	Low	$\geq 39.2, < 56.1$	$\geq 82.7, < 86.1$
(FFMC)	Moderate	$\geq 56.1, < 68.9$	$\geq 86.1, < 89.2$
	High	$\geq 89.9, < 79.1$	$\geq 89.2, < 93$
	Very High	$\geq 79.1, < 87.3$	–
	Extreme	≥ 87.3	≥ 93
Duff Moisture	Very Low	< 42.5	< 15.7
Code	Low	$\geq 42.5, < 116.0$	$\geq 15.7, < 27.9$
(DMC)	Moderate	$\geq 116.0, < 212.6$	$\geq 27.9, < 53.1$
	High	$\geq 212.6, < 346.1$	$\geq 53.1, < 140.7$
	Very High	$\geq 346.1, < 515.9$	–
	Extreme	≥ 515.9	≥ 140.7
Drought	Very Low	< 280.5	< 256.1
Code	Low	$\geq 280.5, < 701.2$	$\geq 256.1, < 334.1$
(DC)	Moderate	$\geq 701.2, < 1499.7$	$\geq 334.1, < 450.6$
	High	$\geq 1499.7, < 2868.0$	$\geq 450.6, < 749.4$
	Very High	$\geq 2868.0, < 4548.2$	–
	Extreme	≥ 4548.2	≥ 749.4
Initial Spread	Very Low	< 6.0	< 3.2
Index	Low	$\geq 6.0, < 15.0$	$\geq 3.2, < 5.0$
(ISI)	Moderate	$\geq 15.0, < 30.2$	$\geq 5.0, < 7.5$
	High	$\geq 30.2, < 61.2$	$\geq 7.5, < 13.5$
	Very High	$\geq 61.2, < 123.6$	–
	Extreme	≥ 123.6	≥ 13.4
Build Up	Very Low	< 55.7	< 24.2
Index	Low	$\geq 55.7, < 151.2$	$\geq 24.2, < 40.7$
(BUI)	Moderate	$\geq 151.2, < 279.7$	$\geq 40.7, < 73.3$
	High	$\geq 279.7, < 470.7$	$\geq 73.3, < 178.1$
	Very High	$\geq 470.7, < 709.9$	–
	Extreme	≥ 709.9	≥ 178.1

Table 4.6: Fire Weather Index (FWI) thresholds

Scheme / Class	Thesis (Sardinia)	EFFIS (Europe)	Dimitra'los (Crete)	v. Wagner (Canada)
Very Low	< 9.2	< 5.2	< 24.5	≤ 1
Low	$\geq 9.2, < 25.0$	$\geq 5.2, < 11.2$	$\geq 24.5, < 39.3$	$\geq 2, < 4$
Moderate	$\geq 25.0, < 43.4$	$\geq 11.2, < 21.3$	$\geq 39.3, < 48.2$	$\geq 5, < 8$
High	$\geq 43.4, < 66.7$	$\geq 21.3, < 38.0$	$\geq 48.2, < 51.3$	$\geq 9, < 16$
Very High	$\geq 66.7, < 109.7$	$\geq 38.0, < 50.0$	$\geq 51.3, < 60.8$	$\geq 17, < 29$
Extreme	≥ 109.7	≥ 50.0	≥ 60.8	≥ 30

4.2 Statistical Methodology

4.2.1 Inverse Distance Weighting (IDW)

IDW is a two-dimensional, exact interpolation method to produce a continuous surface from irregularly spaced data points. Such a surface is utilised to compare sets of data points covering an identical region, or to analyse a given dataset regarding a specific purpose, such as gradients (Shepard, 1968). IDW represents a useful tool in the analysis of areal data via computer mapping. It is one of the most frequently used deterministic models in spatial interpolation (Lu and Wong, 2008, following Burrough and McDonnell, 1998), since it can be computed efficiently. Moreover, the model is straightforward and easy to interpret (Lu and Wong, 2008). Its substantial premise is that the rate of correlations and similarities between neighbouring points is inversely related to the distance of their locations (Yasrebi et al., 2009).

However, studies such as the one conducted by Fotheringham and O'Kelly (1989) have revealed that the decrease in spatial connection between two points cannot be seen as proportional to distance. The distance weight is therefore often modified by a power or exponential function for more realistic modelling results.

Lloyd (2005) applied the power of two for interpolation of precipitation, which is a frequently used setting (Lu and Wong, 2008).
Still, it has to be acknowledged that the IDW methodology encompasses several limitations. Most notably, the inverse distance weights are not determined by the empirical data, but have to be chosen preliminarily by the user. Additionally, the distance-decay relationship is assumed to be constant over the spatial extent, without consideration of the distribution of data in the area of interest. Also, it is not possible to estimate the variances of predicted values in locations that are not sampled, as can be conducted with geostatistical methods such as Kriging (Lu and Wong, 2008, following Burrough and McDonnell, 1998). Hence, IDW provides no internal measure of prediction quality. Its results need to be validated using independent data.
Equation 4.1 shows details for Inverse Distance Weighted interpolation, utilising a power to be defined by the user (β; following Bartier and Keller, 1996):

$$z_{x,y} = \frac{\sum_{i=1}^{n} z_i d_{x,y,i}^{-\beta}}{\sum_{i=1}^{n} d_{x,y,i}^{-\beta}} \tag{4.1}$$

Here $\boldsymbol{z_{x,y}}$ represents the point to be calculated and $\boldsymbol{z_i}$ is the control value for the i^{th} point of the data sample. The term $d_{x,y,i}^{-\beta}$ is a weight that defines the relative importance of control point $\boldsymbol{z_i}$ in the interpolation procedure, where $\boldsymbol{d_{x,y,i}}$ addresses the distance between $\boldsymbol{z_{x,y}}$ and $\boldsymbol{z_i}$. $\boldsymbol{\beta}$ is the exponent defined by the user (Bartier and Keller, 1996).
A power of two in the interpolation function is, as done by Lloyd (2005), used in the present thesis for interpolating precipitation. This setting is also used for interpolating temperature, relative humidity, and wind speed values. Since IDW does not consider differences in height, temperature values are preliminarily recalculated to sea level before the interpolation process, using a lapse rate of 6.49 K / 1,000 m. This procedure is also undertaken in the FWI implementation of the Canadian Forest Service (NRC, 2013). Afterwards, the interpolated values are again recalculated according to the specific heights. The purpose of the

interpolation of these values is to avoid data gaps, so that values can be assigned to locations of weather stations not having sent measuring data. IDW has also been used for the interpolation of results from FWI as well as from ISI, BUI, FFMC, DMC, and DC computations. This interpolation steps forms the basis of the creation of fire danger maps. IDW is one of the three methods proposed by Lawson et al. in the weather guide for the Canadian forest fire danger rating system for interpolation of weather parameters (Lawson and Armitage, 2008), and is the one used by the Canadian Forest Service in the local FWI implementation (NRC, 2013).

4.2.2 k-means clustering

Cluster analysis methods are used to classify observations and variables into disjoint partitions (Lee and Macqueen, 1980). The partitioning is done in a way that data points belonging to one cluster are similar, while points from different clusters are dissimilar (Ding and He, 2004). The methods can be separated into hierarchical and non-hierarchical approaches. The latter usually use an initial, arbitrary classification, which is consequently modified through an iterative revision process aimed at minimising the variations within the clusters, or equivalently at maximising the variations between the clusters (Lee and Macqueen, 1980). One of the most efficient methods of data clustering is the k-means method, which minimises the sum of squared errors for the partitioning (Ding and He, 2004). The computation is shown in equation 4.2; following Ding and He, 2004:

$$J_K = \sum_{k=1}^{K} \sum_{i \in C_k} \left(x_i - m_k\right)^2 \tag{4.2}$$

where $(x_1,...,x_n) = X$ is the data matrix, $m_k = \sum_{i \in C_k} x_i / n_k$ is the centroid of cluster C_k, and n_k is the number of points in C_k.
Three steps are required in the process of sorting m data points into k clusters (Lee and Macqueen, 1980):

- **Step 1:** The first arbitrary number of data units are taken for the first k clusters, containing one point each.

- **Step 2:** The remaining m to k data units are assigned to one of the k clusters, depending on the shortest distance between mean value (centroid) of the cluster and the data unit. After having gained a member, the centroid is recomputed for the specific cluster.

- **Step 3:** When each data unit has been classified to one of the k clusters, the preliminary determination of centroids is finished, and all the data points are again assigned to one of the clusters according to the shortest distance to a centroid. After the re-assignment, the centroid for each cluster is computed once again.

Step 3 can be conducted for a specific number of times or until a certain condition is met, such as the absence of change in the variance within the clusters (Lee and Macqueen, 1980). While k-means is known to operate efficiently, the time requirements of this method are proportional to the product of number of data units and the number of clusters per iteration, which proves computationally expensive for large datasets (Alsabti, Ranka and Singh, 1997).
In this thesis, the k-means clustering algorithm was used for partitioning the data range of the long-term study FWI results into six classes representing fire danger, named 'Very Low', 'Low', 'Moderate', 'High', 'Very High', and 'Extreme'. This step was conducted equivalently regarding the FWI subindices ISI, BUI, FFMC, DMC, and DC. The cluster borders are then used as threshold values for classification in the forecasting process. K-means differs from other popular clustering algorithms, such as ISODATA, in that it needs the number of desired classes to be predefined. Because the number of intended fire danger classes was known, k-means was given preference over ISODATA for efficiency reasons.

4.3 Implementation

4.3.1 Processing chain

For the long-term study of hotspot recall ratio as well as for operating the fire danger forecasting platform, it is the daily fire danger forecasting maps that form the substantial background for analysis.

A fully automated work flow has been designed for preparing the daily maps. This high degree of automating was necessary because of two reasons: First, maps for each of the six indices for each day between 2001 and 2010 needed to be calculated in the frame of the long-term study, both for Sardinia and Crete, summing up to 43,824 resulting maps (3652 days x 6 indices x 2 regions of interest).
This enormity of processing steps makes automation inevitable. Secondly, automation is required for fire danger forecasts on the web platform. Its processing takes place on a server with forecasts for ten days to come for each of the indices on a daily basis. The production of 60 maps each day at early morning is, at least in the frame of this thesis, only possible in a technically automated manner.
Scripting in the `Python` language is used for the implementation of the required working steps. Each script serves a specific function and has to be executed on a specific point in the production line. This is done by two master shell scripts listed in A.3.1 for processing of historic, and A.3.2 for forecasting data.
In the cases of missing values, Inverse Distance Weighting (IDW) is used as a statistical means of interpolating weather condition values between weather stations.
Since IDW is incapable of incorporating terrain heights in the interpolation process, all temperature values are modified in terms of elevation in that they are intermediately calculated to sea level, as it is done in the Canadian FWI implementation (NRC, 2013)
All scripts are described in detail in the following pages and are listed completely in the appendix. Figure 4.8 on page 107 shows the working steps and the order of execution in a flow chart. The processing mainly takes place in two sets of steps. The first set comprises the preprocessing of historical weather data, which is needed as a basis for predictions for the current and the following days. It as visualised as a light-grey block in the figure mentioned. The second set constitutes the steps for processing the forecasted weather data, shown in the dark-grey block on the right side. Building on these two preprocessing activities, the actual fire danger forecasts and the presentation of the results is carried out, visualised by the two dark-grey blocks in the upper right.
The following pages outline the scripts listed in figure 4.8 with regard to their functionality.

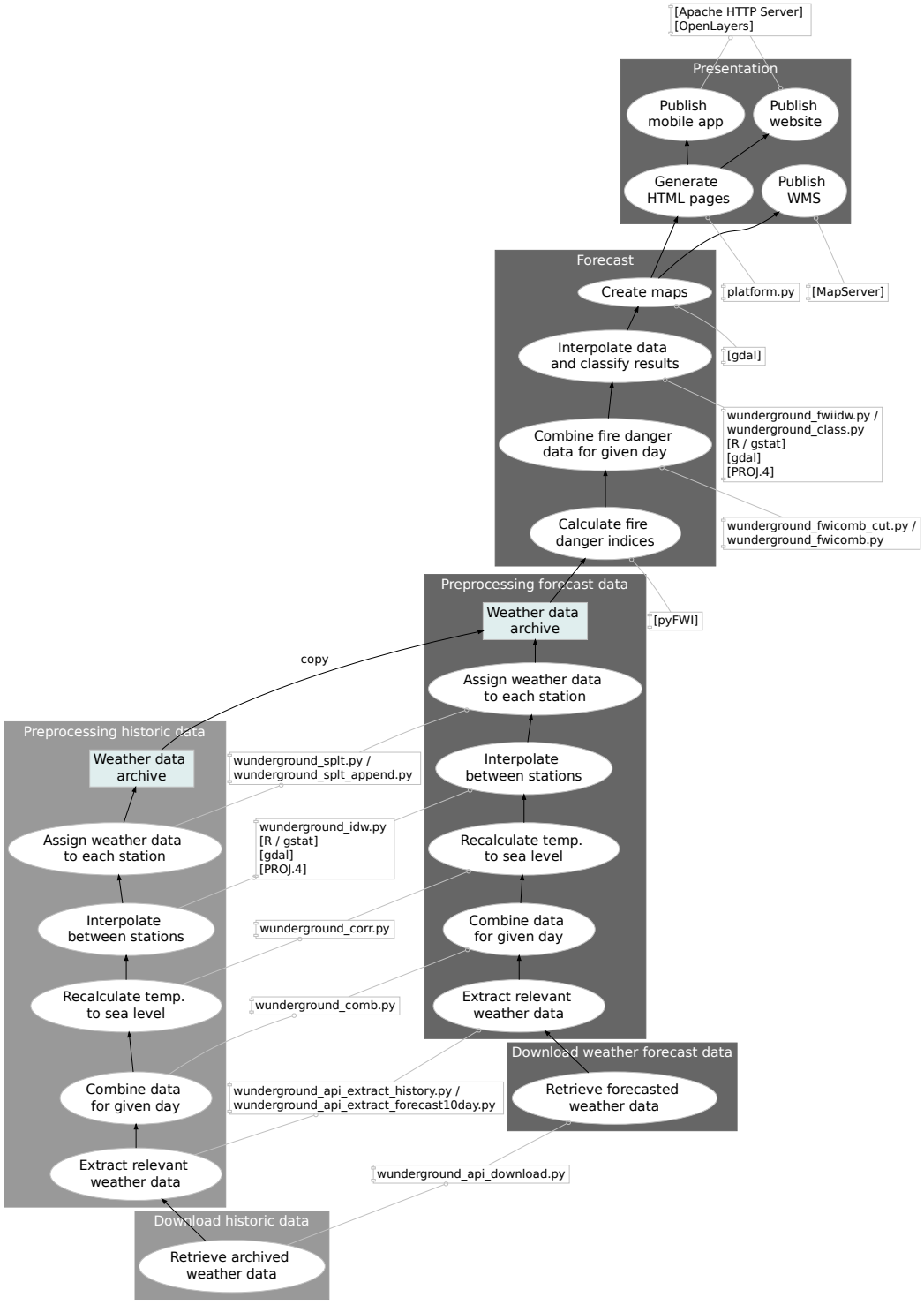


Figure 4.8: Processing chain of the thesis' fire danger forecasting platform

- **wunderground_api_download.py**
  The task of the script is the obtaining of previous day's data as well as predicted weather data for the desired stations from the Weather Underground database by using the Application Programming Interface (API). The previous day's data is necessary because the FWI is a cumulative method, in that the output of the previous day serves as input for present day calculations and forecast. It is therefore essential to have a consistent time line up to the day before the present one, so that forecasts on this base can be calculated starting with the present day. The script takes a list of stations as input, for which data is to be downloaded. It additionally takes the desired date (or a time range), and the type of data to obtain, comprising real time conditions, historic data, as well a three-day forecast or a ten-day forecast. Because of the limitations of the so called developer account regarding access rates, being 500 calls per day and 10 calls per minute, a delay time in seconds between each download may be passed on to the script as input. In order to provide the functional capacity, possible network time out or server overload errors are taken into account, since they might lead to cancelled downloads. In such occasions, the script ensures that the download is repeated with respect to the given delay time until it is finished successfully. However, in some occasions the desired data cannot be retrieved, due to technical difficulties at the weather station or varying access rates on the Weather Underground server on days with high traffic load. The required values are then computed by interpolation of measurements from other stations in one of the next steps. Figure 4.9 shows the number of successful calls to the database made by the operating forecasting platform during an exemplary timespan of three weeks in February and March 2012. The fluctuations are caused by weather data of specific stations being unreachable. In the regular case, 46 automated calls are made every day to operate the forecasting platform developed in this thesis. Half of the amount is needed for obtaining the previous day's data regarding both personal and official weather stations, the latter 23 are used for stations forecasts. The complete daily data transfer volume accounts for 3.0 megabyte. The output is the requested data as JSON-formatted ASCII-file for each station, date and feature. The source code is found in the appendix subsection A.1.9.

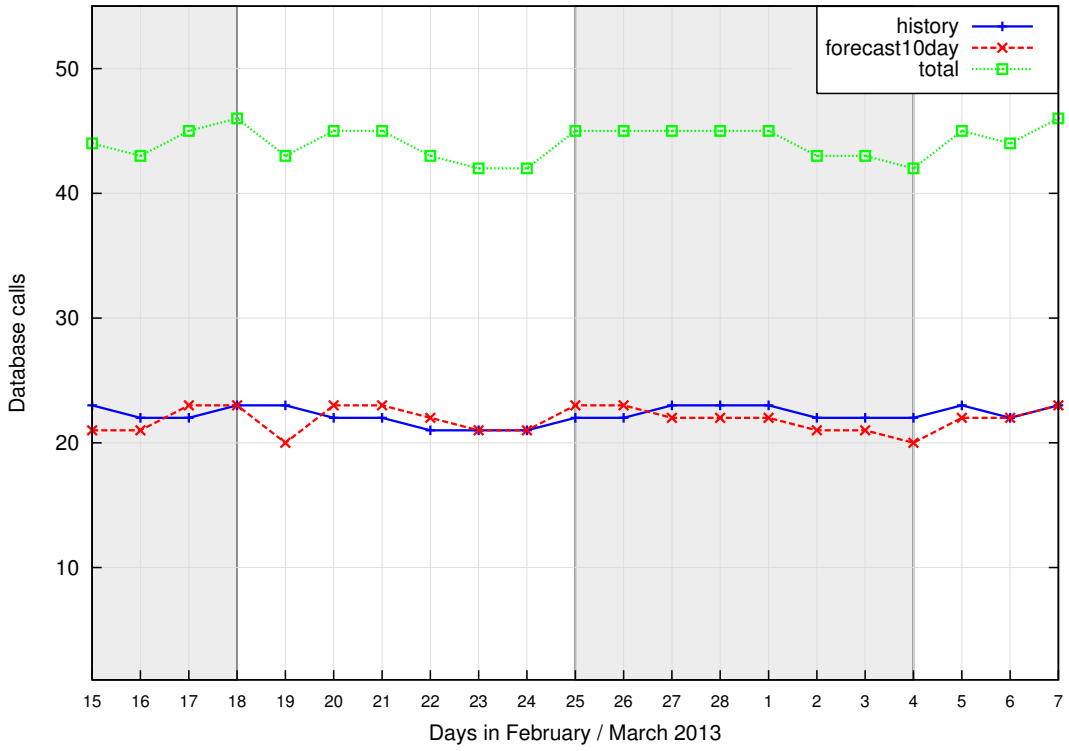


Figure 4.9: Weather Underground database calls

The figure shows the availability of Weather Underground weather station measurements, in a timespan of three weeks. The red line represents the daily number of successful database calls regarding forecasting data, the blue one shows the number for the previous day's data. The green line sums up the values of the two former ones.

- **wunderground_api_extract_history.py**
  The purpose of this script is to extract the relevant data from the previously downloaded weather datasets, to transform it as needed and to save it into a useful structure for further processing. The script takes as input a list of station names, which is extended internally with latitude, longitude and height information of the stations, and the desired day of extraction. Gap filled Shuttle Radar Topography Mission (SRTM, version 4.1) data is used for height information. The attributes of interest are average temperature and total 24-hour precipitation, as well as noon values for temperature, humidity, and wind speed. Although the results of FWI reflect the fire danger situation at 1600 hours local time, the basis for the calcu-

lation must be measurements taken at noon. The subsumed values, that is average temperature and 24-hour precipitation, are readily available in the 'dailysummary'-node of the JSON format, and can be obtained just by parsing the input data. While main attributes mentioned above are included as a rule, the station's data files differ in additional attributes listed, in temporal intervals of measurements and in points of time of measurements. For determining the noon values of temperature, humidity and wind speed, it is necessary to iterate through the list of frequent measurements and to interpolate between the nearest values before and after noon. The interpolation is done linearly. It has to be taken into account that the points of time stated in the data files are given in Universal Time Coordinated (UTC) time scale, meaning that local time is shifted one hour compared to Greenwich time for Sardinia, and two hours for Crete. The encircling values are therefore taken as near as possible to 13:00 and 14:00 UTC time on Sardinia and Crete, respectively. As the next step, meteorological values are interpolated between stations, so that stations malfunctioning or exhibiting missing data are assigned values calculated on the basis of stations working correctly. Since air temperature depends on height, the script recalculates the values for mean temperature and noon temperature to sea level to harmonise temperature values of different stations. The lapse rate is set to be 6.49 K / 1,000 m. A.1.10 shows a listing of the source code for this script.

- **wunderground_api_extract_forecast10day.py**
  This script for extracting forecasting information works similar to the one for historic data, with the exception that the data is extracted for ten days (the current date and nine days following) instead of one. The qualitative content of the forecasting data is lower compared to the actual measurements of the historical data, in that only average and extreme values are available except for precipitation data, which is available as a cumulative forecasting value of the whole day. To make sure that the fire danger platform illustrates the approximate worst case scenario, the maximal values of temperature and wind are used, as well as the minimal value of humidity. Missing data is replaced by a no-data value and interpolated within one of the following scripts. The respective source code is found in A.1.11.

- **wunderground_comb.py**
  The script is utilised to recombine the relevant meteorological information from the set of stations into one single file for a specific day. It lists all the input stations with name, coordinates, height, date and the five meteorological values extracted and partially harmonised. Its required inputs are the list of stations which are to be registered in the output file, and a date or range of dates. The output is an ASCII-formatted text file, listing stations and values as described above. Wind measurements of Personal Weather Stations (PWS) have been found to be unreliable, in that the values in the majority of the cases are 0.0, even though the Official Weather Stations (OWS) did record wind speeds. Therefore, the script is designed to set PWS wind speed values to the no-data value, the required information being assigned on base of the OWS data via interpolation. The source code is listed in A.1.12.

- **wunderground_corr.py**
  This script is intended to secure the validity of the data. Some stations transmit a value of '0.0' or '0' instead of a no-data value if a measurement could not be performed. As it is impossible to determine whether these values represent real meteorological conditions, for example a rain-free day, or whether they indicate that there was no data available, 0.0 or 0 values are generally replaced by a no-data value. These are then interpolated in the next step. If however, all values for a specific weather condition show 0.0, for instance regarding precipitation, then the likelihood of the day of interest actually having been rain-free is very high. In this case, the '0.0' value is kept. The script reads a set of data files produced by 'wunderground_comb.py' of a given timespan and returns ASCII files listing the stations with their checked measurement values. The code is found in A.1.13.

- **wunderground_idw.py**
  This script is responsible for the interpolation of meteorological values. The reason for not directly calculating the Fire Weather Index and to interpolate the danger rating results, is the varying heights of the stations (ranging from

2.7 m to 1029.0 m), which are not considered for in the FWI computation. Therefore, the meteorological attributes are interpolated consecutively over the area of interest, with station temperature values being calculated to sea level. One ASCII grid file for each meteorological parameter is produced (mean temperature, 24h precipitation, temperature at noon, humidity at noon, and wind speed at noon), and the values belonging to the specific stations are gained by querying the output maps for the given stations location. The information regarding the single stations are then collected in a separate file for the specific day, the temperature are calculated back to the actual stations heights. In this step, no-data values have been replaced by IDW interpolation results.

The script requires as input the timespan to be processed, a geo-registered grid file of the study site as base for the spatial interpolation and the projection parameters for the coordinates of the output file. The spatial resolution used for the interpolation depends on the one of the input file, a setting of 3,000 m both horizontally and vertically has been found convenient regarding outputted information density and acceptable calculation speed. Concerning the projection parameters, the calculations for Sardinia are generally made on the Universal Transversal Mercator coordinate system (UTM, with zone 32 north) on the World Geodetic System 1984 (WGS84) ellipsoid. For Crete, UTM with zone 35 North is used.

The interpolation is done by means of Inverse Distance Weighting (IDW) using the Open Source software `R` via a script in the `R` language (see A.2.1 for source code), which is internally executed from this `Python` script. The respective source code is listed in A.1.14.

- **wunderground_strt.py**
  Due to the fact that the fuel codes FFMC, DMC, and DC are based on a cumulative procedure, their output values of one day are required as inputs for the following day. This necessarily implies that a starting day with initial conditions has to be set. Two methods are described by the Canadian Wildland Fire Information System (NRC, 2013), following Turner and Lawson (1978). Method I is purposed for stations that experience

significant snow cover throughout the year. The starting day is the day when the station has been snow-free for three consecutive days. Since the study and the testing sites in this thesis do not undergo considerable snow coverage, method II is the one worthwhile, stating that three consecutive days of daily mean temperature of 6 °C or higher have to take place to mark the start date. This temperature indicates the approximate lower limit related to plant growth (NRC, 2013). If these conditions are met, the initial values are as follows:

  - **FFMC:** 85
  - **DMC:** two times the number of days since precipitation
  - **DC:** five times the number of days since precipitation

  The task of this script is to determine the start day by checking the dataset for three consecutive days with an average temperature of 6 °C, as required, for every station used. To render the determination more precise, it is set as a condition that there must have been one day of rain before the consecutive rain-free ones, to exclude the possible source of error deriving from the starting of the check within a dry period. The script takes as input the timespan of data to be checked. The calculated starting days are stored in an ASCII file. The code is listed in A.1.15.

- **wunderground_splt.py**
  The script screens the ASCII file containing the starting date for every station and extracts the latest one in order to use them as initial point for the FWI calculations. The start-up values are set as proposed by Turner and Lawson (1978; NRC, 2013): However, the script mainly serves formatting tasks, reorganising the meteorological data from day-based structure, where the relevant weather values of different stations are listed in a file for each specific day, to a station-based structure, registering the daily values for a specific station. The daily files are parsed for that purpose, and the data belonging to the same weather station is extracted and saved to a temporary file. The only required input is the location of the file with starting days. The actual appending of the daily weather values from the

temporal file to the station's total file is done by the simple script 'wunderground_splt_append.py' (A.1.16), and in doing so, collecting the data for each day in an archive file for every specific weather station. The archive files include all historic weather data used for the creation of this thesis, and date back to January 1st 2001. They serve as a base for the FWI forecasts, in that these files are copied and extended with previously extracted forecasting weather data. The source code is found in A.1.17.

- **pyFWI**
  The reorganised data structure equals the desired input format required by `pyFWI`, an Open Source `Python` script for FWI calculation that was found during research for a planned script serving the same purpose in the frame of this thesis. It reads an ASCII file listing latitude, FFMC, DMC, and DC initial values, as well as date, noon temperature, noon relative humidity, noon wind speed, and 24h-precipitation. The FWI together with its subindices is calculated from the meteorological information. The output files (one for each station) contain the original information extended by the computed values of FFMC, DMC, DC, ISI, BUI, and FWI. As stated before, FFMC, DMC, and DC outputs depend on the previous days results, so it is necessary to have the complete historic data available in the archive file for computing up-to-date values.

- **wunderground_fwicomb.py**
  Before the start of the actual processing, the script 'wunderground_fwicomb_cut.py' (A.1.19) is used to extract the last line out of each stations archive file, which represents the collection of newly computed fire danger index values, and to save the information in a temporary ASCII file. The main script then is utilised to regroup the data to a listing of stations with corresponding FWI values for the given day. Its input values are a list of stations to be taken as data sources, and the date (or a range of days) to process. A listing of a source code is located in A.1.18.

- **wunderground_fwiidw.py**
  This script represents the final step in fire danger estimation. As in the first interpolation script, `R` is used for interpolation purposes, using a script in the `R`-language (see A.2.2 for code). The `R` commands are executed

from the `Python` script. The required input parameters are, as before, a geo-registered grid file with the desired resolution and the projection parameters for the output file. But instead of meteorological values, this time the previously calculated outputs for FWI, ISI, BUI, FFMC, DMC, and DC are interpolated, and the results stored as ASCII grid files for the area of interest. The subscript 'wunderground_class.py' (A.1.21) is then used for the actual classification of index outputs into levels of fire danger by comparing the FWI results with certain threshold values. Both classification schemes, the one by EFFIS and the one developed in this thesis, which is based on a long-term study regarding Sardinia can be utilised. These raster maps containing the danger classes as pixel values are converted to vectorial data by the `gdal_polygonize`-script included in the `GDAL` software package, whereby coordinates of joining pixels with equal values are combined to polygons, with an attribute representing the pixel's value. The final results are `Geography Markup Language (GML)` vectorial files in the same coordinate system as the input files having polygons representing levels of fire danger. An exemplary result regarding July 14th 2012 is shown in figure 4.10, opposed to a screenshot of the EFFIS forecast for that day.

The maps are additionally converted to a WGS84 / Mercator projection as used by Microsoft `Bing Maps`, `GoogleMaps` and `OpenStreetMap`, so that the danger maps can be underlaid with these Web Mapping Services in the forecasting platform. This is done by using the Open Source software `GDAL`, which is called from inside the script. The code for this script is located in A.1.20.

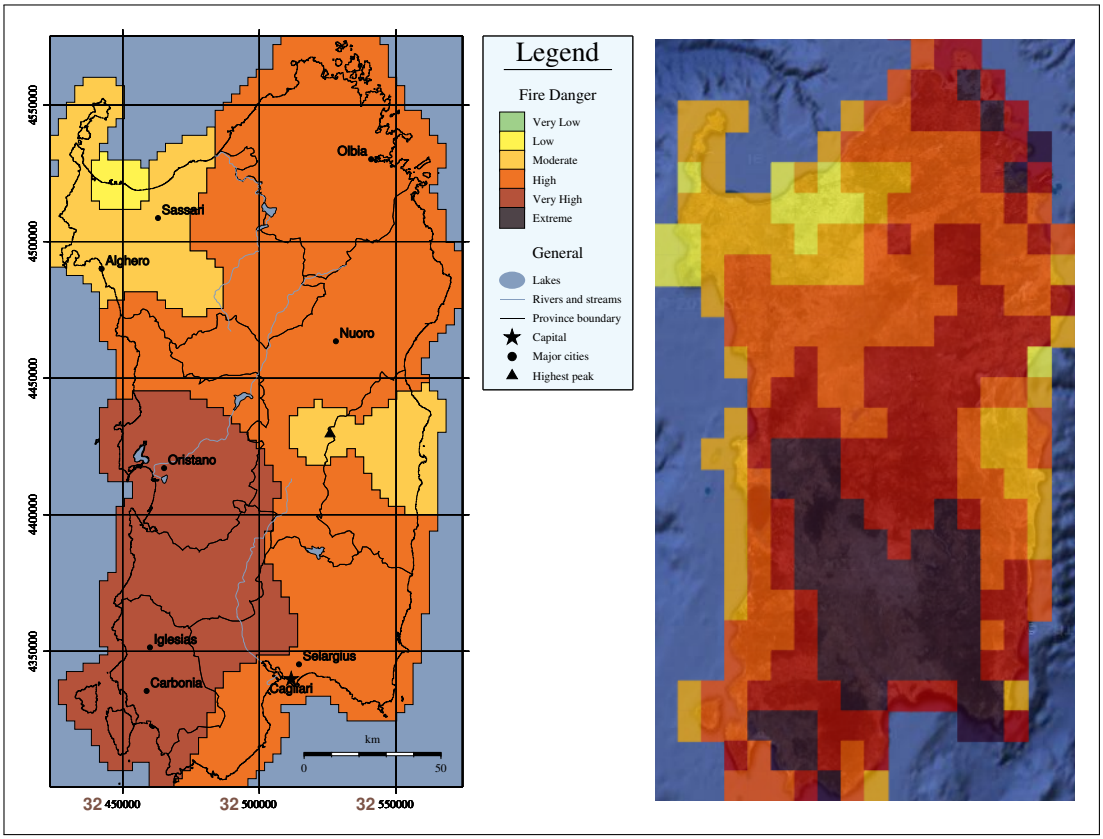


Figure 4.10: Exemplary comparison between thesis (left) and EFFIS (right) results for July 14th 2012

In both pictures, the areas of low and moderate fire danger are easily discernible in the northwestern part of Sardinia, as well as in the central eastern part. But while the thesis result shows only high and very high fire danger for the South, EFFIS marks most of this region as experiencing extreme fire danger. This is due to the more conservative classification scheme, while the EFFIS classification tends to be exaggerating. No fire occurred on that day on Sardinia.

- **wunderground_platform.py**

  The final script intends to publish the previously created forecast maps to the internet. This is done in several ways. Mainly, a set of interconnected Hyper Text Markup Language (HTML) documents is created representing the actual fire danger forecasting platform. Dynamic maps for FWI, ISI, BUI, FFMX, DMC, and DC are shown for each of the ten forecasting days. The user can zoom into and pan these created maps. Moreover, individual

severity levels can be turned visible or invisible. The `OpenLayers` software library is used for the creation of dynamic maps, transmitted to the user via `Apache HTTP`, the web server software chosen for this purpose. In addition to the main platform, the maps showing the current FWI fire danger levels are also made available through a web-based mobile application, which enables the user to also see his own position directly on the map. Figure 4.12 represents a screen shot of this web app. Finally, current FWI danger levels are available by the user as a Web Mapping Service using a desktop- or a web-based GIS client. This enables GIS workers to integrate daily fire danger forecasts into one's own GIS calculations and layouts. The Open Source package `Mapserver` is used for communicating the created content via the web. The code can be found in A.1.22.

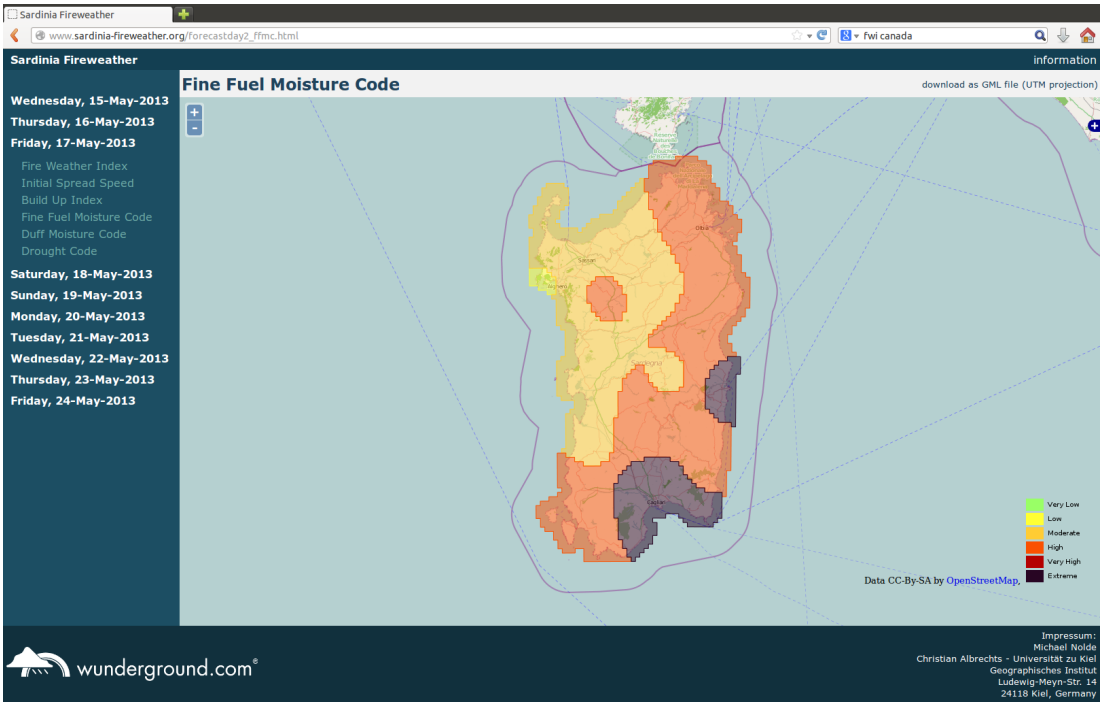


Figure 4.11: Screenshot of the platform, May 15th 2013

The visualisation shows the prediction results for the FFMC.

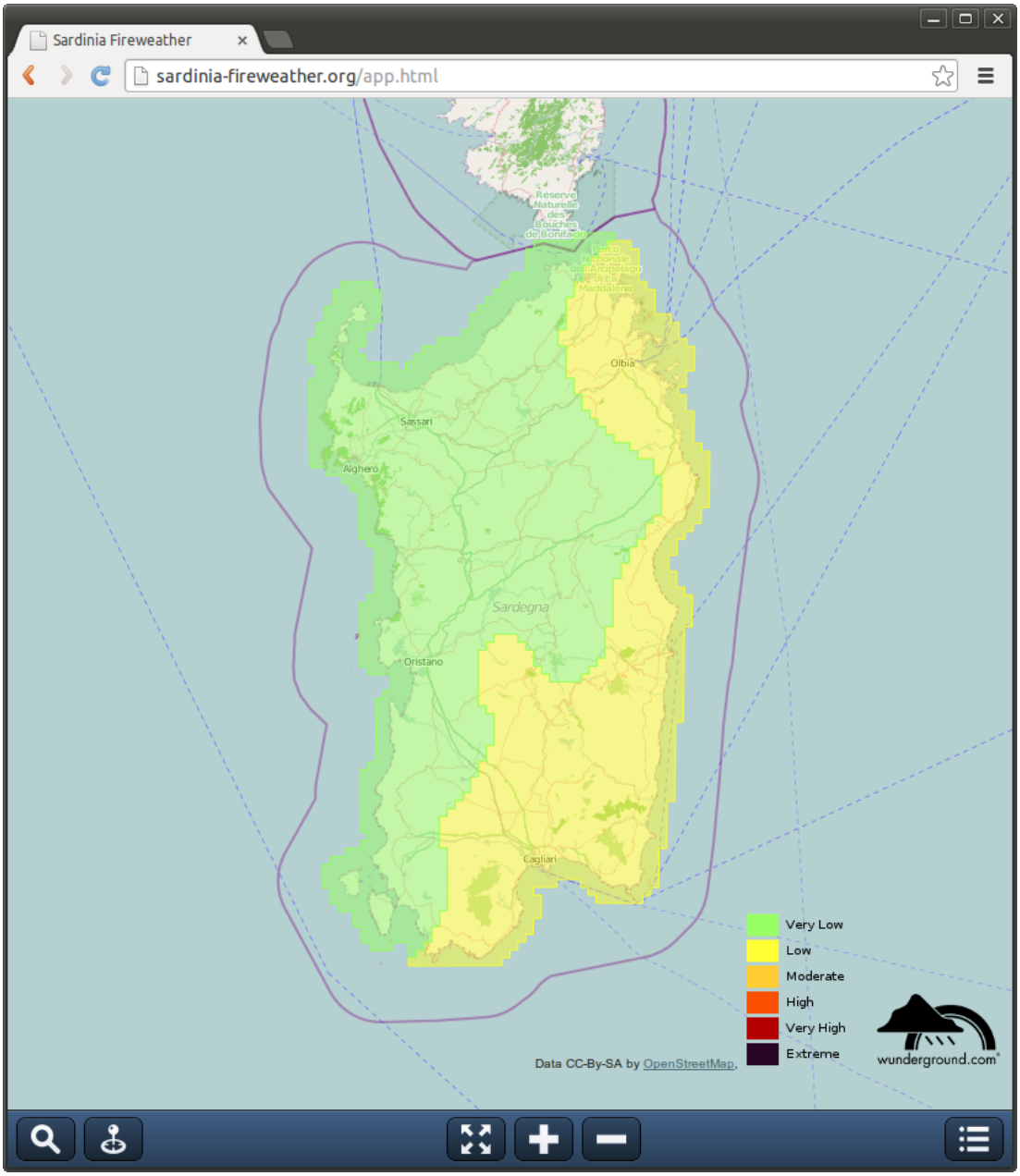


Figure 4.12: Screenshot of the mobile application, May 29th 2013

The FWI prediction is shown. The applications QR code is found at the bottom.

4.3.2 Forecasting platform

The forecasting system follows a modular design concept and is implemented as a set of `Python` scripts, each one serving a specific purpose. The scripts make heavy use of `shapely`, a `python` package for manipulation and analysis of planar geometric objects. It is based on the `Geometry Engine Open Source` (`GEOS`), the engine used in `PostGIS` for geometric objects, which is itself a port from the `Java Topology Suite` (`JTS`; Gillies, 2013b). The most frequently used functions are, firstly, point-in-polygon analysis (to check, for example, whether a hotspot is located inside a burned area) and secondly, determination of area dimensions. Some of these scripts make use of external software packages:

- `R`: for IDW interpolation (using interpolation weight 2.0)
- `GDAL/OGR`: for reading, writing, converting and reprojecting geo-registered data in grid formats or vectorial file formats
- `PROJ.4`: for projection of coordinates (geographic to projected and vice versa)
- `pyFWI`: for computing FWI indices out of weather conditions data

The scripts are executed in a specified order and with required attributes by two master scripts, realised in `Unix Bourne Shell` code. The first one controls the downloading processing of the previous day's weather data, while the second master script organises, on the one hand, the handling of forecasting weather data, and, on the other hand, the creation of actual forecasting maps as well as their publication to the web (as shown in figure 4.8). The resulting website can be seen exemplary in 4.11. These two master scripts are executed every day at 8 o'clock a.m. by a cron job within the platform. This starting time is due to the time lag between Europe and the United States, by this the required weather information is gathered as soon as it is provided by the server infrastructure of Weather Underground. See appendix sections '0200b_live_yesterday.sh' (A.3.1) and '0200c_live_forecast.sh' (A.3.2) for the source code of the two master scripts. GNU/Debian 6, a Linux distribution, is the chosen operating system, which is completely free software as is all the software used in this thesis and distributed

under the GNU Public License (GPL). The complete software system is implemented as a virtual machine, set up using the virtualisation infrastructure of the `Kernel-based virtual machine (KVM)` software package, as well as `QEMU` for the actual virtualisation and `libvirt` for management purposes. The complete virtual machine in `QEMU Copy On Write format version two (qcow2)` is available for download.
The address of the running implementation of the platform is as follows:
http://www.sardinia-fireweather.org

The fire danger forecasting map of the current day is also provided by the platform as Web Map Service (WMS). Through the WMS interface, agencies in charge can implement the forecasts directly into their GIS work flow and process the data in an automated manner.
WMS is the abbreviation of the Web Map Service Interface Standard of the Open Geospatial Consortium (OGC), purposed for managing the delivery of geo-registered grid maps from a remote geospatial data source through a Hypertext Transfer Protocol (HTTP) request/response communication. In the request, the grid layer and the desired geographic extent are defined. The response consists of one or more geo-registered images (usually in the formats JPEG or PNG), which then can be displayed by a GIS client, both web-based and in a desktop environment (OGC, 2013c). The OGC is an international consortium of companies, government agencies and universities developing free interface standards in the field of geospatial applications (OGC, 2013a).
The service is included via its Uniform Resource Identifier (URI). It is provided with geographic coordinates and in a projected coordinate system. For latitude and longitude coordinates, based on the WGS84 ellipsoid or '4326' in form of the European Petroleum Survey Group (EPSG) code, the URI is:

http://sardinia-fireweather.org/cgi-bin/mapserv?map=firedanger_sardinia.map
&VERSION=1.1.1&LAYERS=Firedanger Sardinia&SRS=EPSG:4326
&BBOX=8,38,10,42&WIDTH=800&HEIGHT=600

The WMS in projected coordinates of the Mercator coordinate system, based on the WGS84 ellipsoid (EPSG code: 3785), is as follows:

http://sardinia-fireweather.org/cgi-bin/mapserv?
map=firedanger_sardinia_900913.map&VERSION=1.1.1
&LAYERS=Firedanger Sardinia&SRS=EPSG:3785
&BBOX=-900419,4700326,1099739,5064834&WIDTH=800&HEIGHT=600

This is the coordinate system used by Google, Microsoft BingMaps and the OpenStreetMap WMS service, hence the forecasting maps can easily be enriched by services of that kind. The WMS can easily included in an SDI-enabled GIS application, for example Quantum GIS (QGIS).
Web services with geospatial reference, such as WMS or the Web Feature Service (WFS), are of major importance in Spatial Data Infrastructures (SDI), for example the one established in the frame of the European directive for Infrastructure for Spatial Information in the European Community (INSPIRE). The task of INSPIRE is to support community policies and activities affecting the environment by aggregating the infrastructures for spatial information maintained by the member states of the European Union.
On the platform, the delivery of the grid forecasting data is done by `mapserver`, an Open Source system for publishing spatial data to the web, both from local and remote sources. It requires a web server in order to work, and it is `Apache HTTP` that is used for this purpose.
In addition to the forecasting website, an alternative front-end is offered for viewing of forecasts on mobile devices. This is realised in form of a web app, which can be used on any device regardless of brand and is not dependent on vending systems such as Apples AppStore. The device's web browser is used to access the application. The Open Source mobile development framework `SenchaTouch` has been used in connection with the mapping client `OpenLayers` for the implementation of the mobile application.
The web address of the mobile application is as follows:
http://www.sardinia-fireweather.org/app.html

For convenience, it is also possible to reach the application by scanning a Quick Response (QR) code with the device's camera (see figure 4.12).

Chapter 5

Results & Discussion

This chapter centres on the results of hotspot validation data tested against fire danger forecasts, both using the newly developed classification scheme and the one used by EFFIS. The findings are represented in figures and tables, which are discussed in this chapter.

As it was found inadequate to equalise one hotspot detection with one wildfire, an attempt has been made to group single hotspots to a specific fire event if possible, depending on their spatial and temporal proximity. A setting of 5 km for maximum spatial distance, and 2 days for maximum temporal distance was taken. The processing is done via the script 'fireevents.py' (see A.1.6). Tabular listings of determined fire events are found in tables 5.3 and 5.4 for Sardinia as well as in table 5.5 for Crete. The fire events are represented by grey boxes within the type II error figures. Weather Underground data is abbreviated as 'WU' in figures and tables. The results gained via the threshold values of Dimitrakopoulos are labelled using the author's name, and the results originating from the EFFIS website are given the label 'COSMO-EU/EFFIS'. COSMO-EU is the European weather model that is used internally by EFFIS for production of fire danger maps.

The type I and type II errors are determined using the scripts 'hitratio.py' (A.1.4) and 'randompoints.py' (A.1.5).

5.1 Results

5.1.1 Long-term study 2001 - 2010

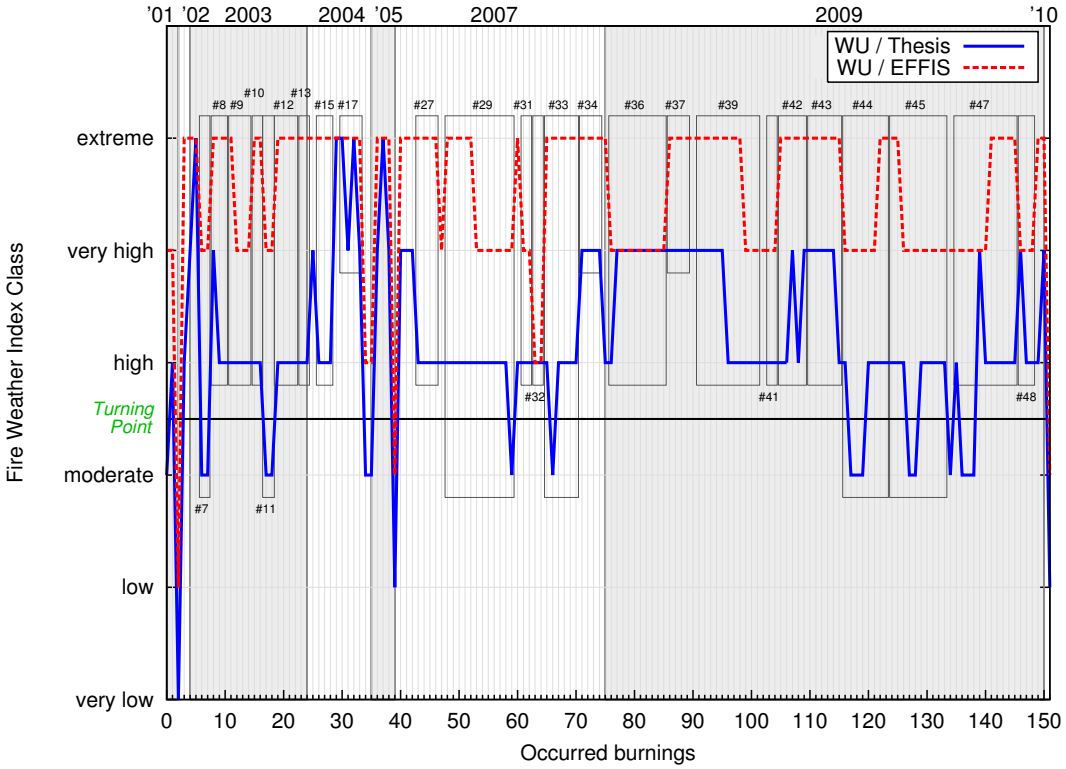


Figure 5.1: Type II error, Sardinia 2001 - 2010 (recall ratio)

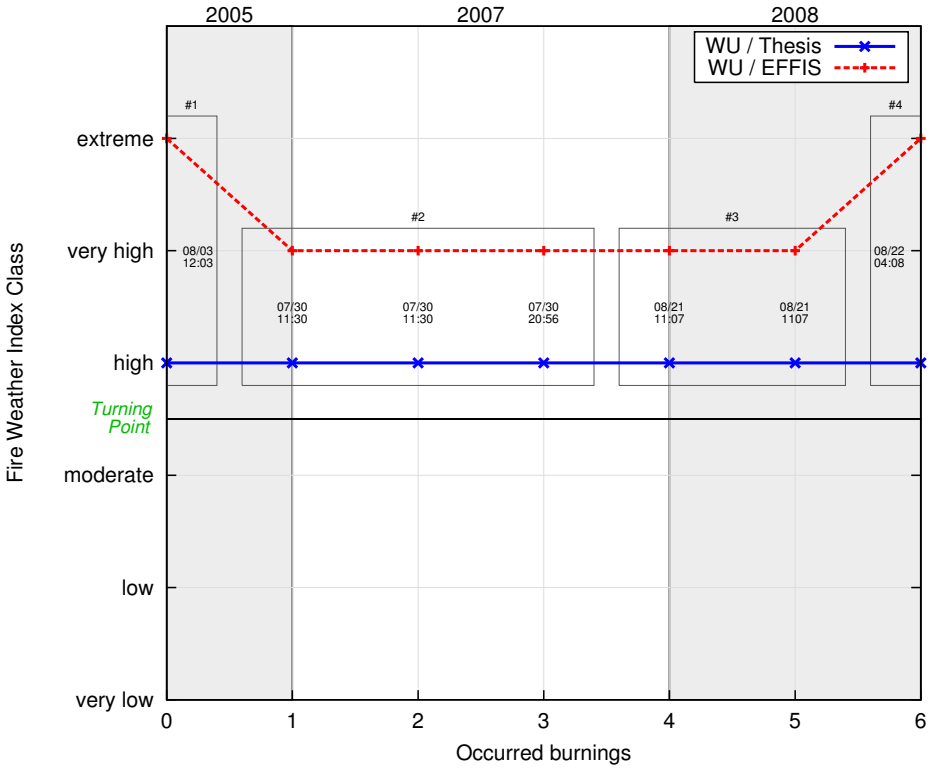


Figure 5.2: Type II error, Crete 2001 - 2010 (recall ratio)

Figures 5.1 and 5.2 indicate, for every fire, the forecasted fire danger level for its point of occurrence. The figures cover a timespan from 2001 to 2010. The date and location of the validation hotspots have been compared with the fire danger map of interpolated FWI results for the specific day.

As stated in chapter 2, the FWI is a measure of the potential intensity of a propagating flame front, but is best used as a rating for general fire danger. Since the validation hotspots are known to definitely represent actually occurred fires, the validity of FWI forecasting results can be checked, in this case done retrospectively for historic weather conditions.

The forecasts are produced by using both classification schemes, the one of the thesis and the one by EFFIS. The red line in figures 5.1 and 5.2 represents the predicted fire danger class for a specific hotspot location using the EFFIS classification, the blue one marks the thesis classification. In ideal case, all hotspots should be marked by the classification scheme as being located in high to extreme

fire danger areas. In order to keep the levels of low fire danger and the ones of high danger apart, a horizontal black line is plotted in the figures.
As can be seen on the Y-axis of the figures 5.1 and 5.2, the forecasts produced when using the thesis classification are generally more moderate compared to the ones of EFFIS, and sometimes even underestimating fire danger conditions.
The numbered grey boxes enclose hotspots which most likely belong to the same fire event (see table 5.3 for a detailed listing of these events). Weather Underground (WU) data has been used as weather input source in all cases. See table 5.1 on page 127 for a listing of result given in percentages.

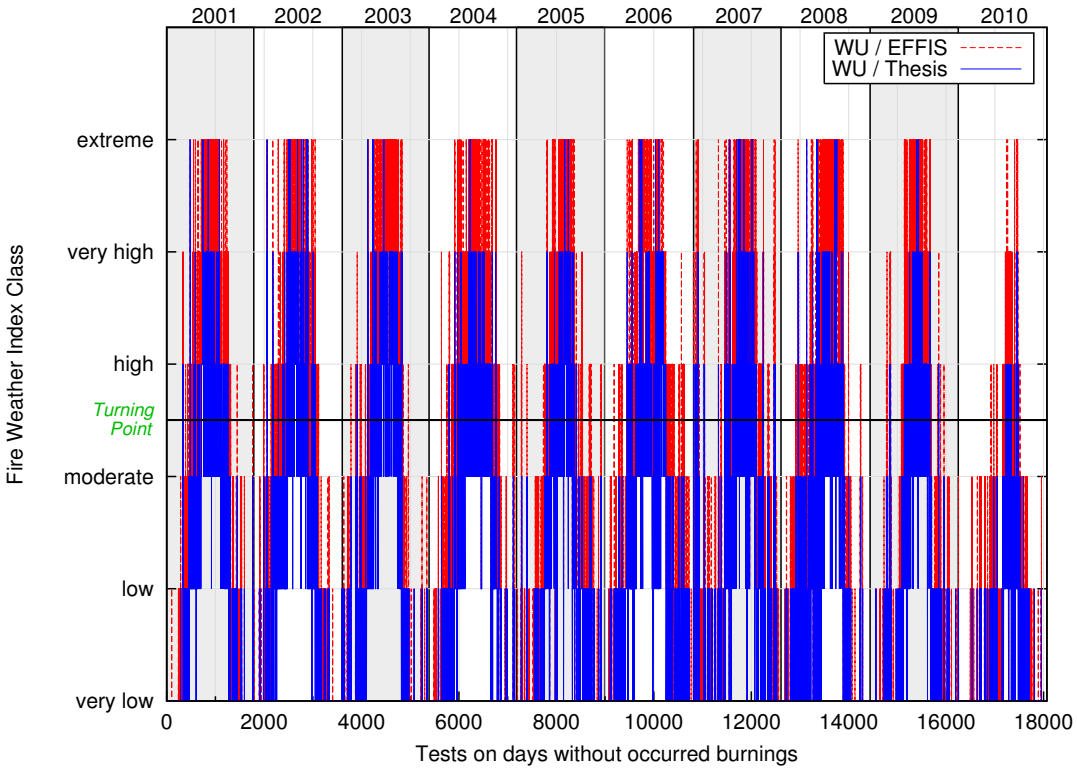


Figure 5.3: Type I error, Sardinia 2001 - 2010 (false alarms)

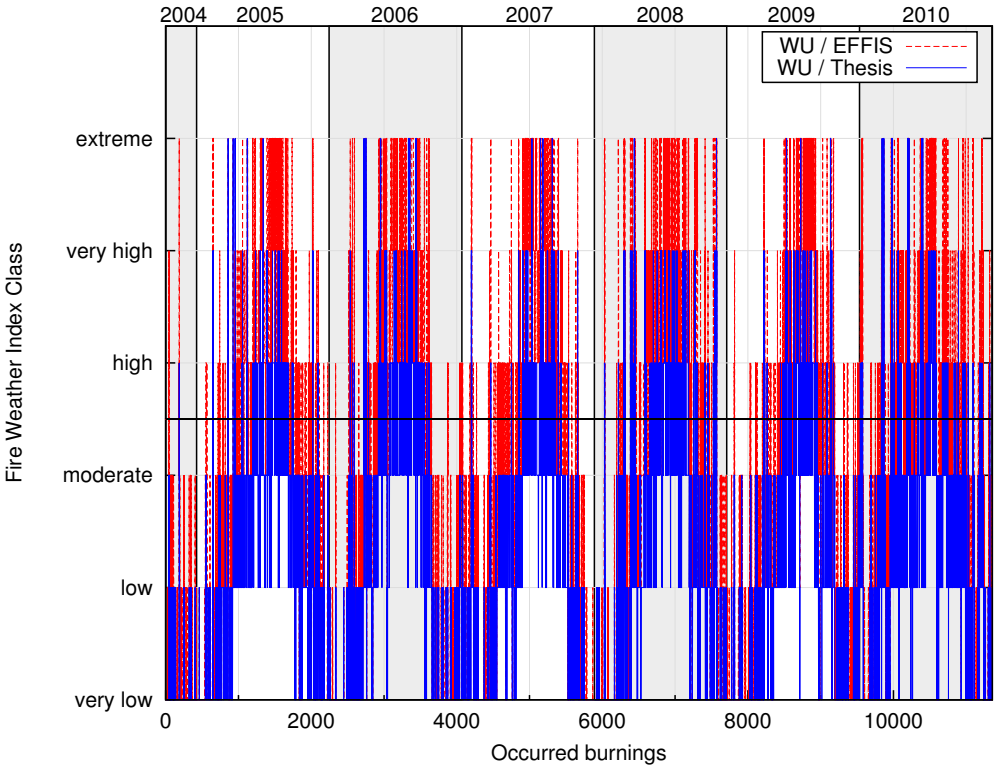


Figure 5.4: Type I error, Crete 2001 - 2010 (false alarms)

Figures 5.3 and 5.4 show the kind of fire danger level that was forecasted for a set of testing point locations on days when no fire occurred in the region of interest between 2001 and 2010. In ideal case, all areas would be marked by both classifications as exhibiting low fire danger since no fire actually occurred. The red line represents the forecasted fire danger class for a specific hotspot location using the EFFIS classification, the blue one stands for the thesis classification. For distinction purposes between the levels of low and the ones of high fire danger, a horizontal black line is plotted in the figure. As can be seen, the EFFIS classification tends to produce more 'Extreme' fire weather predictions on days without fire occurrence. The wave movement is due to differences in predicted fire danger during summer and winter, respectively. 18,070 testing points for Sardinia and 11,360 points for Crete have been randomly generated for this test. Weather Underground data is used as input. See table 5.2 for a listing of result percentages.

Table 5.1: Results of the type II error test (recall ratio) for 2001 - 2010

	Hotspots	**Very Low (%)**	**Low (%)**	**Moderate (%)**	**High (%)**	**Very High (%)**	**Extreme (%)**
Sardinia							
WU/Thesis	152	0.66	1.32	11.84	54.61	28.29	3.29
WU/EFFIS	152	0.00	0.66	1.32	2.63	38.82	56.58
Crete							
WU/Thesis	7	0.00	0.00	0.00	100.00	0.00	0.00
WU/EFFIS	7	0.00	0.00	0.00	0.00	71.43	28.57

Table 5.1 shows the percentages of hotspots that occurred in a region of a specific forecasted fire danger level, for both the thesis and the EFFIS classification. Since these represent fires that actually occurred, in ideal case all hotspots would have occurred in areas marked as 'High', 'Very High', or 'Extreme' fire danger areas. This is the case for 86.18% using the thesis classification, and 98.03% in the case of EFFIS regarding Sardinia. Since there are only seven testing points available for Crete regarding the timespan from 2001 to 2010, the respective results cannot be regarded as significant.

Table 5.2: Results of the type I error test (false alarms) for 2001 - 2010

	Hot-spots	**Very Low (%)**	**Low (%)**	**Mode-rate (%)**	**High (%)**	**Very High (%)**	**Ex-treme (%)**
Sardinia							
WU/Thesis	18070	49.55	19.21	16.43	11.29	3.09	0.43
WU/EFFIS	18070	39.80	13.31	12.67	14.36	10.82	9.04
Crete							
WU/Thesis	11360	33.49	29.44	21.94	12.45	2.12	0.56
WU/EFFIS	11360	22.45	16.18	18.63	21.65	12.45	8.64

Table 5.2 shows the percentage of testing points situated in a region of a specific forecasted fire danger level, on days when no fires actually occurred in the study region. As already mentioned, 18,070 testing points for Sardinia have randomly been generated, of which 49.73% have been located in areas with combustible vegetation. Analogous, 11,360 points have been generated for the Isle of Crete, with 46.58% being located in vegetated areas.
Ideally, all testing points would be situated in regions marked with low predicted fire danger. In case of the thesis classification, this is true in 85.19% of tests for Sardinia, and in 65.78% regarding the EFFIS scheme. For Crete, this false positive test results in 84.87% correct results using the thesis, and in 57.25% using the EFFIS classification. Regarding these test results, the thesis classification scheme is superior to the one of EFFIS, since its produced forecasts represent a closer simulation of real conditions.

Table 5.3: Fire events on Sardinia, 2001 - 2010, part I

Year	No.	Province	Date	Land cover	Burned area (km^2)
2001	1	Olbia-Tempio	08/01	Sclerophyllous veg.	0.18
	2	Cagliari	08/08	Sclerophyllous veg.	0.18
	3	Oristano	10/03	Grasslands	2.18
2002	4	Nuoro	06/29	Grasslands	3.64
	5	Carbonia-Igl.	07/05	Broad-leaved forest	4.26
2003	6	Oristano	06/23	Broad-leaved forest	3.11
	7	Medio-Camp.	07/07	Sclerophyllous veg.	1.84
	8	Oristano	07/22-23	Grasslands	1.27
	9	Nuoro	07/23-24	Grasslands, forest	7.28
	10	Nuoro	08/03	Grasslands	0.91
	11	Olbia-Tempio	08/04	Sclerophyllous veg.	1.44
	12	Olbia-Tempio	08/14	Grasslands, forest	6.88
	13	Olbia-Tempio	08/15	Sclerophyllous veg.	0.54
2004	14	Medio-Camp.	07/09	Sclerophyllous veg.	6.27
	15	Oristano	07/17-19	Grasslands	0.18
	16	Ogliastra	08/21	Sclerophyllous veg.	1.47
	17	Olbia-Tempio	08/26	Sclerophyllous veg.	0.54
	18	Oristano	09/08	Grasslands	0.18
	19	Nuoro	09/21	Broad-leaved forest	0.55
2005	20	Oristano	07/16	Sclerophyllous veg.	1.83
	21	Oristano	08/15	Sclerophyllous veg.	0.18
	22	Oristano	08/15	Grasslands	0.91
	23	Oristano	08/30	Grasslands	0.18
2007	24	Carbonia-Igl.	07/16	Sclerophyllous veg.	0.19
	25	Medio-Camp.	07/16	Grasslands	4.05
	26	Carbonia-Igl.	07/17	Sclerophyllous veg.	0.74

Table 5.4: Fire events on Sardinia, 2001 - 2010, part II

Year	No.	Province	Date	Land cover	Burned area (km^2)
2007	27	Nuoro	07/23	Broad-leaved forest	81.75
	28	Nuoro	07/24	Sclerophyllous veg.	0.18
	29	Nuoro	07/23-24	Grasslands, forest	85.21
	30	Nuoro	07/23	Broad-leaved forest	81.75
	31	Nuoro	07/24	Sclerophyllous veg.	81.75
	32	Ogliastra	07/24	Sclerophyllous veg.	2.56
	33	Cagliari	07/24	Sclerophyllous veg.	4.97
	34	Sassari	08/09	Sclerophyllous veg.	2.72
	35	Cagliari	08/26	Sclerophyllous veg.	0.37
2009	36	Olbia-Tempio	07/23	Sclerophyllous veg.	21.52
	37	Nuoro/Sassari	07/23	Grasslands	55.89
	38	Oristano	07/23	Grasslands	2.73
	39	Oristano	07/23-24	Coniferous forest	11.74
	40	Sassari	07/24	Grasslands	55.71
	41	Sassari/Nuoro	07/24	Grasslands	62.99
	42	Sassari	07/23	Grasslands	73.73
	43	Sassari	07/23	Grasslands	44.98
	44	Nuoro	07/23-24	Sclerophyllous veg.	7.42
	45	Sassari	07/23-24	Grasslands	53.32
	46	Sassari	07/24	Grasslands	55.71
	47	Cagliari	07/24-25	Grasslands	9.21
	48	Sassari	07/25	Grasslands	8.73
	49	Cagliari	07/25	Sclerophyllous veg.	0.74
	50	Cagliari	07/25	Sclerophyllous veg.	1.66
2010	51	Nuoro	09/17	Grasslands	1.46

Table 5.5: Fire events on Crete, 2001 - 2010

Year	No.	Province	Date	Land cover	Burned area (km^2)
2003	1	Heraklion	07/17	Sclerophyllous veg.	1.37
	2	Rethymnon	07/23	Grasslands	1.76
2005	3	Heraklion	08/03	Sclerophyllous veg.	4.50
2007	4	Chania	07/30	Sclerophyllous veg.	1.76
2008	5	Rethymnon	08/21-22	Sclerophyllous veg.	6.64

Tables 5.3, 5.4, and 5.5 list fire events on Sardinia and Crete, respectively, for the timespan from 2001 to 2010. Hotspots have been analysed in terms of spatial and temporal distance from each other to build clusters of probably related fire occurrences. By this approach, it is possible to gain information on the actual extend of a wildfire from pure hotspot data. It can be seen that the largest burnings on Sardinia occurred in the province of Nuoro in 2007.

5.1.2 Validation in summer 2012

Although the forecasting system has been validated using historic data for the timespan of 2001 to 2010, it still needs to be tested in the field. It was therefore brought into operation during the summer of 2012, and the results were compared with the wildfire danger warning maps of EFFIS on a daily basis. Several processing steps have been carried out to make the two testing systems comparable, the sequence of which is shown in figure 5.5 (page 133).

- **(a)** Daily screenshots were taken of the EFFIS forecasting website for each index, both for Sardinia and Crete. Beforehand, the best view settings were determined which offer complete view of the respective island while at the same time providing the best possible pixel resolution. The settings were stored by using the 'permalink'-function of the map viewer, which

enables the user to view the fire danger forecast at exactly the same map location every day. It was of upmost importance that every captured image covered exactly the same spatial extent in order to make automated analysis possible. Screenshots were taken from the beginning of June till the end of September 2012. However, EFFIS changed the forecasting range from four to six indices (by adding DMC and DC) in July, so it was decided to start the validation period on that day. Due to the fact that the EFFIS website was intruded into by hackers in September and could not be accessed for two weeks, the validation period covers merely the timespan of July and August 2012.

- **(b)** The screenshots were clipped afterwards, using the command line image manipulation software `ImageMagick`.

- **(c)** A mask was used to exclude sea pixels, the remaining colours were maximally saturated.

- **(d)** The resulting image is analysed by the `Python` script 'effisextr.py' (see A.1.7) where the pixel values are categorised into fire danger classes using thresholds for values of red, green, and blue. A Tagged Image File Format (TIFF) grid file with fire danger classes as pixel values is returned (class values have been replaced by actual colours in the example figure).

- **(e)** In the final step, the TIFF is expanded by a world file, which is a simple ASCII file listing position on the surface of the globe and resolution of the grid file it refers to, and by this geo-registering the TIFF file. It is now possible to vectorise the grid file using `GDAL/OGR`, thus gaining a vector map of predicted fire danger areas from the EFFIS website. This map can now be queried with a point location to execute a point-in-polygon analysis, as it is done with the maps computed by the platform. The results are represented by the entries labelled 'COSMO-EU/EFFIS' in the following figures.

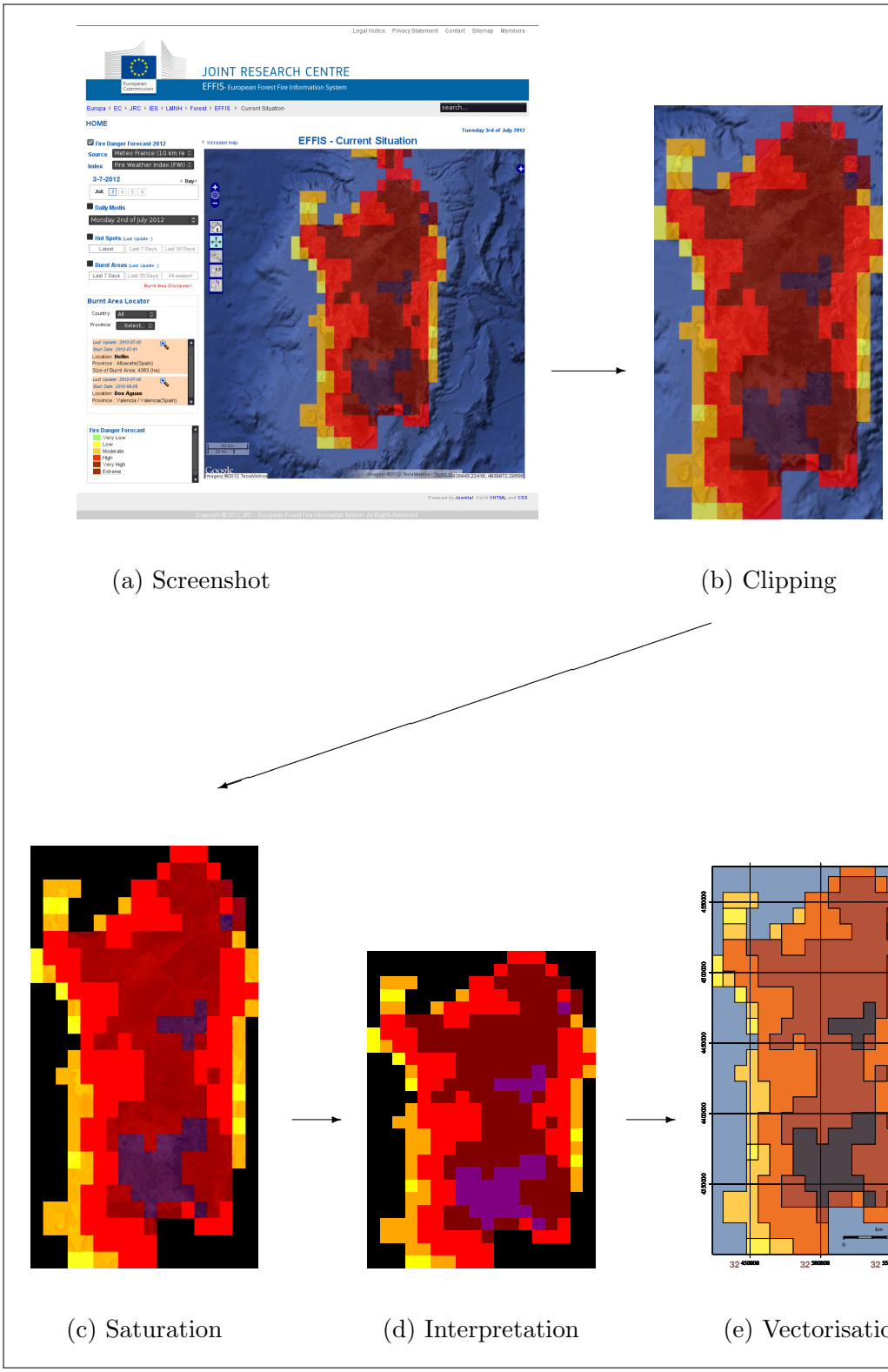


Figure 5.5: Process of data extraction from the EFFIS website

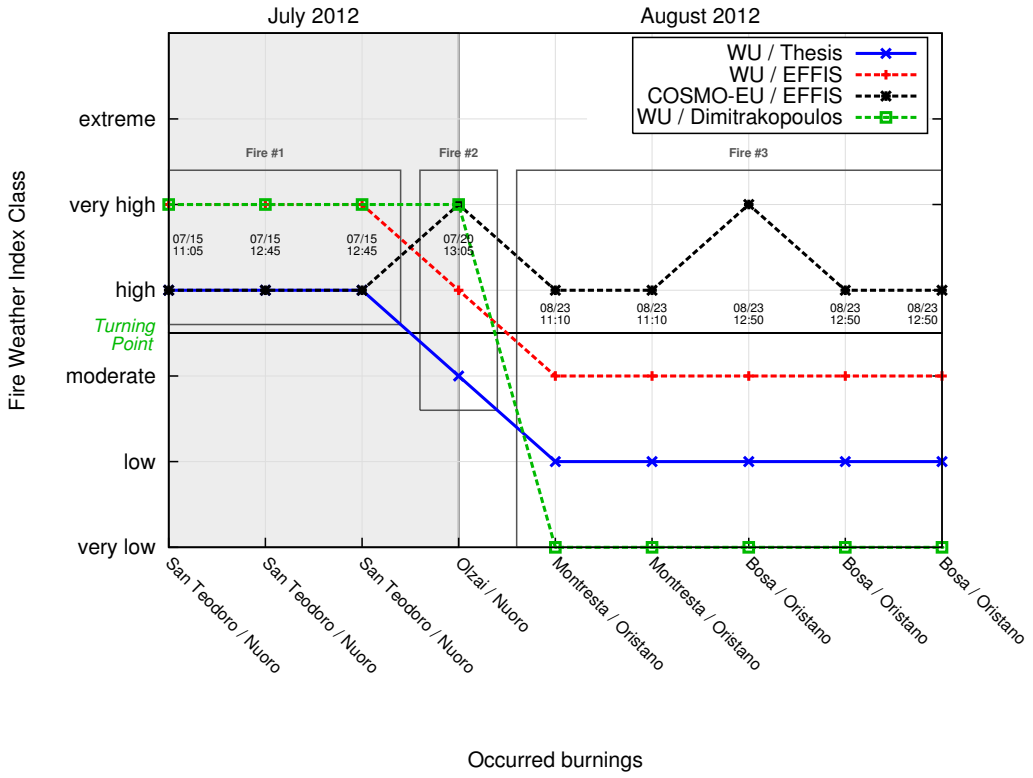


Figure 5.6: Type II error, Sardinia, summer 2012 (recall ratio)

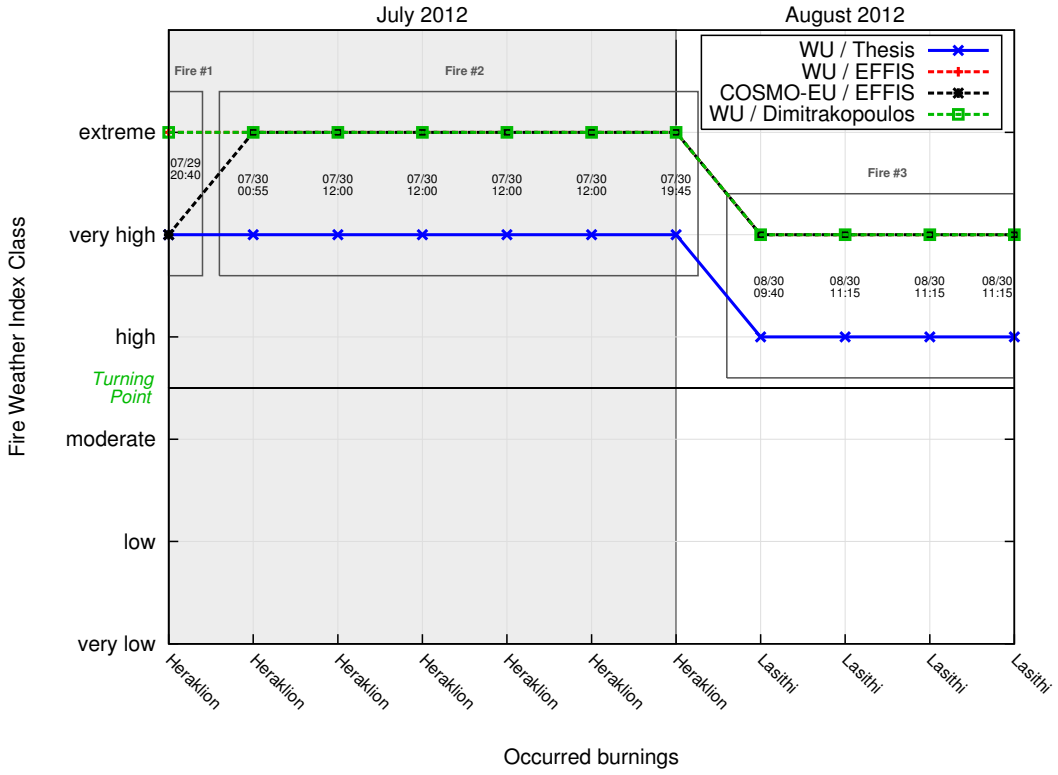


Figure 5.7: Type II error, Crete, summer 2012 (recall ratio)

Analogue to figures 5.1 and 5.2, figures 5.6 and 5.7 present the danger classes of areas in which each of the testing hotspots have been found to be located. Nine hotspots were available for Sardinia and 11 for Crete. The red line represents the testing results when using the EFFIS classification scheme, the blue one shows the results for the thesis' scheme. The results acquired with the classification by Dimitrakopoulos et al., 2011 are represented by the green line. In all three cases, Weather Underground data has been utilised as a data source. The EFFIS classification threshold values have been generously provided by JRC. By utilising a consistent source it is possible to directly compare the schemes, excluding the influence of varying underlying data. The black line shows results of the EFFIS classification scheme with the official data source of the COSMO-EU model. It represents the official forecasts available on the EFFIS website in the time of summer 2012. The grey boxes indicate fire events. See table 5.6 for a listing of result percentages.

Table 5.6: Results of the type II error test (recall ratio) for summer 2012

	Hot-spots	**Very Low (%)**	**Low (%)**	**Mode-rate (%)**	**High (%)**	**Very High (%)**	**Ex-treme (%)**
Sardinia							
WU/Thesis	9	0.00	55.56	11.11	33.33	0.00	0.00
WU/EFFIS	9	0.00	0.00	55.56	11.11	33.33	0.00
COSMO-EU/EFFIS	9	0.00	0.00	0.00	77.78	22.22	0.00
WU/Dimitrakopoulos	9	55.56	0.00	0.00	0.00	44.44	0.00
Crete							
WU/Thesis	11	0.00	0.00	0.00	36.36	63.64	0.00
WU/EFFIS	11	0.00	0.00	0.00	0.00	36.36	63.64
COSMO-EU/EFFIS	11	0.00	0.00	0.00	0.00	45.45	54.55
WU/Dimitrakopoulos	11	0.00	0.00	0.00	0.00	36.36	63.64

In analogy to table 5.1, table 5.6 shows the percentages of hotspots that occurred in a region of a specific forecasted fire danger level. It lists both the results for the thesis and the EFFIS classification. On Sardinia, all areas featuring hotspot locations have been correctly marked as high fire danger areas using the EFFIS classification. However, with the thesis classification only 66.7% of the hotspots have been situated in areas correctly marked.

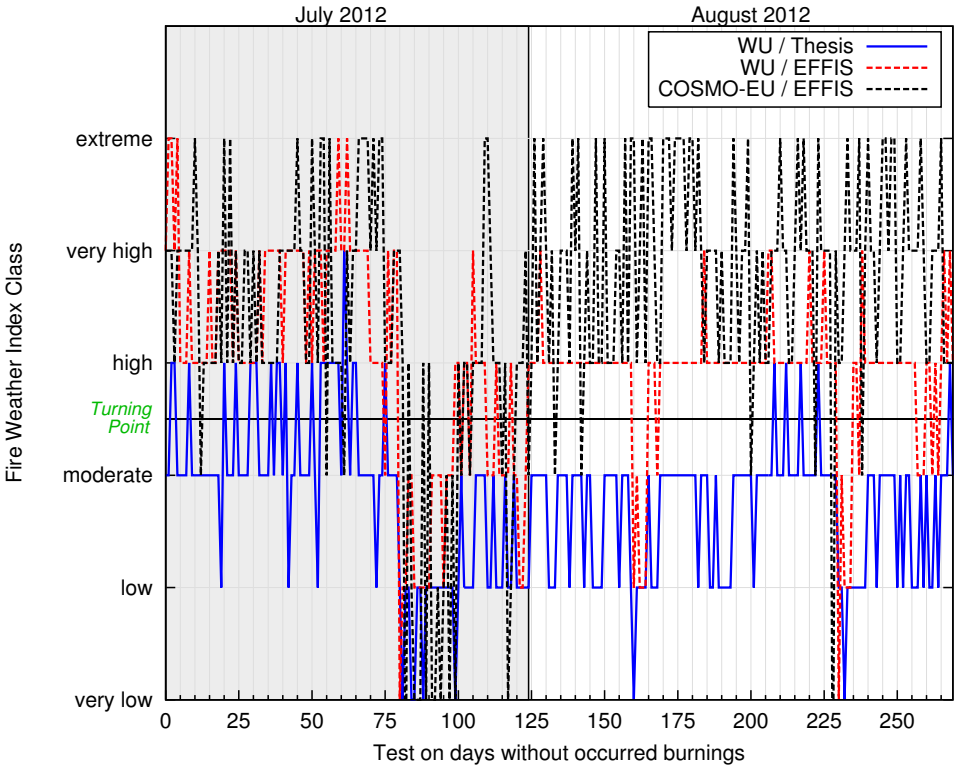


Figure 5.8: Type I error, Sardinia, summer 2012 (false alarms)

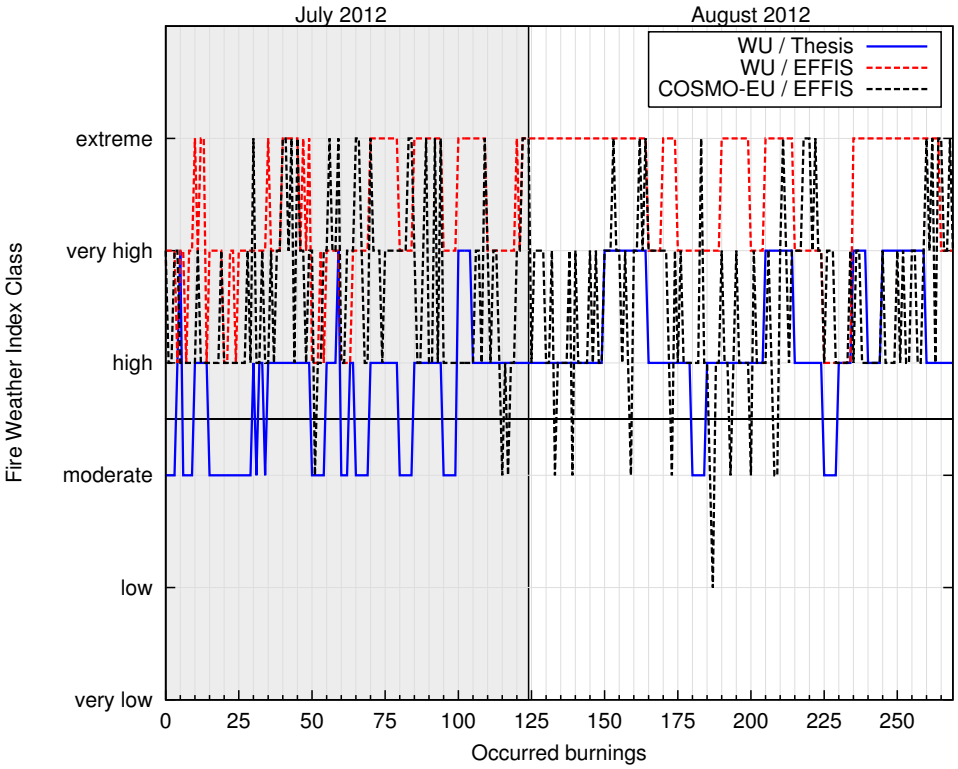


Figure 5.9: Type I error, Crete, summer 2012 (false alarms)

As did figures 5.3 and 5.4, figures 5.8 and 5.9 show the kind of fire danger level that was forecasted for a set of testing point locations on days when no fire occurred in the region of interest. The timespan covers July and August 2012. The figures show the results using the thesis' and the EFFIS classification in combination with Weather Underground data (drawn in blue and red, respectively) as well as the ones using the EFFIS scheme in connection with official weather data from the COSMO-EU model. The latter represents the official forecasts having been available on the EFFIS website during the summer of 2012. 270 randomly generated locations have been used to test the classifications on days without occurrence of a fire. See table 5.7 for percentage values.

Table 5.7: Results of the type I error test (false alarms) for summer 2012

	Hot-spots	**Very Low (%)**	**Low (%)**	**Mode-rate (%)**	**High (%)**	**Very High (%)**	**Ex-treme (%)**
Sardinia							
WU/Thesis	270	2.59	30.37	55.56	11.11	0.37	0.00
WU/EFFIS	270	0.74	6.67	11.48	59.26	20.00	1.85
COSMO-EU/EFFIS	270	3.70	1.48	5.56	33.33	34.81	20.37
Crete							
WU/Thesis	270	0.00	0.00	21.48	59.26	19.30	0.00
WU/EFFIS	270	0.00	0.00	0.00	8.89	38.89	52.22
COSMO-EU/EFFIS	270	0.00	0.37	4.44	46.30	36.67	0.00

Analogue to table 5.2, table 5.7 shows the percentage of testing points situated in an area of a specific predicted fire danger level. The test are conducted at dates when no fires occurred in the study region, covering the timespan of July and August 2012. Regarding Sardinia, 88.52% of the areas being tested with random testing point locations have been marked correctly by the thesis classification.

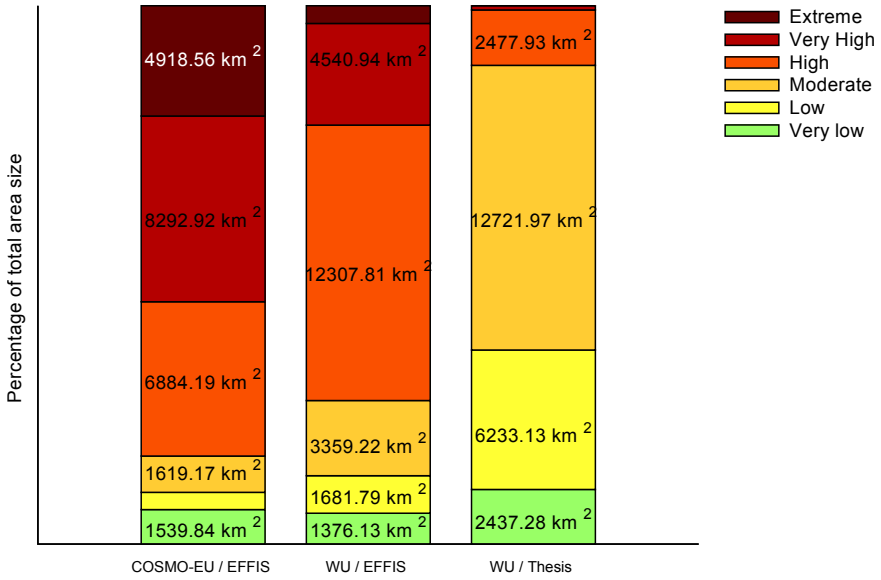


Figure 5.10: Comparison of average area size using different classification schemes

The results for FWI regarding Sardinia, summer 2012, are shown.

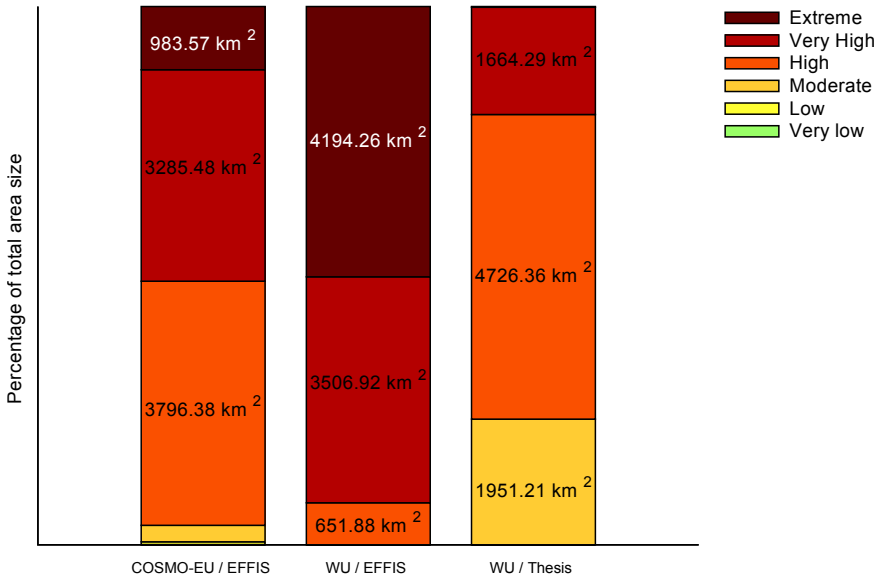


Figure 5.11: Comparison of average area size using different classification schemes.

The FWI-results regarding Crete, summer 2012, can be seen here.

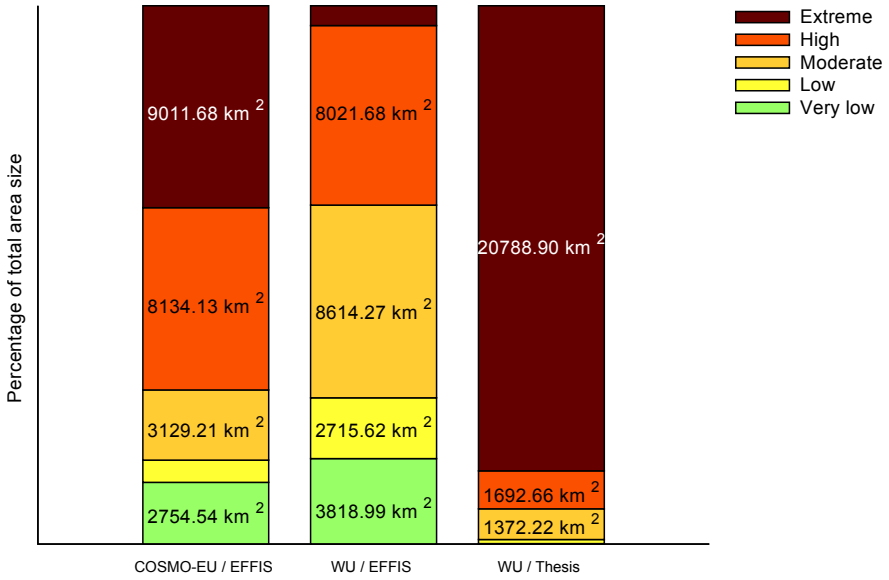


Figure 5.12: Comparison of average area size using different classification schemes. The results for FFMC regarding Sardinia, summer 2012, are shown.

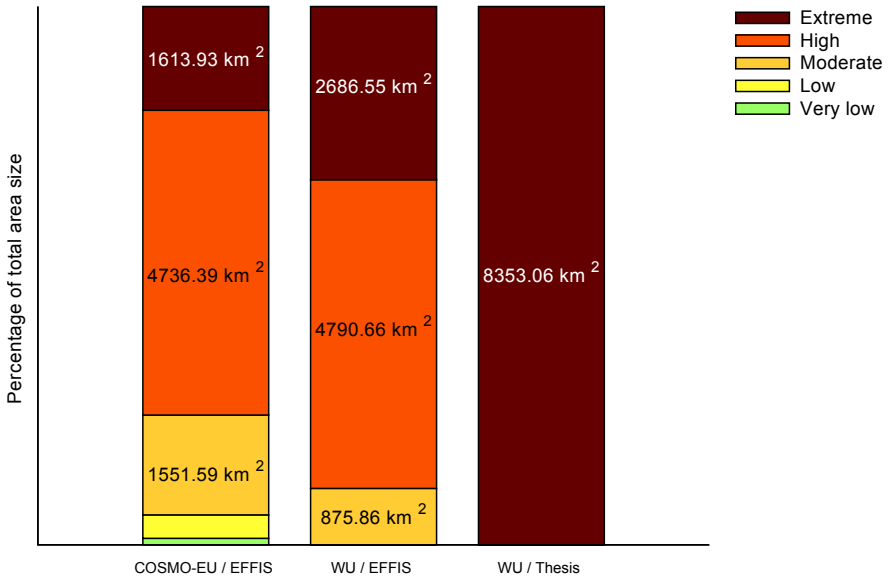


Figure 5.13: Comparison of average area size using different classification schemes. The FFMC-results regarding Crete, summer 2012, can be seen here.

The above figures represent a comparison of summed up average sizes of areas marked as exhibiting a specific fire danger level, both for Sardinia (figure 5.10) and Crete (figure 5.11). They again show the more levelled character of the thesis classification, in that the total size of areas marked as 'Extreme' or 'Very High' is considerably lower compared to the two other schemes.
Comparing the analogue computations of area sizes for the remaining indices, ISI, BUI, FFMC, DMC, and DC, it has to be noted that the differences between the classifications are most pronounced for the fuel moisture codes. Most notably the FFMC, representing water content of fine surface litter and therefore being an indicator of the relative ease of ignition of these fuels, shows the major deviations. It therefore becomes evident that the differing forecasting results are mainly due to fuel moisture values. Since the computation methods for calculating the indices are the same for the thesis platform and the EFFIS website, one comes to the conclusion that the origin of differing forecast results lies in the varying information on the required 24h precipitation. For the thesis forecasts, an obviously higher value for precipitation is incorporated in the index computations, consecutively causing a more moderate outcome in fire danger prediction.
The average area size regarding the FFMC is shown in 5.12 for Sardinia and 5.13 for Crete.
The official outputs from the EFFIS website in summer 2012, based on the COSMO-EU model and using the EFFIS classification, are opposed to the computed area sizes based on Weather Underground data, both using the EFFIS classification and the thesis classification scheme. The script 'areacompare_qad.py' (A.1.8) is used for determining the area sizes with respect to the different danger classes. The classification system of EFFIS only covers five danger classes for the indices BUI, ISI, FFMC, DMC, and DC, excluding the 'Very High' class. For comparability purposes, the classes 'Very High' and 'Extreme' have been merged into one single 'Extreme' class in the figures showing results for the thesis' classification.

Table 5.8: Fire events, Sardinia, summer 2012

No.	Province	Date	Land cover	Burned area (km^2)
1	Nuoro	07/15	Scelerophyllous vegetation	3.62
2	Nuoro	07/20	Coniferous forest	0.18
3	Oristano	08/23	Grasslands, Broad-leaved forest	6.0

Table 5.9: Fire events, Crete, summer 2012

No.	Province	Date	Land cover	Burned area (km^2)
1	Chania	07/18	Grasslands	0.98
2	Heraklion	07/29	Grasslands	1.17
3	Heraklion	07/30	Grasslands, Scelerophyllous veg.	5.27
4	Heraklion	07/30	Grasslands, Scelerophyllous veg.	8.01
5	Lasithi	08/30	Grasslands	6.05

In analogy to tables 5.3, 5.4, and 5.5, tables 5.8 and 5.9 list fire events on Sardinia and Crete, respectively, for summer 2012. Hotspots in close spatial and temporal distance from each other have been combined to clusters of probably related fire occurrences. It is thus possible to gain information on the actual extend of a wildfire from pure hotspot data. The prominent position of the province of Heraklion in terms of wildfire activity in the Greek 2012 fire season can clearly be seen.

5.2 Discussion of findings

The objective of this thesis was the implementation of a reliable fire forecasting platform for the Isle of Sardinia. Aiming at presenting higher a spatial resolution, a longer prediction period and more accurate predictive information compared to what is currently available for the Mediterranean area, the operating platform was, at the same time, designed to exclusively depend on freely accessible data, and to be implemented using nothing but open source software.

The only system currently delivering fire danger forecasts especially for the European region is the EFFIS forecasting site run by JRC. Consequently, it is the EFFIS system that the present platform has to be tested against. Both the thesis platform and the EFFIS platform rely on the Fire Weather Index (FWI) from the Canadian Forest Fire Danger Rating System (CFFDRS) for fire danger forecasting. One advantage of the FWI systems is that it can be computed by readily available weather data. The complete CFFDRS was subjected to extensive testing and had a scientifically proven theoretical background, so that it was decided to use, as EFFIS does, this system within the scope of this paper. The FWI system comprises six subindices, one of which is the actual FWI index.

The FWI system has to be adapted to the specific region of interest, which in the case of EFFIS is the whole of Europe and Northern Africa. The EFFIS forecasts necessarily have to cover a variety of different fuels and other conditions, in contrast to the far more homogeneous situation on the Isle of Sardinia. The thesis adaptation to the area of interest takes place by determining threshold values for the fire danger classes. To subsequently obtain those values, a ten-year study was carried out regarding the fire danger situation on Sardinia and was followed by a cluster analysis. The results differ considerably from the thresholds used by EFFIS. They are, however, similar to the ones found by Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011 in that the needed minimum of the FWI value - to be interpreted as indication of an extreme fire danger situation - had to be more than twice as high as in the EFFIS classification scheme. Also the thesis results are similar in that the value of the turning point between dangerous and

not dangerous conditions is nearly identical. Due to these established thresholds the thesis forecasts yield a more detailed and realistic picture of fire danger in terms of reduction of false alarms, compared to EFFIS.

5.2.1 Validity of thresholds

K-means clustering was used to partition the resulting index values of the long-term study into six groups for each index. In case of FWI, ISI, and BUI, the limiting values of these groups represent the thresholds for the fire danger classes 'Very Low' to 'Extreme'. The procedure is equivalent for FFMC, DMC, and DC, with the difference that the classes represent fuel moisture. Since the number of desired classes was known from the outset, which is required for the k-means algorithm, this method was given preference over other classification methods, such as ISODATA. This was mostly due to efficiency reasons. To test the resilience of the found threshold values, a sensitivity analysis was conducted, decreasing and increasing these six values for each index by five percent and 10 percent, respectively. The long-term study was consecutively repeated with the adapted thresholds, concerning both error types. The results are shown in table 5.10 for the type II error, and in table 5.11 for the type I error.

Results for original and adapted threshold values regarding the type II error are shown in table 5.10 (page 146). A percentage value of five and ten, respectively, has been used for adaptation of the original values. But since there are only 152 testing points for Sardinia, and only seven for Crete, the results are of little expressiveness. For Sardinia, the mean deviation between the results regarding original and thresholds decreased by five percent is 14.66%, and 0.26% when using a five percent increase. For a decrease of ten percent, the mean deviation is 28.35%, and 4.73% in case of a ten percent increase. Regarding Crete, no meaningful statement can be concluded regrading the deviation. The only measurable deviation is for a ten percent increase, being 4.17%.

This is different for the type I error, where due to more testing points, the results are much more expressive. As can be seen in the corresponding table 5.11, the results of the long-term study do not vary significantly when threshold values

Table 5.10: Sensitivity analysis for 2001 - 2010, Type II error

Original results from the k-means clustering are highlighted grey

	Hot-spots	**Very Low (%)**	**Low (%)**	**Mode-rate (%)**	**High (%)**	**Very High (%)**	**Ex-treme (%)**
Sardinia							
WU/Thesis -10%	152	0.00	1.97	3.29	53.29	36.18	5.26
WU/Thesis -5%	152	0.00	1.97	4.61	53.95	35.53	3.95
WU/Thesis	152	0.66	1.32	11.84	54.61	28.29	3.29
WU/Thesis +5%	152	0.66	1.32	13.82	56.58	24.34	3.29
WU/Thesis +10%	152	0.66	1.97	23.03	47.37	24.34	2.63
Crete							
WU/Thesis -10%	7	0.00	0.00	0.00	100.00	0.00	0.00
WU/Thesis -5%	7	0.00	0.00	0.00	100.00	0.00	0.00
WU/Thesis	7	0.00	0.00	0.00	100.00	0.00	0.00
WU/Thesis +5%	7	0.00	0.00	0.00	100.00	0.00	0.00
WU/Thesis +10%	7	0.00	0.00	42.86	57.14	0.00	0.00

are increased or decreased by a minor percentage. Regarding Sardinia, for a five percent decrease in threshold values, the mean deviation between the results of the original and the adapted values is 7.61%, and 8.08% in case of a five percent increase. Analogue, it is 11.70% and 17.04%, respectively, in case of a ten percent increase. For Crete, the mean deviation for a five percent decrease is 6.63%, and 7.33% for an equally high increase. Using a ten percent decrease, the mean deviation is 11.16%, and 19.55% in case of a ten percent increase. The results vary in the order of the variations applied to the threshold values, as was expected. The thresholds determined by the k-means clustering method can therefore be considered to be solid.

Table 5.11: Sensitivity analysis for 2001 - 2010, type I error

Original results from the k-means clustering are highlighted grey

	Hot-spots	Very Low (%)	Low (%)	Mode-rate (%)	High (%)	Very High (%)	Ex-treme (%)
Sardinia							
WU/Thesis -10%	18070	47.59	18.93	15.05	13.09	4.65	0.69
WU/Thesis -5%	18070	48.29	19.19	15.89	12.28	3.76	0.58
WU/Thesis	18070	49.55	19.21	16.43	11.29	3.09	0.43
WU/Thesis +5%	18070	50.07	19.52	17.37	10.17	2.52	0.35
WU/Thesis +10%	18070	51.20	19.71	17.61	9.09	2.10	0.30
Crete							
WU/Thesis -10%	11360	30.81	28.23	21.45	15.60	3.12	0.80
WU/Thesis -5%	11360	31.12	29.24	22.14	14.26	2.55	0.68
WU/Thesis	11360	33.49	29.44	21.94	12.45	2.12	0.56
WU/Thesis +5%	11360	33.49	30.68	22.71	10.82	1.85	0.46
WU/Thesis +10%	11360	35.53	30.77	22.80	9.13	1.37	0.40

5.2.2 Performance-Criteria

Preparation of validation data

In the long-term study, fire danger index values for each day between 2001 and 2010 for every available weather station have been computed using historic weather data, both for Sardinia and Crete. Based on this source data, both classification schemes, the one by EFFIS and the one created in this thesis, have been applied to create vector maps of fire danger forecasting. These maps have then been tested against actually occurred fires, employing hotspot satellite data.

Hotspots are regularly used to access the occurrence of wildfires (see Feltman et al., 2012 and Koubarakis, Kontoes and Manegold, 2012, for example). Despite the fact, that cloud cover prevents burnings from being detected so that the real

numbers of fires might be underestimated (Fraser, Li and Cihlar, 2000), hotspots have proven to be an important source of information in wildfire research in recent decades. However, if a hotspot is detected by a sensor, the result still might be erroneous and represent a false detection, for example due to water reflection or sun glint (Giglio et al., 2003). More importantly, many detections might not be related to wildfires at all, but instead represent industry fires, or prescribed burnings used by farmers for land clearing reasons. When checking fire danger forecasts against the actual occurrence of wildfires in order to determine the forecast reliability, it is crucial that the occurrences actually represent wildfires, meaning that false detections and industry fires have to be excluded from this dataset. Only then, the remaining data can be used for validation, even if this means that the majority of data items is lost in that procedure. Using all the hotspots without an exclusion process would most likely lead to erroneous validation results. The first step of exclusion is done by using CORINE land cover (CLC) data, to retain hotspots occurred in natural vegetation and to eliminate those situated on dumps and industrial facilities.

It has to be stated that the spatial resolution of the CORINE 2000/06 land cover dataset, however, is suboptimal for application on a regional scale. Considering that the minimum size of a feature to be included is 25 ha, correlating to a scale of 1:100.000, many features might get lost or interpreted incorrectly during the data creation process. The thematic accuracy of 85% (see Büttner and Kosztra, 2007) involves that there exists a likelihood of 15% that a given area was classified erroneously. Furthermore, the two used CLC datasets originating in 2000 and 2006 might be considered out of date. Still, CORINE remains the most accurate and up-to-date dataset covering the whole European region. In the course of preparing this thesis, homogeneous and consistent land cover data was needed both for Sardinia and Crete, consequently, CLC data was the obvious choice. The age of the datasets was considered to be tolerable, since land use change in both regions of interest is regarded to be a slow process due to most of the islands' area being rural regions. Additionally, the data is updated regularly and it is version 15 that is used in this thesis.

To exclude possible false detections, it is checked in the second step for the remaining hotspots if the area, in which a hotspot is located, was also detected as

having been burned within the following two weeks. This is done by using the MODIS MCD45A1 product. Since hotspot data may be prone to error, this test is required as a conclusive check to ensure the reliability of the created validation data.

The filtered hotspot data is then used to validate the performance of the forecasting system. This validation is realised by means of two error tests: The type I error test, used to determine the number of false alarms, when a high fire danger is predicted without a fire actually occurring, and the type II error test, used to identify the number of situations where fires actually occurred but appropriate fire danger has not been predicted.

Type I and type II error tests

To determine the number of false alarms, referred to as type I error, the fire danger maps resulting from the long-term study have been queried by a point-in-polygon analysis. The location of five random points in the area of interest for every day between 2001 and 2010 have been used as input, for all days when no fire occurred in the study region. The results of the EFFIS approach show that, on one third of days within the investigated timespan, parts of the area were marked with 'High', 'Very High', or 'Extreme' fire danger levels, yet without an actual fire having occurred in the study area. In ten percent of the cases, EFFIS areas have been marked misleadingly with 'Extreme'. Hence, with an outcome of merely 14.8% for aggregated classes 'High', 'Very High' and 'Extreme' the type I error of the classification developed in this thesis is half as high compared to the type I error statistic of EFFIS (34.22%, see table 5.2). The results in terms of false alarms are visualised in figures 5.3 for Sardinia and 5.4 for Crete, which clearly show the more balanced character of the thesis' classification.

On the other hand, EFFIS is more successful in estimating actually occurring fires, which is addressed by the type II error. The concerning results are visualised in figure 5.1 for Sardinia, and figure 5.2 for Crete. Table 5.1 shows the percentages of successful and unsuccessful predictions. In the case of EFFIS, there are only 1.97% of the areas where occurred burnings were wrongly marked with 'Very low',

'Low' or 'Moderate' fire danger, whereas the wrongly marked areas account to 13.82% in the thesis classification.
The results of those type II errors are shown in figure 5.1 for Sardinia and 5.2 for Crete, in the study timespan of 2001 to 2010. Table 5.1 lists their values. Still, 56.6% of all fires appeared in areas marked as extreme fire danger areas using the EFFIS classification, compared to just 3.3% using the thesis scheme. Alexander, 1994 defined fire raging under these conditions as being uncontrollable by conventional means, with intensities above 4000 kW/m. However, the majority of the fires between 2001 and 2010 merely burned an area of less than 3 km^2, and so must have been necessarily manageable (see table 5.3). Only five percent of those fires burned an area larger than 25 km^2, and may therefore justly be classified as uncontrollable. It can therefore be stated that, while the EFFIS classification overall performs better than the thesis scheme regarding the type II error, it significantly overestimates the number of extreme conditions. Concerning this, the thesis classification yields more realistic results.

The results yielded by the long-term study are also clearly evident in the test for the 2012 fire season. The type I error is visualised in figure 5.8 regarding Sardinia and in figure 5.9 regarding Crete, their values are being given in table 5.7. The figures for both classifications are plotted against the actual fire danger forecasting shown on the EFFIS website (termed COSMO-EU/EFFIS in the figure legend), as well as the results of Dimitrakopoulos' classification. The figure regarding the EFFIS classification resemble each other, both with underlying Weather Underground data as well as with the modelling results of COSMO-EU. The thesis classification again performs considerably better, having a false detection rate of only 11.48%, compared to 88.52% for the EFFIS forecast. Similar values apply to Crete. The higher performance of the thesis classification is attributed to the conservativeness of computed threshold values.

Regarding the results of the type I error test, shown in 5.6 and 5.7 for Crete with values listed in 5.6, the balancing and sometimes underestimating character of the thesis scheme becomes evident. While the first two fire events have been captured by all classification schemes, regardless of used data sources, the results

differ significantly with regard to the third fire event on August 23th. While the EFFIS platform forecasting delivers appropriate results, the thesis approach underestimates the situation by marking the affected area with 'Low' fire danger. The scheme of Dimitrakopoulos even forecasts 'Very Low' fire danger. Regarding the thesis' approach, the underestimation on this day is due to the classification the three moisture codes, FFMC, DMC, and DC. As these codes were considerably less sensitive than in the EFFIS scheme, they represented moisture values at which a fire is expected to be self-extinguishing, thus exhibiting no fire danger. Effort should be put into the adaptation of the classifications of these three sub-indices to make them better suited for real-world application, such as the fire danger prediction service website of EFFIS.

The resemblance of the results in the 2012 type I and type II error tests, where the EFFIS classification has been tested both with using Weather Underground as well as official COSMO-EU input data indicates that the quality of both data sources is comparable. It is mainly the classification scheme that is responsible for the different forecasting results. This assumption about the significance of the classification scheme is further supported by a comparison of area size. In this context, the assigned percentage of area is set in relation to the respective danger levels for each index. Figures 5.10 and 5.12 show the results for Sardinia, figures 5.11 and 5.13 visualise these for Crete. Regarding the EFFIS and the thesis forecasts, the classification of the three moisture codes, and especially of DMC and DC, can be identified as the main difference in the two approaches. The conservative and more moderate character of the thesis classification can clearly be seen in the figures regarding FWI, both for Sardinia and Crete.
Over all, the newly developed scheme produces more reliable results. Yet, in field application, it is important that fire danger is rather over- than underestimated. Still, giving false alarm in thirty percent of the time, as it is the case for EFFIS during the testing period, might also reduce people's level of alertness regarding upcoming fire danger. It is therefore hard to make judgements regarding the superiority of one classification scheme over the other.

Spatial resolution, prediction period and software used

With respect to the spatial resolution, EFFIS achieves a 10 km resolution through analysis of weather models COSMO-EU and GME, which use official stations of the weather services in the respective countries for input. Aeronautica Militare Servizio Meteorologico, the Italian national meteorological office, maintains nine stations on the Isle of Sardinia (Servizio Meteorologico, 2013). On Crete, six stations are kept by the Hellenic National Meteorological Service of Greek authorities (HNMS, 2013). The platform of this thesis, however, resorts to a far greater number of weather stations. In particular, 23 stations for Sardinia and ten for the Isle of Crete were utilised from the Weather Underground database, each one also featuring historic weather data ranging back to 2001 and 2004 for Sardinia and Crete, respectively. In consequence, this allows for a higher resolution of fire danger forecasts. While the used Inverse Distance Interpolation itself is not restricted in terms of spatial resolution, a setting of 3 km was considered to be a reasonable distance regarding both expressiveness and computation speed.

Concerning the lengths of prediction terms, EFFIS provides fire danger information for the current date and three days into the future, while the thesis platform supplies a forecast for the current day and the next nine days to come. This is possible by the use of detailed weather conditions forecasts provided by Weather Underground, Inc., which maintains the worlds largest network of privately run weather stations. Of course, one has to admit that due to the high complexity of the worldwide weather system the prediction reliability significantly decreases with predicted period. The longer the forecasting period, the less accuracy can be expected (Allgöwer, Carlson and Wagtendonk, 2003). Yet, there exists a need of having a ten-days forecast, for example in tourism.

Regarding the approach of using only Open Source Applications and freely accessible data taken in this thesis, the experience has been comprehensively positive. The used software is mature and suited to address the challenge of the respective tasks. It could be shown that crowd sourced weather data is more than adequate and suitable for processing fire danger ratings. Moreover, this local fire danger

rating system implementation is workable without costs (excluding the costs of maintaining a server machine) and achievable by the work of only one person.

5.2.3 Interpolation of weather data and index results

Spatial interpolation is frequently required in a variety of applications, such as provision of input data for modelling purposes. Yet, no single method is optimal for all regions (Nalder and Wein, 1998). Aguado (2003) states that methods for spatial interpolation are always susceptible to error, this is especially true for attributes showing greater spatial variability, most notably wind (Aguado et al., 2003). However, several methods are in use for exact point data interpolation, such as Kriging, different distance-weighting methods, and spline interpolation. Geostatistical Universal Kriging and Multiple Linear Regression (MLR) have been used to model precipitation by Perčec Tadić, Gajić-Čapka and Patarčić, 2002. Stahl (2006) compared different methods for interpolation of air temperature in a large region with complex topography, concluding that MLR provided the most accurate results (with longitude, latitude and elevation used as predictor variables; Stahl et al., 2006).
In the FWI implementation of the Canadian Forest Service (CFS, NRC, 2013), Inverse Distance Weighting (IDW) is used as a means of weather parameter interpolation. Since IDW is restricted to two-dimensional space and does not consider differences in terrain, temperature values as well as relative humidity are corrected for elevation. Lawson et al. proposed IDW for interpolation of weather parameters in the weather guide for the Canadian forest fire danger rating system (Lawson and Armitage, 2008). In accordance with this proposal, the weather parameters required for computation of fire danger levels in this thesis are also interpolated using IDW. Temperature values are, as done by the CFS, height-corrected, by preliminarily recalculating these values to sea level using a lapse rate of 6.49 K / 1,000 m and transferring the interpolation results back to the actual heights of the weather stations. Through this interpolation step, data values are assigned to weather stations not having delivered measurement results. IDW is also used for interpolation of computed index results for FWI, BUI, ISI, FFMC, DMC, and DC.

However, the usage of IDW as the method of interpolation can be regarded sceptically due to the low number of weather stations available. The application of a method more suitable, such as MLR supporting topography and gradients, can be considered to be very reasonable for more accurately predicting fire danger.

5.2.4 Comparison with related works

Van Wagner originally designed the FWI system for Canadian fire danger conditions (Wagner, 1987). However, the usability of the FWI system outside of Canada was acknowledged by by Taylor and Alexander, 2006, Fogarty et al., 1998, Taylor, 2001, and Lin et al., 2000. According to the authors named, the FWI system was fully or in part implemented in North- and Southamerica, Europe, Asia, and Oceania. The possibility of implementing the system especially in region of the Mediterranean was first investigated by Viegas et al., 2000, and was afterwards confirmed by Sol (1999, see Moriondo et al., 2006) as well as by Aguado et al., 2003, Viegas et al., 2001, and Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011. In accordance with these findings, the system was implemented successfully within this thesis for the study region as well as the testing region.

The index outputs do not represent absolute measures (Fogarty et al., 1998), so the results have to be interpreted according to the region of operation. To ease interpretation, the threshold values for classifying the fire danger are set according to experience, or defined by computational means. Turner (1973) defined thresholds in a way that the frequency of occurrence of each fire danger class was identical throughout the year (Turner, 1973). Van Wagner proposed a classification according to how many days should be rated as exhibiting extreme danger (Wagner, 1987). Dimitrakopoulos, Bemmerzouk and Mitsopoulos, 2011 used logistic regression to determine thresholds adapted to eastern Mediterranean conditions. In this thesis, thresholds are also determined statistically, using k-means clustering for the range of resulting index values found during a long-term study of fire occurrence on the Isle of Sardinia.

The Fire Weather Index has been used as a means of fire danger forecasting in web-based danger prediction environments: EWS-Fire, the forecasting component of the Wildland Fire Early Warning Portal of the Global Fire Monitoring Center (GFMC; Groot et al., 2006), uses the FWI systems for worldwide predictions. It is furthermore applied in the daily operation of the European Forest Fire Information System (EFFIS) run by the Joint Research Center (JRC), for the whole European area and parts of northern Africa (Camia and Bovio, 2000, San-Miguel-Ayanz et al., 2002). The web platform developed in this thesis serves the same purpose as the online systems mentioned for the Isle of Sardinia, with the forecasts being especially adapted to this region of interest.
Sirca (2007) developed an integrated fire rating index (Ichnusa Fire Index, IFI) for Mediterranean forest types of Sardinia. The index requires meteorological, micrometeorological, topological and ecophysiological data in order to be computed. Sirca compared the IFI to other commonly used fire indices, the FWI amongst others, concluding that it produced realistic estimates of fire danger and could provide useful information on fire potential assessment on Sardinia (Sirca et al., 2007). There is, however, no publicly available implementation of this index system, so the outputs could not be tested against the prediction results yielded within the approach of this thesis.

5.2.5 Methods of optimisation

Improvement of fuel modelling

The classification of the thesis may be the superior one compared to the scheme of EFFIS for the Isle of Sardinia, provided that the classification set up is further developed by putting effort into calibrating vegetation moisture classification, while at the same time taking care of maintaining its balancing characteristics. In fact, there is a general need for constantly monitoring and enhancing a fire danger forecasting system adopted to a new region of application.
To better adapt the fuel moisture classification to the conditions on Sardinia, through field sampling over an extended period of time, however, is a time-consuming and costly process. An alternative could be the usage of actual vegetation stress data gained from extraction of Normalized Difference Vegetation Index

(NDVI) satellite data. Chuvieco (2004) combined NDVI and surface temperature for estimating the live fuel moisture content (FMC) of Mediterranean grasslands and shrub species. He successfully used NDVI values computed from NOAA-AVHRR data to model FMC, validated against six years of field measurements. Chuvieco also stressed the moisture content of fuels as a critical parameter of fire ignition, originating from the close connection to flammability (Chuvieco et al., 2004). Chen (2005) used the NDVI as well as the Normalized Difference Water Index (NDWI) computed from MODIS infrared bands to model vegetation water content of corn and soybeans, and concluded that this approach is applicable to the crops under investigation (Chen, Huang and Jackson, 2005).

An attempt of utilising NDVI data has been made during this thesis, using data from the Foreign Agricultural Service (FAS), downloadable from the MODIS Rapid Response platform at European scale, beginning in the year 2005. Unfortunately, the quality of the data was poor, so that the extraction of colour values for a given location have been found to produce unreliable results when transferring the colours into NDVI values. This approach was therefore not pursued.

Yebra (2008) developed a model based on MODIS data from 2001 to 2005 as well as field collected FMC data for Mediterranean vegetation species. It was discovered that the simulating approach and the one based on empirical data produced similar results, but that the first one exhibited greater robustness (Yebra, Chuvieco and Riaño, 2008).

The concept of using remote sensing data for fuel estimation can be considered to be a valuable way of further improving the system's reliability. This is especially valid with regard to the low and degraded vegetation on Sardinia, which would allow a satellite sensor to detect moisture related parameters on the ground without being blocked by tree crowns. Usually, discrimination between the forest canopy and understory is problematic in that live as well as dead fuels beneath tree crowns can hardly be accessed by remote sensing (Allgöwer, Carlson and Wagtendonk, 2003).

Integration of the human factor

A further useful addition to improve the forecasting quality would be to introduce the factor of human caused fire danger into the forecasting platform, as this is the most important element in fire ignition, particularly in the Mediterranean. Such a component is intended to be included in the fire behaviour part of CFFDRS, and has already been implemented in the American NFDRS by monitoring residence of people who have been delinquent because of arson in the past. Also, locations favoured by tourists and barbecue areas should lead to an increased predicted fire danger for those regions, as well as current construction sites and other places and activities known to play a reinforcing role in the probability of fire ignition. In the preparation of this thesis, a study has been executed to determine how many fires originate from natural as opposed to human activities on Sardinia. Due to lightning strokes generally being the predominant natural fire ignition cause, data on strokes on Sardinia taken from the Italian lightning detection system (SIRF) of the Centro Elettrotecnico Sperimentale Italiano (CESI) has been checked against hotspot locations. This was done by setting the conditions for a stroke and a hotspot to be possibly connected to spatial proximity of 500 m and temporal proximity of 72h. The timespan was chosen due to lags of time in satellite overpasses and in respect of smouldering fires, which could turn into wildfires a considerable time after ignition by a stroke. It was found that lightning strokes could possibly be considered responsible for as little as 0.28% of the fires occurring on Sardinia between 2001 and 2010, which is consistent with the numbers found by Susmel (1973; see Naveh, 1975) and just a little lower than the results stated by (Leone et al., 2003). This indicates that the remaining majority are probably due to human activities. For the future, the forecasting platform established in this thesis may easily be enhanced through the inclusion of additional factors or of different weather data. This once again emphasises the flexible and innovative structure of this platform.

Table 5.12: Comparison of thesis' platform and EFFIS characteristics

(Values refer to the Isle of Sardinia)

Category	**Thesis platform**	**EFFIS platform**
Forecasting region	Sardinia	Europe
Fire danger rating system	FWI system	FWI System
No. of indices available	6	6
Spatial resolution	3 km	10 km
Weather stations used	23	9
Period of prediction	10 days	4 days
data sources	crowd sourced	official
data cost expenditure	free of charge	commercial
dynamic maps of fire danger	yes	yes
re-utilisation of forecasts	available as WMS	not possible
mobile access	mobile application	no
background layer	OpenStreetMap WMS	Google WMS
satellite imagery	–	daily MODIS
type I error, 2001 - 2010	14.81%	34.22%
type II error, 2001 - 2010	13.82%	1.97%
reuse of implementation	possible	impossible

Chapter 6

Conclusions

This thesis attempted to show that a reliable fire danger forecasting system can be developed based on crowd sourced weather data. The utilised Fire Weather Index (FWI) of the Canadian Forest Fire Danger Rating system has been found to provide a reliable means of fire danger forecasting in the Mediterranean area. Sardinia was chosen as the representative region of study, and a classification scheme for FWI outputs based on a long-term study of fire occurrence on Sardinia has been developed, representing regionally adapted fire danger conditions. Testing showed that the developed system provides a more realistic picture on fire danger in the study region compared to the European Forest Fire Information System (EFFIS). However, it partly failed to predict adequate fire danger levels for some fire danger events in the testing timespan of summer 2012, probably due to inappropriate fuel moisture content modelling. Yet, regarding the long-term study, the results yielded in this work are satisfying with regard to the sensitivity of the thresholds and the reliability of the prediction. To date, unfortunately there exists no publicly available validation of the performance of EFFIS. In this respect, the thesis provides the first attempt to present such a validation.
The thesis itself stands as an example that an automated online platform implementing this system can be realised using nothing but Open Source software. It is believed that this work could be a useful contribution to fire management decision making in the Mediterranean area. Despite the fact that improvements such as improved fuel modelling should be undertaken and usage of a better suited interpolation technique is worth considering to enhance prediction quality, the created

forecasting system in the current status of development represents an applicable and innovative contribution to the growing field of research on fire forecasting in Europe. In particular, this platform can be implemented and adapted by anyone free of charge to produce fairly reliable fire danger forecasts for Sardinia. This advantage might be of special value in situations like the one that occurred to the EFFIS platform at the end of September 2012. Fundamentalist hackers successfully intruded the EFFIS website and it could not be accessed for almost two weeks (zone-h, 2012). The Joint Research Center (JRC) subsequently reacted by switching from the `Joomla!` Content Management System (CMS) to the `Python` based `Django` platform in order to increase security. Thus, decentralisation of danger forecasts and the usage of own, independent implementations may be crucial in times of increasing cyber attacks, yet more importantly, in times of immanent fire danger.

References

Aeronautica Militare Servizio Meteorologico (May 2013): *Previsioni Sardegna*. URL: http://www.meteoam.it/?regione=sardegna&timespread=prev (visited on 17/05/2013). (Cit. on p. 152).

Aguado, I et al. (2003): "Assessment of forest fire danger conditions in southern Spain from NOAA images and meteorological indices". In: *International Journal of Remote Sensing* 24.8, pp. 1653–1668. (Cit. on pp. 59, 153, 154).

Alexander, ME (1994): *Proposed revision of fire danger class criteria for forest and rural areas in New Zealand*. National Rural Fire Authority. (Cit. on p. 150).

Allgöwer, B, JD Carlson and JW van Wagtendonk (2003): "Introduction to fire danger rating and remote sensing - will remote sensing enhance wildland fire danger rating?" In: *Wildland fire danger estimation and mapping - the role of remote sensing data*. Ed. by E Chuvieco. World Scientific Publishing Co. Pte. Ltd. (Cit. on pp. 8, 18, 19, 27–29, 152, 156).

Alsabti, K, S Ranka and V Singh (1997): "An efficient k-means clustering algorithm". In: (cit. on p. 105).

Alves, CA et al. (2010): "Smoke emissions from biomass burning in a Mediterranean shrubland". In: *Atmospheric Environment* 44.25, pp. 3024–3033. (Cit. on p. 30).

Arca, B et al. (2007): "Evaluation of FARSITE simulator in Mediterranean maquis". In: *International Journal of Wildland Fire* 16.5, pp. 563–572. (Cit. on p. 17).

Bartier, PM and CP Keller (1996): "Multivariate interpolation to incorporate thematic surface data using inverse distance weighting (IDW)". In: *Computers & Geosciences* 22.7, pp. 795–799. (Cit. on p. 103).

Bodini, A and QA Cossu (2010): "Vulnerability assessment of Central-East Sardinia (Italy) to extreme rainfall events". In: *Nat. Hazards Earth Syst. Sci* 10, pp. 61–72. (Cit. on p. 22).

Boschetti, L, D Roy and AA Hoffmann (2009): "MODIS collection 5 burned area product (MCD45) user's guide version 2.0". In: *University of Maryland, South Dakota State University, LM University of Munich*. (Cit. on pp. 78, 79).

Bowman, DMJS et al. (2009): "Fire in the Earth system". In: *science* 324.5926, pp. 481–484. (Cit. on pp. 5–7, 10–12).

Brown, TJ, BL Hall and AL Westerling (2004): "The impact of twenty-first century climate change on wildland fire danger in the western United States: an applications perspective". In: *Climatic Change* 62.1-3, pp. 365–388. (Cit. on pp. 9, 19).

Burgan, RE (1988): *1988 revisions to the 1978 national fire-danger rating system*. US Department of Agriculture, Forest Service, Southeastern Forest Experiment Station. (Cit. on pp. 33, 34).

Büttner, G and B Kosztra (2007): "CLC2006 technical guidelines". In: *European Environment Agency, Technical Report*. (Cit. on pp. 87, 148).

Cai, Y, KL Judd and TS Lontzek (2012): "Open science is necessary". In: *Nature Climate Change* 2.5, pp. 299–299. (Cit. on p. 3).

Camia, A and G Bovio (2000): *Description of the indices implemented in EUDIC software for the European meteorological forest fire risk mapping*. Tech. rep. Joint Reseacrh Center of the European Commission. (Cit. on pp. 36, 41, 45, 48, 50, 52, 56, 155).

Chartzoulakis, K and G Psarras (2005): "Global change effects on crop photosynthesis and production in Mediterranean: the case of Crete, Greece". In: *Agriculture, ecosystems & environment* 106.2, pp. 147–157. (Cit. on p. 25).

Chen, D, J Huang and TJ Jackson (2005): "Vegetation water content estimation for corn and soybeans using spectral indices derived from MODIS near-and short-wave infrared bands". In: *Remote Sensing of Environment* 98.2, pp. 225–236. (Cit. on p. 156).

Chessa, PA, D Cesari and AMS Delitala (1999): "Mesoscale precipitation and temperature regimes in Sardinia (Italy) and their related synoptic circulation". In: *Theoretical and applied climatology* 63.3-4, pp. 195–221. (Cit. on p. 22).

Chessa, PA and A Delitala (1997): "Objective analysis of daily extreme temperatures of Sardinia (Italy) using distance from the sea as independent variable". In: *International journal of climatology* 17.13, pp. 1467–1485. (Cit. on p. 22).

Chou, YH (1992): "Management of wildfires with a geographical information system". In: *International Journal of Geographical Information Systems* 6.2, pp. 123–140. (Cit. on pp. 19, 29).

Chuvieco, E and RG Congalton (1989): "Application of remote sensing and geographic information systems to forest fire hazard mapping". In: *Remote Sensing of Environment* 29.2, pp. 147–159. (Cit. on p. 29).

Chuvieco, E and J de la Riva (2009): "Fire danger assessment using remote sensing and geographic information system Technologies". In: *Remote sensing for a changing Europe. Proceedings of the 28th Symposium of the European Association of Remote Sensing Laboratories, Istanbul, Turkey, 2-5 June 2008.* IOS Press, pp. 412–419. (Cit. on pp. 18, 29).

Chuvieco, E et al. (2004): "Combining NDVI and surface temperature for the estimation of live fuel moisture content in forest fire danger rating". In: *Remote Sensing of Environment* 92.3, pp. 322–331. (Cit. on p. 156).

Cohen, JD and JE Deeming (1985): *The national fire-danger rating system: basic equations*. US Department of Agriculture, Forest Service, Pacific Southwest Forest and Range Experiment Station. (Cit. on pp. 30, 33).

Columbia Encyclopedia (2012): "Crete." In: *The Columbia Encyclopedia*. 6th. Encyclopedia.com. URL: http://www.encyclopedia.com/doc/1E1-Crete.html (visited on 12/11/2012). (Cit. on p. 25).

Creative Commons (CC) (Feb. 2013): *About The Licenses*. URL: http://creativecommons.org/licenses/ (visited on 17/02/2013). (Cit. on p. 66).

Croke, B et al. (2000): "Water resources in the desertification-threatened Messara Valley of Crete: estimation of the annual water budget using a rainfall-runoff model". In: *Environmental modelling & software* 15.4, pp. 387–402. (Cit. on p. 25).

Delitala, A et al. (2000): "Precipitation over Sardinia (Italy) during the 1946–1993 rainy seasons and associated large-scale climate variations". In: *International Journal of Climatology* 20.5, pp. 519–541. (Cit. on p. 21).

Detto, M et al. (2006): "Soil moisture and vegetation controls on evapotranspiration in a heterogeneous Mediterranean ecosystem on Sardinia, Italy". In: *Water Resources Research* 42.8. (Cit. on pp. 18, 22).

Dimitrakopoulos, AP, AM Bemmerzouk and ID Mitsopoulos (2011): "Evaluation of the Canadian fire weather index system in an eastern Mediterranean environment". In: *Meteorological Applications* 18.1, pp. 83–93. (Cit. on pp. 5, 13, 16, 32, 38, 60, 144, 154).

Ding, C and X He (2004): "K-means clustering via principal component analysis". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 29. (Cit. on p. 104).

Edelsward, LM and P Salzman (1996): "Sardinians." In: *Encyclopedia of World Cultures*. Encyclopedia.com. URL: http://www.encyclopedia.com/doc/1G2-3458000700.html (visited on 12/11/2012). (Cit. on p. 22).

Englefield, P, BS Lee and RM Suddaby (2000): "Spatial fire management system". In: *Proc. 20th annual ESRI user conf., Paper*. Vol. 489. (Cit. on p. 35).

euronews.com (July 2012): *Hundreds evacuated as wildfire ravages Sardinia*. URL: http://www.euronews.com/2012/07/16/hundreds-evacuated-as-wildfire-ravages-sardinia/ (visited on 18/11/2012). (Cit. on p. 20).

European Environment Agency (July 2012a): *Corine Land Cover 2006 seamless vector data*. URL: http://www.eea.europa.eu/data-and-maps/data/clc-2006-vector-data-version-2 (visited on 22/07/2012). (Cit. on p. 87).

European Environment Agency (July 2012b): *Corine Land Cover 2006 seamless vector data*. URL: http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-clc2000-seamless-vector-database-4 (visited on 22/07/2012). (Cit. on p. 87).

Feltman, JA et al. (2012): "Geospatial Analysis Application to Forecast Wildfire Occurrences in South Carolina". In: *Forests* 3.2, pp. 265–282. (Cit. on pp. 65, 147).

Field, RD, Y Wang and Guswanto Roswintiarti O (2004): "A drought-based predictor of recent haze events in western Indonesia". In: *Atmospheric Environment* 38, pp. 1869–1878. (Cit. on p. 32).

Flannigan, M, B Stocks and M Weber (2003): "Fire regimes and climatic change in Canadian forests". In: *Fire and Climatic Change in Temperate Ecosystems of the Western Americas*, pp. 97–119. (Cit. on pp. 5–7, 10).

focus.de (July 2012): *Feuer wütet auf Sardinien und vertreibt 800 Menschen*. URL: http://www.focus.de/panorama/welt/braende-feuer-wuetet-auf-sardinien-und-vertreibt-800-menschen_aid_782469.html (visited on 18/11/2012). (Cit. on p. 20).

Fogarty, LG et al. (1998): "Adoption vs. adaptation: lessons from applying the Canadian forest fire danger rating system in New Zealand". In: *Proceedings, 3rd International Conference on Forest Fire Research and 14th Fire and Forest Meteorology Conference, Luso, Coimbra, Portugal*, pp. 1011–1028. (Cit. on pp. 32, 58–60, 63, 154).

Food and Agriculture Organization of the United Nations (2007): *Fire Management: Global Assessment 2006: a Thematic Study Prepared in the Framework*

of the Global Forest Resources Assessment 2005. FAO forestry paper, 151. Stylus Pub Llc. ISBN: 9789251056660. (Cit. on pp. 4, 6–13, 16, 32).

Food and Agriculture Organization of the United Nations, Forestry Department (2010): *Global Forest Resources Assessment 2010: Main Report*. FAO. (Cit. on pp. 4, 5).

Fraser, RH, Z Li and J Cihlar (2000): "Hotspot and NDVI differencing synergy (HANDS): A new technique for burned area mapping over boreal forest". In: *Remote Sensing of Environment* 74.3, pp. 362–376. (Cit. on pp. 64, 148).

Frazier, S (2012d): *MODIS*. NASA. URL: http://modis.gsfc.nasa.gov/about/ (visited on 16/11/2012). (Cit. on p. 74).

— (2012e): *MODIS Design Concept*. NASA. URL: http://modis.gsfc.nasa.gov/about/design.php (visited on 16/11/2012). (Cit. on p. 74).

— (2012f): *MODIS Specifications*. NASA. URL: http://modis.gsfc.nasa.gov/about/specifications.php (visited on 16/11/2012). (Cit. on p. 74).

Fyllas, NM, PG Dimitrakopoulos and AY Troumbis (2008): "Regeneration dynamics of a mixed Mediterranean pine forest in the absence of fire". In: *Forest Ecology and Management* 256.8, pp. 1552–1559. (Cit. on p. 18).

geonames.org (July 2012): *Free Gazetteer Data*. URL: http://download.geonames.org/export/dump/ (visited on 18/07/2012). (Cit. on pp. 24, 26).

Giglio, L et al. (2003): "An enhanced contextual fire detection algorithm for MODIS". In: *Remote sensing of environment* 87.2, pp. 273–282. (Cit. on pp. 65, 74, 75, 92, 148).

Gillies, S (Feb. 2013a): *Python Cartographic Library*. URL: http://zmapserver.sourceforge.net/PCL/ (visited on 14/02/2013). (Cit. on p. 69).

Gillies, S (Feb. 2013b): *Shapely*. URL: http://pypi.python.org/pypi/Shapely (visited on 14/02/2013). (Cit. on p. 119).

Goldammer, JG and V Furyaev (1996): *Fire in ecosystems of boreal Eurasia*. Vol. 48. Springer. (Cit. on p. 7).

Groisman, PY et al. (2007): "Potential forest fire danger over Northern Eurasia: changes during the 20th century". In: *Global and Planetary Change* 56.3, pp. 371–386. (Cit. on p. 34).

Groot, WJ de et al. (2006): "Developing a global early warning system for wildland fire". In: *Forest Ecology and Management* 234.1, p. 10. (Cit. on pp. 6, 29, 30, 32, 35, 78, 155).

Hahn, R (2009): "Government Policy Toward Open Source Software". In: *Available at SSRN 1411617*. (Cit. on p. 3).

Hellenic National Meteorological Service (HMNS) (May 2013): *Current observation*. URL: http://www.hnms.gr/hnms/english/observation/observation_region_html (visited on 17/05/2013). (Cit. on p. 152).

Holhs, D (Feb. 2013): *Numeric and Scientific [Packages]*. URL: http://wiki.python.org/moin/NumericAndScientific (visited on 14/02/2013). (Cit. on p. 69).

Jaiswal, RK et al. (2002): "Forest fire risk zone mapping from satellite imagery and GIS". In: *International Journal of Applied Earth Observation and Geoinformation* 4.1, pp. 1–10. (Cit. on p. 30).

Jarvis, A et al. (2008): *Hole-filled SRTM for the globe Version 4*. URL: http://www.cgiar-csi.org/data/srtm-90m-digital-elevation-database-v4-1 (visited on 18/01/2011). (Cit. on pp. 24, 26, 90, 91).

Joint Research Center of the European Commission (JRC) (July 2001): *Forest Fires in Southern Europe*. Tech. rep. No. 1. (Cit. on p. 15).

— (July 2002): *Forest Fires in Europe 2001 Fire Campaingn*. Tech. rep. No. 2. (Cit. on p. 15).

— (July 2003): *Forest Fires in Europe 2002 Fire Campaingn*. Tech. rep. No. 3. (Cit. on p. 15).

— (July 2004): *Forest Fires in Europe 2003 Fire Campaingn*. Tech. rep. No. 4. (Cit. on p. 15).

— (July 2005): *Forest Fires in Europe 2004*. Tech. rep. No. 5. (Cit. on p. 15).

— (July 2006): *Forest Fires in Europe 2005*. Tech. rep. No. 6. (Cit. on p. 15).

— (July 2007): *Forest Fires in Europe 2006*. Tech. rep. No. 7. (Cit. on p. 15).

— (July 2008): *Forest Fires in Europe 2007*. Tech. rep. No. 8. (Cit. on pp. 13–15).

— (July 2009): *Forest Fires in Europe 2008*. Tech. rep. No. 9. (Cit. on p. 15).

— (July 2010): *Forest Fires in Europe 2009*. Tech. rep. No. 10. (Cit. on p. 15).

— (July 2011): *Forest Fires in Europe 2010*. Tech. rep. No. 11. (Cit. on p. 15).

— (July 2012): *Forest Fires in Europe, Middle East and North Africa 2011*. Tech. rep. No. 12. (Cit. on pp. 13–15, 55).

— (Jan. 2013a): *About EFFIS*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/ (visited on 04/01/2013). (Cit. on p. 54).

— (Jan. 2013b): *Active Fire Detection*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/technical-background/active-fire-detection/ (visited on 04/01/2013). (Cit. on p. 56).

— (Jan. 2013c): *Brief History*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/brief-history/ (visited on 04/01/2013). (Cit. on p. 54).

Joint Research Center of the European Commission (JRC) (Jan. 2013d): *Data License*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/data-license/ (visited on 04/01/2013). (Cit. on p. 55).

— (Jan. 2013e): *EFFIS*. URL: http://forest.jrc.ec.europa.eu/effis/ (visited on 04/01/2013). (Cit. on pp. 54, 55).

— (Jan. 2013f): *EFFIS Network*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/effis-network/ (visited on 04/01/2013). (Cit. on p. 54).

— (Jan. 2013g): *European Fire Database*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/technical-background/european-fire-database/ (visited on 04/01/2013). (Cit. on p. 55).

— (Jan. 2013h): *Fire Danger Forecast*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/technical-background/fire-danger-forecast/ (visited on 04/01/2013). (Cit. on p. 56).

— (Jan. 2013i): *Legal Background*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/legal-background/ (visited on 04/01/2013). (Cit. on p. 55).

— (Jan. 2013j): *Rapid Damage Assessment*. URL: http://forest.jrc.ec.europa.eu/effis/about-effis/technical-background/rapid-damage-assessment/ (visited on 04/01/2013). (Cit. on p. 56).

Keane, RE, R Burgan and J van Wagtendonk (2001): "Mapping wildland fuels for fire management across multiple scales: Integrating remote sensing, GIS, and biophysical modeling". In: *International Journal of Wildland Fire* 10.4, pp. 301–319. (Cit. on p. 29).

Koubarakis, M, C Kontoes and S Manegold (2012): "Real-Time Wildfire Monitoring Using Scientific Database and Linked Data Technologies". In: (cit. on pp. 65, 147).

Kroes, N (2010): "How to get more interoperability in Europe". In: *Open Forum Europe*. (Cit. on p. 3).

Lawson, BD and OB Armitage (2008): *Weather guide for the Canadian forest fire danger rating system*. (Cit. on pp. 61, 104, 153).

Lee, HB and JB Macqueen (1980): "A K-Means cluster analysis computer program with cross-tabulations and next-nearest-neighbor analysis". In: *Educational and Psychological Measurement* 40.1, pp. 133–138. (Cit. on pp. 104, 105).

Leone, V et al. (2003): "The human factor in fire danger assessment". In: *Wildland fire danger estimation and mapping. The role of remote sensing data. World Scientific Publishing, Singapore*, pp. 143–196. (Cit. on pp. 7, 8, 157).

Li, Z et al. (2000): "Satellite-based mapping of Canadian boreal forest fires: evaluation and comparison of algorithms". In: *International Journal of Remote Sensing* 21.16, pp. 3071–3082. (Cit. on p. 64).

Lin, CC et al. (2000): "The development, systems, and evaluation of forest fire danger rating: a review." In: *Taiwan Journal of Forest Science* 15.4, pp. 507–520. (Cit. on pp. 7, 29, 30, 32–36, 154).

Lu, GY and DW Wong (2008): "An adaptive inverse-distance weighting spatial interpolation technique". In: *Computers & Geosciences* 34.9, pp. 1044–1055. (Cit. on pp. 102, 103).

Mantzavelas, A (2009): *FireParadox, Deliverable 5.1-1: Method to assess with good spatial accuracy the meteorological and fuel moisture components if the fire risk*. (Cit. on pp. 33, 34, 36).

Marchetti, M, C Ricotta and F Volpe (1995): "A qualitative approach to the mapping of post-fire regrowth in Mediterranean vegetation with Landsat TM data". In: *International Journal of Remote Sensing* 16.13, pp. 2487–2494. (Cit. on pp. 17, 22).

McAlpine, RS et al. (1990): "Forest fire behavior research in Canada". In: *Proceedings of the International Conference on Forest Fire Research*, pp. 19–22. (Cit. on p. 60).

Miller, C (2003): "Simulation of effects of climatic change on fire regimes". In: *Fire and climatic change in temperate ecosystems of the western Americas*, pp. 69–94. (Cit. on p. 5).

Missoula Fire Sciences Laboratory (Feb. 2013): *Fire Behavior and Fire Danger Software*. URL: http://www.firemodels.org/ (visited on 18/02/2013). (Cit. on p. 35).

Moriondo, M et al. (2006): "Potential impact of climate change on fire risk in the Mediterranean area". In: *Climate Research* 31.1, pp. 85–95. (Cit. on pp. 11, 17, 59, 154).

Mouillot, F, S Rambal and R Joffre (2002): "Simulating climate change impacts on fire frequency and vegetation dynamics in a Mediterranean-type ecosystem". In: *Global Change Biology* 8.5, pp. 423–437. (Cit. on pp. 12, 16, 23).

Murphy, K (2012a): *About FIRMS*. NASA. URL: http://earthdata.nasa.gov/data/nrt-data/firms/about (visited on 16/11/2012). (Cit. on pp. 74, 76).

Murphy, K (2012b): *Disclaimer*. NASA. URL: http://earthdata.nasa.gov/data/near-real-time-data/disclaimer (visited on 16/11/2012). (Cit. on p. 74).

— (2012c): *FAQ*. NASA. URL: http://earthdata.nasa.gov/data/nrt-data/help/faq (visited on 16/11/2012). (Cit. on p. 77).

— (2013): *Near Real-Time Data*. NASA. URL: https://earthdata.nasa.gov/data/near-real-time-data/firms/about (visited on 16/03/2013). (Cit. on p. 77).

Nakayama, M et al. (1999): "Contextual algorithm adapted for NOAA-AVHRR fire detection in Indonesia". In: *International Journal of Remote Sensing* 20.17, pp. 3415–3421. (Cit. on p. 65).

Nalder, IA and RW Wein (1998): "Spatial interpolation of climatic normals: test of a new method in the Canadian boreal forest". In: *Agricultural and forest meteorology* 92.4, pp. 211–225. (Cit. on p. 153).

Natural Resources Canada (Jan. 2013): *Background Information - Canadian Forest Fire Weather Index (FWI) System*. URL: http://cwfis.cfs.nrcan.gc.ca/background/dsm/fwi (visited on 30/01/2013). (Cit. on pp. 103, 104, 106, 112, 113, 153).

Naveh, Z (1975): "The evolutionary significance of fire in the Mediterranean region". In: *Vegetatio* 29.3, pp. 199–208. (Cit. on pp. 8, 16, 17, 157).

Open Data Commons (ODC) (Feb. 2013): *ODC Open Database License (ODbL) Summary*. URL: http://opendatacommons.org/licenses/odbl/summary/ (visited on 17/02/2013). (Cit. on p. 66).

Open Geospatial Consortium (OGC) (Feb. 2013a): *About OGC*. URL: http://www.opengeospatial.org/ogc (visited on 13/02/2013). (Cit. on p. 120).

Open Geospatial Consortium (OGC) (Feb. 2013b): *Geography Markup Language*. URL: http://www.opengeospatial.org/standards/gml (visited on 15/02/2013). (Cit. on p. 72).

— (Feb. 2013c): *Web Map Service*. URL: http://www.opengeospatial.org/standards/wms (visited on 13/02/2013). (Cit. on p. 120).

Open Source Geospatial Foundation (OSGeo) (Feb. 2013): *About the Open Source Geospatial Foundation*. URL: http://www.osgeo.org/content/foundation/about.html (visited on 15/02/2013). (Cit. on p. 67).

Open Source Initiative (OSI) (Feb. 2013a): *GNU General Public License, version 3 (GPL-3.0)*. URL: http://opensource.org/licenses/GPL-3.0 (visited on 17/02/2013). (Cit. on p. 66).

— (Feb. 2013b): *The BSD 3-Clause License*. URL: http://opensource.org/licenses/BSD-3-Clause (visited on 17/02/2013). (Cit. on p. 66).

— (Feb. 2013c): *The GNU Lesser General Public License, version 3.0 (LGPL-3.0)*. URL: http://opensource.org/licenses/LGPL-3.0 (visited on 17/02/2013). (Cit. on p. 66).

— (Feb. 2013d): *The MIT License (MIT)*. URL: http://opensource.org/licenses/MIT (visited on 17/02/2013). (Cit. on p. 66).

— (Feb. 2013e): *The Open Source Definition (Annotated)*. URL: http://opensource.org/docs/definition.php (visited on 17/02/2013). (Cit. on p. 65).

OpenLayers Dev Team (Feb. 2013): *OpenLayers*. URL: http://openlayers.org/ (visited on 15/02/2013). (Cit. on p. 71).

OpenStreetMap Community (Feb. 2013): *About (Open Street Map)*. URL: http://wiki.openstreetmap.org/wiki/About (visited on 15/02/2013). (Cit. on p. 87).

Papadopoulos, A et al. (2013): "Investigating the relationship of meteorological/-climatological conditions and wildfires in Greece". In: *Theoretical and applied climatology* 112.1-2, pp. 113–126. (Cit. on p. 7).

Parry, ML et al. (2007a): *IPCC, 2007: climate change 2007: impacts, adaptation and vulnerability. Contribution of working group II to the fourth assessment report of the intergovernmental panel on climate change*. (Cit. on p. 10).

Pausas, JG (1999a): "Mediterranean vegetation dynamics: modelling problems and functional types". In: *Plant Ecology* 140.1, pp. 27–39. (Cit. on pp. 11, 16–18).

— (1999b): "Response of plant functional types to changes in the fire regime in Mediterranean ecosystems: a simulation approach". In: *Journal of Vegetation Science* 10.5, pp. 717–722. (Cit. on pp. 11, 17).

Perčec Tadić, M, Ma Gajić-Čapka and M Patarčić (2002): "Geostatistical Methods in Producing Mean Annual Precipitation Map for Croatia". In: *European Conference on Applied Climatology*. (Cit. on p. 153).

Perry, GLW, AD Sparrow and IF Owens (1999): "A GIS-supported model for the simulation of the spatial structure of wildland fire, Cass Basin, New Zealand". In: *Journal of Applied Ecology* 36.4, pp. 502–518. (Cit. on p. 30).

Price, OF and RA Bradstock (2012): "The efficacy of fuel treatment in mitigating property loss during wildfires: Insights from analysis of the severity of the catastrophic fires in 2009 in Victoria, Australia". In: *Journal of environmental management* 113, pp. 146–157. (Cit. on p. 7).

Pyne, SJ, PL Andrews and RD Laven (1996): *Introduction to wildland fire*. Wiley. (Cit. on pp. 1, 5, 7–9, 20).

R Foundation (Feb. 2013): *The R Project for Statistical Computing*. URL: http://www.r-project.org/about.html (visited on 14/02/2013). (Cit. on p. 70).

Rossotti, H (1994): *Feuer. Vom zündenden Funken bis zum flammenden Inferno*. Vol. 48. Spektrum Akademischer Verlag GmbH. (Cit. on p. 4).

Roy, DP et al. (2005): "Prototyping a global algorithm for systematic fire-affected area mapping using MODIS time series data". In: *Remote sensing of the environment* 97.2, pp. 137–162. (Cit. on p. 79).

San-Miguel-Ayanz, J (2013): "Free and open source software underpinning the european forest data centre". In: (cit. on p. 3).

San-Miguel-Ayanz, J et al. (2002): "Towards a coherent forest fire information system in Europe: the European Forest Fire Information System (EFFIS)". In: *Forest fire research & wildland fire safety*. (Cit. on pp. 12, 32, 54, 56, 155).

San-Miguel-Ayanz, J et al. (2003a): "Correlation Analysis and fuel moisture estimation based on FMA and FMA+ fire danger indices in a pinus elliottii plantation in Southern Brazil". In: *Wildland fire danger: estimation and mapping: the role of remote sensing data*. Ed. by E Chuvieco. Vol. 4. World Scientific Publishing Company Incorporated. (Cit. on p. 38).

San-Miguel-Ayanz, J et al. (2003b): "Current methods to assess fire danger potential". In: *Wildland fire danger estimation and mapping - the role of remote sensing data*. Ed. by E Chuvieco. World Scientific Publishing Co. Pte. Ltd. (Cit. on p. 34).

San-Miguel-Ayanz, J et al. (2003c): "The European forest fire information system: a European strategy towards forest fire management". In: *Proceedings of*

the 3rd International Wildland Fire Conference, Sydney, Australia. (Cit. on pp. 12, 13, 30, 54, 56).

Schmidt, KM et al. (2002): *Development of coarse-scale spatial data for wildland fire and fuel management*. US Department of Agriculture, Forest Service, Rocky Mountain Research Station Fort Collins (CO). (Cit. on p. 30).

Sencha Inc. (Feb. 2013): *Sencha Touch*. URL: http://www.sencha.com/products/touch (visited on 15/02/2013). (Cit. on p. 72).

Shepard, D (1968): "A two-dimensional interpolation function for irregularly-spaced data". In: *Proceedings of the 1968 23rd ACM national conference*. ACM, pp. 517–524. (Cit. on p. 102).

Sirca, C et al. (2007): "Performance of a newly developed integrated fire rating index in Sardinia, Italy". In: *Proceedings of the 4th International WildLand Fire Conference. Seville, Spain*, pp. 13–17. (Cit. on pp. 22, 155).

Solomon, S et al. (2007b): "The physical science basis". In: *Contribution of working group I to the fourth assessment report of the intergovernmental panel on climate change*, pp. 235–337. (Cit. on p. 10).

Stahl, K et al. (2006): "Comparison of approaches for spatial interpolation of daily air temperature in a large region with complex topography and highly variable station density". In: *Agricultural and Forest Meteorology* 139.3, pp. 224–236. (Cit. on p. 153).

Stallman, R (2005): "Free community science and the free development of science". In: *PLoS Medicine* 2.2, e47. (Cit. on p. 3).

Stocks, BJ et al. (1989): "Canadian forest fire danger rating system: an overview". In: *The Forestry Chronicle* 65.4, pp. 258–265. (Cit. on pp. 29, 30, 37–39, 41, 45, 47, 50–52).

sz-online.de (July 2012): *Feuer auf Sardinien jetzt im Süden*. URL: http://www.sz-online.de/nachrichten/1800-hektar-auf-sardinien-eingeaeschert-feuer-jetzt-im-sueden-1466706.html (visited on 18/11/2012). (Cit. on p. 20).

tagesschau.de (July 2012): *Waldbrände in Südeuropa wüten weiter*. URL: http://www.tagesschau.de/ausland/waldbraende188.html (visited on 17/11/2012). (Cit. on pp. 19, 20).

Taylor, SW (2001): "Considerations for Applying the Canadian Forest Fire Danger Rating System in Argentina". In: *Unpublished report. Canadian Forest Service, Pacific Forestry Centre: Victoria, BC*. (Cit. on pp. 32, 34, 57, 58, 60, 62, 63, 154).

Taylor, SW and ME Alexander (2006): "Science, technology, and human factors in fire danger rating: the Canadian experience." In: *International Journal of Wildland Fire* 15.1, pp. 121–135. (Cit. on pp. 32, 154).

The Weather Channel, LLC weather.com (July 2012a): *Monthly Averages for Island of Crete, Greece*. URL: http://www.weather.com/weather/wxclimatology/monthly/graph/GRXX0046 (visited on 22/07/2012). (Cit. on p. 26).

— (July 2012b): *Monthly Averages for Sardinia, Italy*. URL: http://www.weather.com/weather/wxclimatology/monthly/graph/ITXX0187 (visited on 22/07/2012). (Cit. on p. 21).

Trenberth, KE (1997): "The Definition of El Niño". In: *Bulletin of the American Meteorological Society* 78.12, pp. 2771–2777. (Cit. on p. 12).

Turner, JA (1973): *A Fire Load Index For British Columbia*. Pacific Forest Research Centre, Victoria, British Columbia. (Cit. on pp. 62, 154).

University of Berkeley, Museum of Vertebrate Zoology and the International Rice Research Institute (July 2012): *Global Administrative Areas*. URL: http://www.gadm.org/country (visited on 18/07/2012). (Cit. on pp. 24, 26, 90, 91, 95, 96, 98).

University of Minnesota (Feb. 2013): *MapServer*. URL: http://mapserver.org/en/index.html (visited on 15/02/2013). (Cit. on p. 71).

Vacca, A (2000): "Effect of land use on forest floor and soil of a Quercus suber L. forest in Gallura (Sardinia, Italy)". In: *Land Degradation & Development* 11.2, pp. 167–180. (Cit. on p. 22).

Vacca, A et al. (2002): "Soil degradation in Sardinia (Italy): main factors and processes". In: *7th international meeting on soils with Mediterranean type of climate. Bari: Centre international de hautes études agronomiques méditerranéennes-Instituto Agronomico Mediterraneo di Bari (Ciheam-IAMB)*. (Cit. on pp. 21, 22).

Viegas, DX et al. (2000): "Comparative study of various methods of fire danger evaluation in Southern Europe". In: *International Journal of Wildland Fire* 9.4, pp. 235–246. (Cit. on pp. 30, 32, 59, 154).

Viegas, DX et al. (2001): "Estimating live fine fuels moisture content using meteorologically-based indices". In: *International Journal of Wildland Fire* 10.2, pp. 223–240. (Cit. on pp. 59, 60, 154).

Wagner, CE van (1983): "Fire behavior in northern conifer forests and shrublands". In: *The role of fire in northern circumpolar ecosystems*. Ed. by RW Wein and DA Maclean. John Wiley and Sons, pp. 65–80. (Cit. on p. 6).

Wagner, CE van (1987): *Development and structure of the Canadian forest fire weather index system*. NRC Research Press Ottawa, Canada. (Cit. on pp. 32, 37, 62, 99, 154).

Wagner, CE van and TL Pickett (1985): *Equations and FORTRAN program for the Canadian forest fire weather index system*. Vol. 33. Canadian Forestry Service. (Cit. on pp. 38, 42, 45, 48, 50–52).

Warmerdam, F (Feb. 2013a): *GDAL - Geospatial Data Abstraction Library*. URL: http://www.gdal.org/ (visited on 14/02/2013). (Cit. on p. 69).

— (Feb. 2013b): *GDAL Utilities*. URL: http://www.gdal.org/gdal_utilities.html (visited on 14/02/2013). (Cit. on p. 69).

— (Feb. 2013c): *OGR Simple Feature Library*. URL: http://www.gdal.org/ogr/ (visited on 14/02/2013). (Cit. on p. 70).

— (Feb. 2013d): *PROJ.4*. URL: http://trac.osgeo.org/proj/ (visited on 14/02/2013). (Cit. on p. 70).

— (Feb. 2013e): *Software using GDAL*. URL: http://trac.osgeo.org/gdal/wiki/SoftwareUsingGdal (visited on 14/02/2013). (Cit. on p. 70).

Weather Underground, Inc. (Jan. 2013a): *About Us*. URL: http://www.wunderground.com/about/background.asp (visited on 23/01/2013). (Cit. on p. 80).

— (Jan. 2013b): *Code Samples*. URL: http://www.wunderground.com/weather/api/d/docs?d=resources/code-samples (visited on 23/01/2013). (Cit. on p. 82).

— (Jan. 2013c): *Press Releases*. URL: http://www.wunderground.com/about/pr/news.asp (visited on 23/01/2013). (Cit. on p. 80).

Weather Underground, Inc. (Jan. 2013d): *PWS Network*. URL: http://www.wunderground.com/weatherstation/ListStations.asp (visited on 23/01/2013). (Cit. on pp. 81, 90, 91).

— (Jan. 2013e): *Start Coding in under 5 Minutes for free*. URL: http://www.wunderground.com/weather/api/d/pricing.html (visited on 23/01/2013). (Cit. on p. 86).

— (Jan. 2013f): *Terms and Conditions of Use*. URL: http://www.wunderground.com/members/tos.asp (visited on 23/01/2013). (Cit. on p. 80).

— (Jan. 2013g): *What is a PWS?* URL: http://www.wunderground.com/weatherstation/about.asp (visited on 23/01/2013). (Cit. on p. 81).

Wessel, P (Feb. 2013): *The Generic Mapping Tools*. URL: http://gmt.soest.hawaii.edu/ (visited on 14/02/2013). (Cit. on p. 71).

Wessel, P and WHF Smith (1996): "A global, self-consistent, hierarchical, high-resolution shoreline database". In: *Journal of geophysical research* 101, pp. 8741–8743. (Cit. on pp. 24, 26, 90, 91).

Williams, AAJ, DJ Karoly and N Tapper (2001): "The sensitivity of Australian fire danger to climate change". In: *Climatic change* 49.1-2, pp. 171–191. (Cit. on p. 19).

Wotton, BM (2009): "Interpreting and using outputs from the Canadian Forest Fire Danger Rating System in research applications". In: *Environmental and Ecological Statistics* 16.2, pp. 107–131. (Cit. on p. 30).

Yasrebi, J et al. (2009): "Evaluation and comparison of ordinary kriging and inverse distance weighting methods for prediction of spatial variability of some soil chemical parameters". In: *Research Journal of Biological Sciences* 4.1, pp. 93–102. (Cit. on p. 102).

Yebra, M, E Chuvieco and D Riaño (2008): "Estimation of live fuel moisture content from MODIS images for fire risk assessment". In: *Agricultural and Forest Meteorology* 148.4, pp. 523–536. (Cit. on p. 156).

zeit.de (July 2012): *Feuer auf Sardinien jetzt auch im Süden*. URL: http://www.zeit.de/news/2012-07/17/wetter-feuer-auf-sardinien-jetzt-auch-im-sueden-17105006 (visited on 18/11/2012). (Cit. on p. 20).

zone-h (Sept. 2012): *effis.jrc.ec.europa.eu hacked. Notified by NeT-DeViL*. URL: http://www.zone-h.org/mirror/id/18380180 (visited on 17/03/2013). (Cit. on p. 160).

List of Figures

List of Tables

Eidesstattliche Erklärung

Hiermit versichere ich, dass die Abhandlung - abgesehen von der Beratung durch den Betreuer - nach Inhalt und Form meine eigene Arbeit ist, weiterhin, dass die Arbeit weder ganz noch zum Teil schon einer anderen Stelle im Rahmen eines Prüfungsverfahrens vorgelegen hat, veröffentlicht worden ist oder zur Veröffentlichung eingereicht wurde, und schließlich, dass die Arbeit unter Einhaltung der Regeln guter wissenschaftlicher Praxis der Deutschen Forschungsgemeinschaft entstanden ist.

Michael Nolde

A FIRE DANGER FORECASTING PLATFORM FOR SARDINIA

- APPENDIX -

Appendix

A.1 Python scripts

A.1.1 pipogrextr.py

```
1  import os,os.path,shutil
2  import osgeo.ogr
3  import shapely.wkt
4  import sys
5  #from shapely.geometry import Polygon
6  from shapely.geometry import Point
7  import subprocess
8
9
10 nohead = False
11 ingml_filename = ""
12 incsv_filename = ""
13
14 for j in range(1,len(sys.argv)):
15   if sys.argv[j][0:6].lower() == "ingml=":
16     ingml_filename=sys.argv[j][6:]
17   if sys.argv[j][0:10].lower() == "ingml_tag=":
18     ingml_tag=sys.argv[j][10:]
19   if sys.argv[j][0:16].lower() == "ingml_tagvalues=":
20     ingml_tagvalues=sys.argv[j][16:]
21     ingml_tagvalueslist=ingml_tagvalues.split(",")
```

```
22    if sys.argv[j][0:6].lower() == "incsv=":
23      incsv_filename=sys.argv[j][6:]
24    if sys.argv[j][0:8].lower() == "incsv_x=":
25      incsv_x=int(sys.argv[j][8:])
26    if sys.argv[j][0:8].lower() == "incsv_y=":
27      incsv_y=int(sys.argv[j][8:])
28    if sys.argv[j][0:7].lower() == "outcsv=":
29      outcsv_filename=sys.argv[j][7:]
30
31    if sys.argv[j][0:7].lower() == "-nohead":
32      nohead = True
33
34  inshp_filename = ingml_filename[:-4] + ".shp"
35  return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + inshp_filename +↩
        ' ' + ingml_filename, shell=True)
36
37  geofile = osgeo.ogr.Open(inshp_filename)
38  layer = geofile.GetLayer(0)
39
40
41  incsv=open(incsv_filename, 'r')
42  if nohead == True:
43    titel = incsv.readline().strip()
44
45  outcsv=open(outcsv_filename, 'w')
46  if nohead == True:
47    outcsv.write(titel + "," + ingml_tag + "\n")
48
49  while True:
50    rline = incsv.readline().strip()
51    rlinelist = rline.split(",")
52
53    if len(rline)==0:
54      break
55
56    p = Point(float(rlinelist[incsv_x - 1]), float(rlinelist[incsv_y - 1]))
57
```

```
58
59    # Durchlaufen aller in der Quelldatei enthaltenen Features
60    for i in range(layer.GetFeatureCount()):
61      featureItem = layer.GetFeature(i)
62      geometry = featureItem.GetGeometryRef()
63      feature = shapely.wkt.loads(geometry.ExportToWkt())
64
65      #print feature.geom_type
66
67      if feature.contains(p) == True:
68        print "punkt gefunden!"
69
70        for j in range(0, len(ingml_tagvalueslist)):
71          if str(featureItem.GetField(ingml_tag)) == ingml_tagvalueslist[j]:
72            outcsv.write(rline + "," + ingml_tagvalueslist[j] + "\n")
73
74  incsv.close()
75  outcsv.close()
```

A.1.2 mcd45split.py

```
1  #!/usr/bin/python
2  # mcd45split
3
4  # teilt Features anhand eines Attributwertes in verschiedene Dateien auf
5  # Ergebnisdateien vom letzten Durchlauf "_" muessen vor dem Start geloescht ↩
       werden
6
7  import os
8  import sys
9  import datetime
10
11 removefiles = 0
12
13 if removefiles == 1:
14
15   for h in range(2001,2011):
16     h_str = str(h)
17
18     for j in range(1, 367):
19       j_str = str(j)
20       if len(j_str) == 2:
21         j_str = "0" + j_str
22       if len(j_str) == 1:
23         j_str = "00" + j_str
24
25       for k in range(1, 367):
26         k_str = str(k)
27
28         filename = "output/mcd45monthly_a" + h_str + j_str +"↩
               _win08_005_burndate_" + k_str + ".gml"
29         if os.path.isfile(filename):
30           os.remove(filename)
31
32
33
```

```
34 outfilenames = []
35 infilenames = []
36
37
38
39 infilenames.append('mcd45monthly_a2001001_win08_005_burndate.gml')
40 infilenames.append('mcd45monthly_a2001032_win08_005_burndate.gml')
41 infilenames.append('mcd45monthly_a2001060_win08_005_burndate.gml')
42 infilenames.append('mcd45monthly_a2001091_win08_005_burndate.gml')
43 infilenames.append('mcd45monthly_a2001121_win08_005_burndate.gml')
44 infilenames.append('mcd45monthly_a2001182_win08_005_burndate.gml')
45 infilenames.append('mcd45monthly_a2001213_win08_005_burndate.gml')
46 infilenames.append('mcd45monthly_a2001244_win08_005_burndate.gml')
47 infilenames.append('mcd45monthly_a2001274_win08_005_burndate.gml')
48 infilenames.append('mcd45monthly_a2001305_win08_005_burndate.gml')
49 infilenames.append('mcd45monthly_a2001335_win08_005_burndate.gml')
50
51 infilenames.append('mcd45monthly_a2002001_win08_005_burndate.gml')
52 infilenames.append('mcd45monthly_a2002032_win08_005_burndate.gml')
53 infilenames.append('mcd45monthly_a2002060_win08_005_burndate.gml')
54 infilenames.append('mcd45monthly_a2002091_win08_005_burndate.gml')
55 infilenames.append('mcd45monthly_a2002121_win08_005_burndate.gml')
56 infilenames.append('mcd45monthly_a2002152_win08_005_burndate.gml')
57 infilenames.append('mcd45monthly_a2002182_win08_005_burndate.gml')
58 infilenames.append('mcd45monthly_a2002213_win08_005_burndate.gml')
59 infilenames.append('mcd45monthly_a2002244_win08_005_burndate.gml')
60 infilenames.append('mcd45monthly_a2002274_win08_005_burndate.gml')
61 infilenames.append('mcd45monthly_a2002305_win08_005_burndate.gml')
62 infilenames.append('mcd45monthly_a2002335_win08_005_burndate.gml')
63
64 infilenames.append('mcd45monthly_a2003001_win08_005_burndate.gml')
65 infilenames.append('mcd45monthly_a2003032_win08_005_burndate.gml')
66 infilenames.append('mcd45monthly_a2003060_win08_005_burndate.gml')
67 infilenames.append('mcd45monthly_a2003091_win08_005_burndate.gml')
68 infilenames.append('mcd45monthly_a2003121_win08_005_burndate.gml')
69 infilenames.append('mcd45monthly_a2003152_win08_005_burndate.gml')
70 infilenames.append('mcd45monthly_a2003182_win08_005_burndate.gml')
```

```
71  infilenames.append('mcd45monthly_a2003213_win08_005_burndate.gml')
72  infilenames.append('mcd45monthly_a2003244_win08_005_burndate.gml')
73  infilenames.append('mcd45monthly_a2003274_win08_005_burndate.gml')
74  infilenames.append('mcd45monthly_a2003305_win08_005_burndate.gml')
75  infilenames.append('mcd45monthly_a2003335_win08_005_burndate.gml')
76
77  infilenames.append('mcd45monthly_a2004001_win08_005_burndate.gml')
78  infilenames.append('mcd45monthly_a2004032_win08_005_burndate.gml')
79  infilenames.append('mcd45monthly_a2004061_win08_005_burndate.gml')
80  infilenames.append('mcd45monthly_a2004092_win08_005_burndate.gml')
81  infilenames.append('mcd45monthly_a2004122_win08_005_burndate.gml')
82  infilenames.append('mcd45monthly_a2004153_win08_005_burndate.gml')
83  infilenames.append('mcd45monthly_a2004183_win08_005_burndate.gml')
84  infilenames.append('mcd45monthly_a2004214_win08_005_burndate.gml')
85  infilenames.append('mcd45monthly_a2004245_win08_005_burndate.gml')
86  infilenames.append('mcd45monthly_a2004275_win08_005_burndate.gml')
87  infilenames.append('mcd45monthly_a2004306_win08_005_burndate.gml')
88  infilenames.append('mcd45monthly_a2004336_win08_005_burndate.gml')
89
90  infilenames.append('mcd45monthly_a2005001_win08_005_burndate.gml')
91  infilenames.append('mcd45monthly_a2005032_win08_005_burndate.gml')
92  infilenames.append('mcd45monthly_a2005060_win08_005_burndate.gml')
93  infilenames.append('mcd45monthly_a2005091_win08_005_burndate.gml')
94  infilenames.append('mcd45monthly_a2005121_win08_005_burndate.gml')
95  infilenames.append('mcd45monthly_a2005152_win08_005_burndate.gml')
96  infilenames.append('mcd45monthly_a2005182_win08_005_burndate.gml')
97  infilenames.append('mcd45monthly_a2005213_win08_005_burndate.gml')
98  infilenames.append('mcd45monthly_a2005244_win08_005_burndate.gml')
99  infilenames.append('mcd45monthly_a2005274_win08_005_burndate.gml')
100 infilenames.append('mcd45monthly_a2005305_win08_005_burndate.gml')
101 infilenames.append('mcd45monthly_a2005335_win08_005_burndate.gml')
102
103 infilenames.append('mcd45monthly_a2006001_win08_005_burndate.gml')
104 infilenames.append('mcd45monthly_a2006032_win08_005_burndate.gml')
105 infilenames.append('mcd45monthly_a2006060_win08_005_burndate.gml')
106 infilenames.append('mcd45monthly_a2006091_win08_005_burndate.gml')
107 infilenames.append('mcd45monthly_a2006121_win08_005_burndate.gml')
```

```
108 infilenames.append('mcd45monthly_a2006152_win08_005_burndate.gml')
109 infilenames.append('mcd45monthly_a2006182_win08_005_burndate.gml')
110 infilenames.append('mcd45monthly_a2006213_win08_005_burndate.gml')
111 infilenames.append('mcd45monthly_a2006244_win08_005_burndate.gml')
112 infilenames.append('mcd45monthly_a2006274_win08_005_burndate.gml')
113 infilenames.append('mcd45monthly_a2006305_win08_005_burndate.gml')
114 infilenames.append('mcd45monthly_a2006335_win08_005_burndate.gml')
115
116 infilenames.append('mcd45monthly_a2007001_win08_005_burndate.gml')
117 infilenames.append('mcd45monthly_a2007032_win08_005_burndate.gml')
118 infilenames.append('mcd45monthly_a2007060_win08_005_burndate.gml')
119 infilenames.append('mcd45monthly_a2007091_win08_005_burndate.gml')
120 infilenames.append('mcd45monthly_a2007121_win08_005_burndate.gml')
121 infilenames.append('mcd45monthly_a2007152_win08_005_burndate.gml')
122 infilenames.append('mcd45monthly_a2007182_win08_005_burndate.gml')
123 infilenames.append('mcd45monthly_a2007213_win08_005_burndate.gml')
124 infilenames.append('mcd45monthly_a2007244_win08_005_burndate.gml')
125 infilenames.append('mcd45monthly_a2007274_win08_005_burndate.gml')
126 infilenames.append('mcd45monthly_a2007305_win08_005_burndate.gml')
127 infilenames.append('mcd45monthly_a2007335_win08_005_burndate.gml')
128
129 infilenames.append('mcd45monthly_a2008001_win08_005_burndate.gml')
130 infilenames.append('mcd45monthly_a2008032_win08_005_burndate.gml')
131 infilenames.append('mcd45monthly_a2008061_win08_005_burndate.gml')
132 infilenames.append('mcd45monthly_a2008092_win08_005_burndate.gml')
133 infilenames.append('mcd45monthly_a2008122_win08_005_burndate.gml')
134 infilenames.append('mcd45monthly_a2008153_win08_005_burndate.gml')
135 infilenames.append('mcd45monthly_a2008183_win08_005_burndate.gml')
136 infilenames.append('mcd45monthly_a2008214_win08_005_burndate.gml')
137 infilenames.append('mcd45monthly_a2008245_win08_005_burndate.gml')
138 infilenames.append('mcd45monthly_a2008275_win08_005_burndate.gml')
139 infilenames.append('mcd45monthly_a2008306_win08_005_burndate.gml')
140 infilenames.append('mcd45monthly_a2008336_win08_005_burndate.gml')
141
142 infilenames.append('mcd45monthly_a2009001_win08_005_burndate.gml')
143 infilenames.append('mcd45monthly_a2009032_win08_005_burndate.gml')
144 infilenames.append('mcd45monthly_a2009060_win08_005_burndate.gml')
```

```
145 infilenames.append('mcd45monthly_a2009091_win08_005_burndate.gml')
146 infilenames.append('mcd45monthly_a2009121_win08_005_burndate.gml')
147 infilenames.append('mcd45monthly_a2009152_win08_005_burndate.gml')
148 infilenames.append('mcd45monthly_a2009182_win08_005_burndate.gml')
149 infilenames.append('mcd45monthly_a2009213_win08_005_burndate.gml')
150 infilenames.append('mcd45monthly_a2009244_win08_005_burndate.gml')
151 infilenames.append('mcd45monthly_a2009274_win08_005_burndate.gml')
152 infilenames.append('mcd45monthly_a2009305_win08_005_burndate.gml')
153 infilenames.append('mcd45monthly_a2009335_win08_005_burndate.gml')
154
155 infilenames.append('mcd45monthly_a2010001_win08_005_burndate.gml')
156 infilenames.append('mcd45monthly_a2010032_win08_005_burndate.gml')
157 infilenames.append('mcd45monthly_a2010060_win08_005_burndate.gml')
158 infilenames.append('mcd45monthly_a2010091_win08_005_burndate.gml')
159 infilenames.append('mcd45monthly_a2010121_win08_005_burndate.gml')
160 infilenames.append('mcd45monthly_a2010152_win08_005_burndate.gml')
161 infilenames.append('mcd45monthly_a2010182_win08_005_burndate.gml')
162 infilenames.append('mcd45monthly_a2010213_win08_005_burndate.gml')
163 infilenames.append('mcd45monthly_a2010244_win08_005_burndate.gml')
164 infilenames.append('mcd45monthly_a2010274_win08_005_burndate.gml')
165 infilenames.append('mcd45monthly_a2010305_win08_005_burndate.gml')
166 infilenames.append('mcd45monthly_a2010335_win08_005_burndate.gml')
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
```

```
182 for i in range(0, len(infilenames)):
183
184
185   infilename = infilenames[i]
186
187
188   infile=open(infilename, 'r')
189
190
191   while True:
192     rline = infile.readline()
193
194     if len(rline)==0:
195       break
196
197     if '<gml:featureMember>' in rline:
198       rline2 = infile.readline()
199       rline3 = infile.readline()
200       rline4 = infile.readline()
201       rline5 = infile.readline()
202       rline6 = infile.readline()
203
204       pos1 = rline4.find('<ogr:DN>')
205       pos2 = rline4.find('</ogr:DN>')
206
207       #Julian day
208       jday = rline4[pos1 + 8 : pos2]
209
210       #Gregorian date
211       year = infilename[14:18]
212       datestr = year + " " + jday
213
214       gregdate_tmp = str(datetime.datetime.strptime(datestr, '%Y %j'))
215       gregdate_tmp = gregdate_tmp[:10]
216       gregdate = gregdate_tmp.replace("-","")
217
218       dn = gregdate
```

```
219
220
221         if os.path.isfile("output/" + infilename[:-4] + "_" + dn + ".gml"):
222           outfile=open("output/" + infilename[:-4] + "_" + dn + ".gml", 'a')
223           outfile.write(rline)
224           outfile.write(rline2)
225           outfile.write(rline3)
226           outfile.write(rline4)
227           outfile.write(rline5)
228           outfile.write(rline6)
229           outfile.close()
230
231         else:
232           outfilenames.append(infilename[:-4] + "_" + dn + ".gml")
233
234           outfile=open("output/" + infilename[:-4] + "_" + dn + ".gml", 'w')
235           outfile.write('<?xml version="1.0" encoding="utf-8" ?>' + '\n')
236           outfile.write('<ogr:FeatureCollection' + '\n')
237           outfile.write('     xmlns:xsi="http://www.w3.org/2001/XMLSchema-↩
                  instance"' + '\n')
238           outfile.write('     xsi:schemaLocation="http://ogr.maptools.org/ test.↩
                  xsd"' + '\n')
239           outfile.write('     xmlns:ogr="http://ogr.maptools.org/"' + '\n')
240           outfile.write('     xmlns:gml="http://www.opengis.net/gml">' + '\n')
241           outfile.write('' + '\n')
242           outfile.write(rline)
243           outfile.write(rline2)
244           outfile.write(rline3)
245           outfile.write(rline4)
246           outfile.write(rline5)
247           outfile.write(rline6)
248           outfile.close()
249
250     infile.close()
251
252 for i in range(0, len(outfilenames)):
253
```

```
254     outfile=open("output/" + outfilenames[i], 'a')
255     outfile.write('</ogr:FeatureCollection>')
256     outfile.close()
```

A.1.3 pipburnedarea_extr.py

```
from datetime import datetime
from datetime import date
from datetime import timedelta
from datetime import *
import sys
import subprocess
import os,shutil
from shapely.geometry import Point
import osgeo.ogr
import shapely.wkt


for j in range(1,len(sys.argv)):
  if sys.argv[j][0:8].lower() == "timedif=":
    timedif=int(sys.argv[j][8:])
  if sys.argv[j][0:8].lower() == "incsv_x=":
    incsv_x=int(sys.argv[j][8:])
  if sys.argv[j][0:8].lower() == "incsv_y=":
    incsv_y=int(sys.argv[j][8:])
  if sys.argv[j][0:11].lower() == "incsv_date=":
    incsv_date=int(sys.argv[j][11:])
  if sys.argv[j][0:7].lower() == "outcsv=":
    outcsv_filename=sys.argv[j][7:]
  if sys.argv[j][0:6].lower() == "incsv=":
    incsv_filename=sys.argv[j][6:]
  if sys.argv[j][0:8].lower() == "in_path=":
    in_path=sys.argv[j][8:]
  if sys.argv[j][0:11].lower() == "in_praefix=":
    in_praefix=sys.argv[j][11:]
  if sys.argv[j][0:12].lower() == "in_praefix2=":
    in_praefix2=sys.argv[j][12:]
  if sys.argv[j][0:7].lower() == "-nohead":
    nohead = True
  if sys.argv[j][0:4].lower() == "reg=":
    reg=sys.argv[j][4:]
```

```
36
37  hits=0
38
39  incsv=open(incsv_filename, 'r')
40  if nohead == True:
41    titel = incsv.readline().strip()
42
43  outcsv=open(outcsv_filename, 'w')
44  if nohead == True:
45    outcsv.write(titel + "," + "burnedarea" + "\n")
46
47  while True:
48    rline = incsv.readline().strip()
49    rlinelist = rline.split(",")
50
51
52    if len(rline)==0:
53      break
54
55
56    p = Point(float(rlinelist[incsv_x - 1]), float(rlinelist[incsv_y - 1]))
57    found = False
58
59
60
61
62    s_year = int(rlinelist[incsv_date - 1][0:4])
63    s_month = int(rlinelist[incsv_date - 1][4:6])
64    s_day = int(rlinelist[incsv_date - 1][6:8])
65
66
67
68    start = datetime(s_year, s_month, s_day)
69    end = start + timedelta(timedif)
70    dt = start
71
72
```

```
73    i = 0
74
75    while dt <= end:
76      if found == True:
77        break
78
79      dt_year_str = str(dt.year)
80      dt_month_str = str(dt.month)
81      dt_day_str = str(dt.day)
82
83      dt_month_out_str = dt_month_str
84      dt_day_out_str = dt_day_str
85
86      if len(dt_month_str) == 1:
87        dt_month_out_str = "0" + dt_month_out_str
88      if len(dt_day_str) == 1:
89        dt_day_out_str = "0" + dt_day_out_str
90
91      for jyear in range(2001,2013):
92        jyear_str = str(jyear)
93
94        for jday in range(1, 367):
95          jday_str = str(jday)
96          if len(jday_str) == 1:
97            jday_str = "00" + jday_str
98          if len(jday_str) == 2:
99            jday_str = "0" + jday_str
100
101
102         geofilename = in_path + in_praefix + jyear_str + jday_str + in_praefix2↩
                + dt_year_str + dt_month_out_str + dt_day_out_str
103         if os.path.isfile(geofilename + ".gml"):
104           #print geofilename + ".gml"
105
106           return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + ↩
                geofilename + ' ' + geofilename +'.gml', shell=True)
107           if reg.lower() == "sard":
```

```
108             return_code = subprocess.call('ogr2ogr -t_srs EPSG:32632 -s_srs ↩
                    EPSG:4326 ' + geofilename + '/' + 'out_32632' + '.shp ' + ↩
                    geofilename + '/' + 'out' + '.shp', shell=True)
109           else:
110             return_code = subprocess.call('ogr2ogr -t_srs EPSG:32635 -s_srs ↩
                    EPSG:4326 ' + geofilename + '/' + 'out_32635' + '.shp ' + ↩
                    geofilename + '/' + 'out' + '.shp', shell=True)
111
112           print "hits: " + str(hits)
113
114           #Sardinien
115           #if reg.lower() == "sard":
116             #return_code = subprocess.call('ogr2ogr -clipdst 8.116666666666667 ↩
                    38.84861111111111 9.856944444444444 41.325 ' + geofilename + '/↩
                    out_clipsard.shp ' + geofilename + '/out.shp', shell=True)
117             #return_code = subprocess.call('ogr2ogr -t_srs EPSG:32632 -s_srs ↩
                    EPSG:4326 ' + geofilename + '/out_clipsard_32632.shp ' + ↩
                    geofilename + '/out_clipsard.shp', shell=True)
118             #return_code = subprocess.call('ogr2ogr -f "GMT" ' + geofilename + ↩
                    '_clipsard_32632.gmt' + ' ' + geofilename + '/↩
                    out_clipsard_32632.shp', shell=True)
119
120           #Kreta
121           #if reg.lower() == "kreta":
122             #return_code = subprocess.call('ogr2ogr -clipdst 23.45 ↩
                    34.78055555555555 26.38055555555556 35.71388888888889 ' + ↩
                    geofilename + '/out_clipkreta.shp ' + geofilename + '/out.shp',↩
                     shell=True)
123             #return_code = subprocess.call('ogr2ogr -t_srs EPSG:32632 -s_srs ↩
                    EPSG:4326 ' + geofilename + '/out_clipkreta_32632.shp ' + ↩
                    geofilename + '/out_clipkreta.shp', shell=True)
124             #return_code = subprocess.call('ogr2ogr -f "GMT" ' + geofilename + ↩
                    '_clipkreta_32632.gmt' + ' ' + geofilename + '/↩
                    out_clipkreta_32632.shp', shell=True)
125
126
127           geofile = osgeo.ogr.Open(in_path + in_praefix + jyear_str + jday_str ↩
```

```
            + in_praefix2 + dt_year_str + dt_month_out_str + dt_day_out_str +↩
              "/out.shp")
128         layer = geofile.GetLayer(0)
129
130         if reg.lower() == "sard":
131           geofile2 = osgeo.ogr.Open(in_path + in_praefix + jyear_str + ↩
                jday_str + in_praefix2 + dt_year_str + dt_month_out_str + ↩
                dt_day_out_str + "/out_32632.shp")
132         else:
133           geofile2 = osgeo.ogr.Open(in_path + in_praefix + jyear_str + ↩
                jday_str + in_praefix2 + dt_year_str + dt_month_out_str + ↩
                dt_day_out_str + "/out_32635.shp")
134
135         layer2 = geofile2.GetLayer(0)
136
137         # Durchlaufen aller in der Quelldatei enthaltenen Features
138         #print layer.GetFeatureCount()
139
140         for j in range(layer.GetFeatureCount()):
141           featureItem = layer.GetFeature(j)
142           geometry = featureItem.GetGeometryRef()
143           feature = shapely.wkt.loads(geometry.ExportToWkt())
144
145           featureItem2 = layer2.GetFeature(j)
146           geometry2 = featureItem2.GetGeometryRef()
147           feature2 = shapely.wkt.loads(geometry2.ExportToWkt())
148
149
150           if feature.contains(p) == True:
151             print " punkt gefunden!"
152             hits = hits + 1
153             outcsv.write(rline + "," + str(round(feature2.area / 1000000, 4))↩
                  + "\n")
154             found = True
155
156
157     dt = dt + timedelta(1)
```

```
158
159         i = i + 1
160
161         #sys.exit()
162
163
164 incsv.close()
165 outcsv.close()
```

A.1.4 hitratio.py

```
1  import sys
2  #import time as t
3  #from datetime import datetime
4  #from datetime import date
5  #from datetime import timedelta
6  #from datetime import *
7  import subprocess
8  #import os
9  import os,os.path,shutil
10 import osgeo.ogr
11 import shapely.wkt
12 import sys
13 #from shapely.geometry import Polygon
14 from shapely.geometry import Point
15
16 nohead = False
17 namingconv = ""
18 ndvi = ""
19
20 for j in range(1,len(sys.argv)):
21   if sys.argv[j][0:7].lower() == "s_date=":
22     s_datestr=sys.argv[j][7:]
23   if sys.argv[j][0:7].lower() == "e_date=":
24     e_datestr=sys.argv[j][7:]
25   if sys.argv[j][0:7].lower() == "inpath=":
26     inpath=sys.argv[j][7:]
27   if sys.argv[j][0:6].lower() == "incsv=":
28     incsv_filename=sys.argv[j][6:]
29   if sys.argv[j][0:7].lower() == "outcsv=":
30     outcsv_filename=sys.argv[j][7:]
31   if sys.argv[j][0:8].lower() == "incsv_x=":
32     incsv_x=int(sys.argv[j][8:])
33   if sys.argv[j][0:8].lower() == "incsv_y=":
34     incsv_y=int(sys.argv[j][8:])
35   if sys.argv[j][0:4].lower() == "reg=":
```

```
36      reg=sys.argv[j][4:]
37    if sys.argv[j][0:6].lower() == "index=":
38      index=sys.argv[j][6:]
39    if sys.argv[j][0:11].lower() == "incsv_date=":
40      incsv_date=int(sys.argv[j][11:])
41    if sys.argv[j][0:11].lower() == "namingconv=":
42      namingconv=sys.argv[j][11:]
43    if sys.argv[j][0:7].lower() == "s_date=":
44      s_date=sys.argv[j][7:]
45    if sys.argv[j][0:7].lower() == "e_date=":
46      e_date=sys.argv[j][7:]
47    if sys.argv[j][0:9].lower() == "outdates=":
48      outdates=sys.argv[j][9:]
49    if sys.argv[j][0:7].lower() == "-nohead":
50      nohead = True
51    if sys.argv[j][0:5].lower() == "-ndvi":
52      ndvi = "ndvi_"
53
54  outdateslist = outdates.split(",")
55
56  #sard_fwi_idw_20080313_airports_pws_fwi
57
58
59
60  incsv=open(incsv_filename, 'r')
61  outcsv=open(outcsv_filename, 'w')
62
63  if nohead == True:
64    titel = incsv.readline().strip()
65    outcsv.write(titel + "," + "fwidn" + "\n")
66
67  s_date_int = int(s_date)
68  e_date_int = int(e_date)
69
70  while True:
71    rline = incsv.readline().strip()
72
```

```
73    if len(rline)==0:
74      break
75
76    rlinelist = rline.split(",")
77    datestr = rlinelist[incsv_date - 1]
78    datestr_year = datestr[0:4]
79    datestr_month = datestr[4:6]
80    datestr_day = datestr[6:8]
81
82    outdate=False
83    for j in range(0,len(outdateslist)):
84      if datestr == outdateslist[j]:
85        outdate=True
86
87    if int(datestr) >= s_date_int and int(datestr) <= e_date_int and outdate == ↩
          False:
88
89      p = Point(float(rlinelist[incsv_x - 1]), float(rlinelist[incsv_y - 1]))
90
91      print datestr_day
92
93      if namingconv.lower() == "effis":
94        ingml_filename = inpath + datestr_year + "/" + datestr_month + "/" + ↩
              datestr_year + datestr_month + datestr_day + "_" + reg + "_" + index ↩
              + ".gml"
95        inshp_filename = inpath + datestr_year + "/" + datestr_month + "/" + ↩
              datestr_year + datestr_month + datestr_day + "_" + reg + "_" + index ↩
              + ".shp"
96      else:
97        ingml_filename = inpath + datestr_year + "/" + datestr_month + "/" + reg ↩
              + "_fwi_idw_" + datestr_year + datestr_month + datestr_day + "↩
              _airports_pws_" + ndvi + index + ".gml"
98        inshp_filename = inpath + datestr_year + "/" + datestr_month + "/" + reg ↩
              + "_fwi_idw_" + datestr_year + datestr_month + datestr_day + "↩
              _airports_pws_" + ndvi + index + ".shp"
99
100     return_code = subprocess.call('rm ' + inshp_filename, shell=True)
```

```
101     return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + ↩
            inshp_filename + ' ' + ingml_filename, shell=True)
102
103
104     geofile = osgeo.ogr.Open(inshp_filename)
105     layer = geofile.GetLayer(0)
106
107     for i in range(layer.GetFeatureCount()):
108       featureItem = layer.GetFeature(i)
109       geometry = featureItem.GetGeometryRef()
110
111       feature = shapely.wkt.loads(geometry.ExportToWkt())
112
113       if feature.contains(p) == True:
114         #print "punkt gefunden!"
115         outcsv.write(rline + "," + str(int(featureItem.GetField("DN"))) + "\n")
116
117
118 sys.exit()
```

A.1.5 randompoints.py

```
1  from datetime import datetime
2  from datetime import date
3  from datetime import timedelta
4  from datetime import *
5  import sys
6  import subprocess
7  import os
8  from random import randint
9
10
11 for j in range(1,len(sys.argv)):
12   if sys.argv[j][0:7].lower() == "s_date=":
13     s_datestr=sys.argv[j][7:]
14   if sys.argv[j][0:7].lower() == "e_date=":
15     e_datestr=sys.argv[j][7:]
16   if sys.argv[j][0:6].lower() == "incsv=":
17     incsv_filename=sys.argv[j][6:]
18   if sys.argv[j][0:7].lower() == "outcsv=":
19     outcsv_filename=sys.argv[j][7:]
20   if sys.argv[j][0:8].lower() == "incsv_x=":
21     incsv_x=int(sys.argv[j][8:])
22   if sys.argv[j][0:8].lower() == "incsv_y=":
23     incsv_y=int(sys.argv[j][8:])
24   if sys.argv[j][0:11].lower() == "incsv_date=":
25     incsv_date=int(sys.argv[j][11:])
26   if sys.argv[j][0:13].lower() == "ingml_random=":
27     ingml_random_filename=sys.argv[j][13:]
28   if sys.argv[j][0:9].lower() == "nrpoints=":
29     nrpoints=int(sys.argv[j][9:])
30   if sys.argv[j][0:7].lower() == "-nohead":
31     nohead = True
32
33
34 randompoints=[]
35
```

```
36
37 ingml_random=open(ingml_random_filename, 'r')
38 outcsv=open(outcsv_filename, 'w')
39
40
41 if nohead == True:
42   incsv=open(incsv_filename, 'r')
43   titel = incsv.readline()
44   incsv.close()
45   outcsv.write(titel)
46
47
48 while True:
49   rline = ingml_random.readline()
50
51   if len(rline)==0:
52     break
53
54   searchstr1 = "<ogr:geometryProperty><gml:Point><gml:coordinates>"
55   searchstr2 = "</gml:coordinates></gml:Point></ogr:geometryProperty>"
56
57   if rline.find(searchstr1) <> -1:
58     rline_coords_start = rline.find(searchstr1) + len(searchstr1)
59     rline_coords_end = rline.find(searchstr2)
60     #print rline[rline_coords_start:rline_coords_end]
61     randompoints.append(rline[rline_coords_start:rline_coords_end])
62
63
64
65
66
67 s_year = int(s_datestr[0:4])
68 s_month = int(s_datestr[4:6])
69 s_day = int(s_datestr[6:8])
70
71 e_year = int(e_datestr[0:4])
72 e_month = int(e_datestr[4:6])
```

```
73  e_day = int(e_datestr[6:8])
74
75
76
77  start = datetime(s_year, s_month, s_day)
78  end = datetime(e_year, e_month, e_day)
79  dt = start
80
81
82  i = 0
83
84  while dt <= end:
85
86    dt_year_str = str(dt.year)
87    dt_month_str = str(dt.month)
88    dt_day_str = str(dt.day)
89
90    dt_month_out_str = dt_month_str
91    dt_day_out_str = dt_day_str
92
93    if len(dt_month_str) == 1:
94      dt_month_out_str = "0" + dt_month_out_str
95    if len(dt_day_str) == 1:
96      dt_day_out_str = "0" + dt_day_out_str
97
98
99    #infile = open(incsv_path + incsv_praefix + dt_year_str + dt_month_out_str + ↩
          dt_day_out_str + ".csv", 'r')
100   print dt_year_str + dt_month_out_str + dt_day_out_str
101
102   founddate = False
103   incsv=open(incsv_filename, 'r')
104
105
106   while True:
107     rline = incsv.readline()
108     rlinelist = rline.split(",")
```

```
109
110       if len(rline)==0:
111         break
112
113       if rlinelist[incsv_date - 1] == dt_year_str + dt_month_out_str + ↩
              dt_day_out_str:
114         founddate = True
115
116     incsv.close()
117
118     if founddate == False:
119       for i in range(0, nrpoints):
120         #print len(randompoints)
121         #sys.exit()
122         rpoint = randint(0,len(randompoints)-1)
123         outcsv.write(randompoints[rpoint] + "," + dt_year_str + dt_month_out_str ↩
              + dt_day_out_str + "," + "120300,349.9,2.6,1.5,A↩
              ,95,5.1,311.7,212.4,323" + "\n")
124
125     dt = dt + timedelta(1)
126
127     i = i + 1
128
129   #sys.exit()
```

A.1.6 fireevents.py

```
1  import urllib2
2  import json
3  import sys
4  import time
5  import datetime
6
7
8  for j in range(1,len(sys.argv)):
9    if sys.argv[j][0:8].lower() == "timedif=":
10     timedif=int(sys.argv[j][8:])
11   if sys.argv[j][0:7].lower() == "posdif=":
12     posdif=int(sys.argv[j][7:])
13   if sys.argv[j][0:8].lower() == "incsv_x=":
14     incsv_x=int(sys.argv[j][8:])
15   if sys.argv[j][0:8].lower() == "incsv_y=":
16     incsv_y=int(sys.argv[j][8:])
17   if sys.argv[j][0:11].lower() == "incsv_date=":
18     incsv_date=int(sys.argv[j][11:])
19   if sys.argv[j][0:7].lower() == "outcsv=":
20     outcsv_filename=sys.argv[j][7:]
21   if sys.argv[j][0:6].lower() == "incsv=":
22     incsv_filename=sys.argv[j][6:]
23   if sys.argv[j][0:7].lower() == "-nohead":
24     nohead = True
25
26 outtakes=[]
27
28 incsv1=open(incsv_filename, 'r')
29 outcsv=open(outcsv_filename, 'w')
30
31
32 if nohead == True:
33   titel = incsv1.readline().strip()
34   outcsv.write("Event" + "," + titel + "\n")
35
```

```
36  rline1_c = 0
37  events = 0
38
39  while True:
40    rline1 = incsv1.readline().strip()
41    newevent = True
42    rline1list = rline1.split(",")
43
44    if len(rline1)==0:
45      break
46
47    date1 = datetime.date(int(rline1list[incsv_date-1][0:4]), int(rline1list[↩
          incsv_date-1][4:6]), int(rline1list[incsv_date-1][6:8]))
48    rline1_x = float(rline1list[incsv_x-1])
49    rline1_y = float(rline1list[incsv_y-1])
50
51
52
53    incsv2=open(incsv_filename, 'r')
54    if nohead == True:
55      titel = incsv2.readline().strip()
56
57    rline2_c = 0
58    while True:
59
60      rline2 = incsv2.readline().strip()
61      rline2list = rline2.split(",")
62
63      if len(rline2)==0:
64        break
65
66      date2 = datetime.date(int(rline2list[incsv_date-1][0:4]), int(rline2list[↩
            incsv_date-1][4:6]), int(rline2list[incsv_date-1][6:8]))
67      rline2_x = float(rline2list[incsv_x-1])
68      rline2_y = float(rline2list[incsv_y-1])
69
70
```

```
71      if date2 >= date1 - datetime.timedelta(days=timedif) and date2 <= date1 + ↩
            datetime.timedelta(days=timedif):
72        dist_x = (rline1_x - rline2_x)**2
73        dist_y = (rline1_y - rline2_y)**2
74
75        if dist_x < 0:
76          dist_x = dist_x * -1
77        if dist_y < 0:
78          dist_y = dist_y * -1
79
80        dist = (dist_x + dist_y)**0.5
81        if dist <= posdif:
82          #print dist
83
84
85
86          writecsv = True
87          for i in range(0, len(outtakes)):
88            if rline2_c == outtakes[i]:
89              writecsv = False
90
91          if writecsv == True:
92            if newevent == True:
93              events = events + 1
94              newevent = False
95
96            outcsv.write(str(events) + "," + rline2 + "\n")
97            print str(events) + "," + str(dist) + " " + str(rline1_x) +"," + str(↩
                  rline1_y) + "/" + str(rline2_x) +"," + str(rline2_y)
98            outtakes.append(rline2_c)
99
100
101
102     rline2_c = rline2_c + 1
103
104   incsv2.close()
105   rline1_c = rline1_c + 1
```

```
106
107 incsv1.close()
108
109 outcsv.close()
```

A.1.7 effisextr.py

```
1  import PIL
2  import Image
3  import numpy
4  import sys
5  import colorsys
6
7  import math
8
9  def hsv2rgb(h, s, v):
10     h = float(h)
11     s = float(s)
12     v = float(v)
13     h60 = h / 60.0
14     h60f = math.floor(h60)
15     hi = int(h60f) % 6
16     f = h60 - h60f
17     p = v * (1 - s)
18     q = v * (1 - f * s)
19     t = v * (1 - (1 - f) * s)
20     r, g, b = 0, 0, 0
21     if hi == 0: r, g, b = v, t, p
22     elif hi == 1: r, g, b = q, v, p
23     elif hi == 2: r, g, b = p, v, t
24     elif hi == 3: r, g, b = p, q, v
25     elif hi == 4: r, g, b = t, p, v
26     elif hi == 5: r, g, b = v, p, q
27     r, g, b = int(r * 255), int(g * 255), int(b * 255)
28     return r, g, b
29
30 def rgb2hsv(r, g, b):
31     r, g, b = r/255.0, g/255.0, b/255.0
32     mx = max(r, g, b)
33     mn = min(r, g, b)
34     df = mx-mn
35     if mx == mn:
```

```
36          h = 0
37      elif mx == r:
38          h = (60 * ((g-b)/df) + 360) % 360
39      elif mx == g:
40          h = (60 * ((b-r)/df) + 120) % 360
41      elif mx == b:
42          h = (60 * ((r-g)/df) + 240) % 360
43      if mx == 0:
44          s = 0
45      else:
46          s = df/mx
47      v = mx
48      return h, s, v
49
50
51
52  for j in range(1,len(sys.argv)):
53    if sys.argv[j][0:3].lower() == "in=":
54      infilename=sys.argv[j][3:]
55    if sys.argv[j][0:4].lower() == "out=":
56      outfilename=sys.argv[j][4:]
57    if sys.argv[j][0:4].lower() == "reg=":
58      reg=sys.argv[j][4:]
59
60
61  pic = Image.open(infilename)
62  pic.load()
63  R,G,B = pic.split()
64  f1rot = numpy.asarray(R)
65  f1gruen = numpy.asarray(G)
66  f1blau = numpy.asarray(B)
67
68  if reg == "kreta":
69    regheight = 220
70    regwidth = 540
71    ystep = 22
72    xstep = 18
```

```
73
74  if reg == "sard":
75    regheight = 600
76    regwidth = 324
77    ystep = 24
78    xstep = 18
79
80  erg = numpy.zeros((regheight,regwidth,3), 'uint8')
81  ergpixel_c = 0
82
83  for y in range(0,regheight,ystep):
84    for x in range(0,regwidth,18):
85
86      f1rotval_summe = 0
87      f1gruenval_summe = 0
88      f1blauval_summe = 0
89
90
91      kachelx_c = 0
92      kachely_c = 0
93
94      for kachely in range(y+3, y+19):
95        kachely_c = kachely_c + 1
96        kachelx_c = 0
97
98        for kachelx in range(x+2, x+16):
99          kachelx_c = kachelx_c + 1
100
101         f1rotval = int(f1rot[kachely,kachelx])
102         f1gruenval = int(f1gruen[kachely,kachelx])
103         f1blauval = int(f1blau[kachely,kachelx])
104
105         f1hsv = rgb2hsv(f1rot[kachely,kachelx], f1gruen[kachely,kachelx], ↩
                f1blau[kachely,kachelx])
106
107         f1hue = f1hsv[0]
108         f1sat = f1hsv[1]
```

```
109         f1val = f1hsv[2]
110
111
112         f1rotval_summe = f1rotval_summe + f1rotval
113         f1gruenval_summe = f1gruenval_summe + f1gruenval
114         f1blauval_summe = f1blauval_summe + f1blauval
115
116     #print kachely_c
117     #print kachelx_c
118     #sys.exit()
119
120     f1rotval_schnitt = f1rotval_summe / (16 * 14)
121     f1gruenval_schnitt = f1gruenval_summe / (16 * 14)
122     f1blauval_schnitt = f1blauval_summe / (16 * 14)
123
124     ergtext = False
125
126
127     ergrot = 0
128     erggruen = 0
129     ergblau = 0
130
131     if f1gruenval_schnitt <> 0 or f1rotval_schnitt <> 0 or f1blauval_schnitt <>↩
            0:
132
133       if f1gruenval_schnitt >= 100:
134
135         if f1gruenval_schnitt >= 225:
136
137           #GELB
138           if f1rotval_schnitt >= 225:
139             ergpixel_c = ergpixel_c + 1
140             #ergrot = 255
141             #erggruen = 255
142             #ergblau = 0
143             ergrot = 2
144             erggruen = 2
```

```
145             ergblau = 2
146
147           #GRUEN
148           if f1rotval_schnitt < 225:
149             ergpixel_c = ergpixel_c + 1
150             #ergrot = 0
151             #erggruen = 128
152             #ergblau = 0
153             ergrot = 1
154             erggruen = 1
155             ergblau = 1
156
157         #ORANGE
158         if f1gruenval_schnitt < 225:
159           ergpixel_c = ergpixel_c + 1
160           #ergrot = 255
161           #erggruen = 165
162           #ergblau = 0
163           ergrot = 3
164           erggruen = 3
165           ergblau = 3
166
167       if f1gruenval_schnitt < 100:
168
169         #ROT
170         if f1rotval_schnitt >= 240:
171           ergpixel_c = ergpixel_c + 1
172           #ergrot = 255
173           #erggruen = 0
174           #ergblau = 0
175           ergrot = 4
176           erggruen = 4
177           ergblau = 4
178
179         if f1rotval_schnitt < 240:
180           #DUNKELROT
181           if f1blauval_schnitt < 25:
```

```
182             ergpixel_c = ergpixel_c + 1
183             #ergrot = 128
184             #erggruen = 0
185             #ergblau = 0
186             ergrot = 5
187             erggruen = 5
188             ergblau = 5
189
190           #VIOLET
191           if f1blauval_schnitt >= 25:
192             ergpixel_c = ergpixel_c + 1
193             #ergrot = 128
194             #erggruen = 0
195             #ergblau = 128
196             ergrot = 6
197             erggruen = 6
198             ergblau = 6
199
200
201
202     for ergkachely in range(y, y+ystep):
203       for ergkachelx in range(x, x+18):
204         erg[ergkachely,ergkachelx,0] = ergrot
205         erg[ergkachely,ergkachelx,1] = erggruen
206         erg[ergkachely,ergkachelx,2] = ergblau
207
208 #print "detektierte Pixel: " + str(ergpixel_c)
209
210 im = Image.fromarray(erg)
211 im.save(outfilename)
```

A.1.8 areacompare_qad.py

```
from datetime import datetime
from datetime import date
from datetime import timedelta
from datetime import *
import sys
import subprocess
import os
import osgeo.ogr
import shapely.wkt

s_date_str="20120701"
e_date_str="20120906"
outdates="20120713,20120718,20120721,20120722,20120825"

index = "dc"
reg = "sard"
datasource = "thesis"


if reg.lower() == "sard":
  if datasource.lower() == "thesis":
    outcsv = open("./../data/firedanger/area/areacompare_sard_32632_ownfwi_" + index + ".csv", 'w')
  if datasource.lower() == "effis":
    outcsv = open("./../data/firedanger/area/areacompare_sard_32632_effisfwi_" + index + ".csv", 'w')

if reg.lower() == "kreta":
  if datasource.lower() == "thesis":
    outcsv = open("./../data/firedanger/area/areacompare_kreta_32635_ownfwi_" + index + ".csv", 'w')
  if datasource.lower() == "effis":
    outcsv = open("./../data/firedanger/area/areacompare_kreta_32635_effisfwi_" + index + ".csv", 'w')

```

```
32
33 outdateslist=outdates.split(",")
34
35 s_year = int(s_date_str[0:4])
36 s_month = int(s_date_str[4:6])
37 s_day = int(s_date_str[6:8])
38
39 e_year = int(e_date_str[0:4])
40 e_month = int(e_date_str[4:6])
41 e_day = int(e_date_str[6:8])
42
43
44
45 start = datetime(s_year, s_month, s_day)
46 end = datetime(e_year, e_month, e_day)
47 dt = start
48
49
50 ingml3_area_verylow_total = 0
51 ingml3_area_low_total = 0
52 ingml3_area_moderate_total = 0
53 ingml3_area_high_total = 0
54 ingml3_area_veryhigh_total = 0
55 ingml3_area_extreme_total = 0
56
57 days=0
58
59
60 while dt <= end:
61
62   dt_year_str = str(dt.year)
63   dt_month_str = str(dt.month)
64   dt_day_str = str(dt.day)
65
66   dt_month_out_str = dt_month_str
67   dt_day_out_str = dt_day_str
68
```

```
69    if len(dt_month_str) == 1:
70      dt_month_out_str = "0" + dt_month_out_str
71    if len(dt_day_str) == 1:
72      dt_day_out_str = "0" + dt_day_out_str
73
74    outdate=False
75    for j in range(0, len(outdateslist)):
76      datestr = dt_year_str + dt_month_out_str + dt_day_out_str
77      if datestr == outdateslist[j]:
78        outdate=True
79
80    ingml3_area_verylow = 0
81    ingml3_area_low = 0
82    ingml3_area_moderate = 0
83    ingml3_area_high = 0
84    ingml3_area_veryhigh = 0
85    ingml3_area_extreme = 0
86
87    if outdate==False:
88
89      days=days+1
90      #print "sard_fwi_idw_" + dt_year_str + dt_month_out_str + dt_day_out_str + ↩
          "_airports_pws_" + "fwi" + ".shp"
91
92      if reg.lower() == "sard":
93        if datasource.lower() == "thesis":
94          shpfile = "./../data/weather/wunderground_2011/fwiidw/sard/" + ↩
              dt_year_str + "/" + dt_month_out_str + "/" + "sard_fwi_idw_" + ↩
              dt_year_str + dt_month_out_str + dt_day_out_str + "_airports_pws_" ↩
              + index
95        if datasource.lower() == "effis":
96          shpfile = "./../data/firedanger/effis_2012/erg/sard/" + dt_year_str + "↩
              /" + dt_month_out_str + "/" + dt_year_str + dt_month_out_str + ↩
              dt_day_out_str + "_sard_" + index
97
98        shpfileclip = "./../data/boundaries/gadm_2006/sardinia/↩
            ita_adm_sardclp_32632/ita_adm0_sardclp_32632"
```

```
99
100     if reg.lower() == "kreta":
101       if datasource.lower() == "thesis":
102         shpfile = "./../data/weather/wunderground_2011/fwiidw/kreta/" + ↩
                dt_year_str + "/" + dt_month_out_str + "/" + "kreta_fwi_idw_" + ↩
                dt_year_str + dt_month_out_str + dt_day_out_str + "_airports_pws_" ↩
                + index
103       if datasource.lower() == "effis":
104         shpfile = "./../data/firedanger/effis_2012/erg/kreta/" + dt_year_str + ↩
                "/" + dt_month_out_str + "/" + dt_year_str + dt_month_out_str + ↩
                dt_day_out_str + "_kreta_" + index
105
106       shpfileclip = "./../data/boundaries/gadm2_2012/kreta/↩
              grc_adm_kretaclp_32635/grc_adm0_kretaclp_32635"
107
108     return_code = subprocess.call('rm ' + shpfile + '.shp', shell=True)
109     return_code = subprocess.call('rm ' + shpfile + '_clp.shp', shell=True)
110     return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + shpfile + '.↩
            shp' + ' ' + shpfile + '.gml', shell=True)
111     return_code = subprocess.call('ogr2ogr -clipsrc ' + shpfileclip + '.shp ' +↩
             shpfile + '_clp.shp ' + shpfile + '.shp', shell=True)
112     geofile = osgeo.ogr.Open(shpfile + "_clp.shp")
113
114
115     layer = geofile.GetLayer(0)
116     for h in range(layer.GetFeatureCount()):
117       featureItem = layer.GetFeature(h)
118       geometry = featureItem.GetGeometryRef()
119       feature = shapely.wkt.loads(geometry.ExportToWkt())
120       if int(featureItem.GetField("DN")) == 1:
121         ingml3_area_verylow = ingml3_area_verylow + feature.area / 1000000
122       if int(featureItem.GetField("DN")) == 2:
123         ingml3_area_low = ingml3_area_low + feature.area / 1000000
124       if int(featureItem.GetField("DN")) == 3:
125         ingml3_area_moderate = ingml3_area_moderate + feature.area / 1000000
126       if int(featureItem.GetField("DN")) == 4:
127         ingml3_area_high = ingml3_area_high + feature.area / 1000000
```

```
128         if int(featureItem.GetField("DN")) == 5:
129           ingml3_area_veryhigh = ingml3_area_veryhigh + feature.area / 1000000
130         if int(featureItem.GetField("DN")) == 6:
131           ingml3_area_extreme = ingml3_area_extreme + feature.area / 1000000
132
133
134
135     ingml3_area_verylow_total = ingml3_area_verylow_total + ingml3_area_verylow
136     ingml3_area_low_total = ingml3_area_low_total + ingml3_area_low
137     ingml3_area_moderate_total = ingml3_area_moderate_total + ↩
            ingml3_area_moderate
138     ingml3_area_high_total = ingml3_area_high_total + ingml3_area_high
139     ingml3_area_veryhigh_total = ingml3_area_veryhigh_total + ↩
            ingml3_area_veryhigh
140     ingml3_area_extreme_total = ingml3_area_extreme_total + ingml3_area_extreme
141
142
143     print dt_year_str + dt_month_out_str + dt_day_out_str + ": " + "Average: " + ↩
            str((ingml3_area_verylow_total + ingml3_area_low_total + ↩
            ingml3_area_moderate_total + ingml3_area_high_total + ↩
            ingml3_area_veryhigh_total + ingml3_area_extreme_total) / days)
144
145     if index.lower() <> "fwi":
146       outcsv.write(dt_year_str + dt_month_out_str + dt_day_out_str + "\t" + str(↩
              ingml3_area_verylow) + "\t" + str(ingml3_area_low) + "\t" + str(↩
              ingml3_area_moderate) + "\t" + str(ingml3_area_high) + "\t" + str(↩
              ingml3_area_veryhigh + ingml3_area_extreme) + "\t" + str(↩
              ingml3_area_verylow + ingml3_area_low + ingml3_area_moderate + ↩
              ingml3_area_high + ingml3_area_veryhigh + ingml3_area_extreme) + "\n")
147     else:
148       outcsv.write(dt_year_str + dt_month_out_str + dt_day_out_str + "\t" + str(↩
              ingml3_area_verylow) + "\t" + str(ingml3_area_low) + "\t" + str(↩
              ingml3_area_moderate) + "\t" + str(ingml3_area_high) + "\t" + str(↩
              ingml3_area_veryhigh) + "\t" + str(ingml3_area_extreme) + "\t" + str(↩
              ingml3_area_verylow + ingml3_area_low + ingml3_area_moderate + ↩
              ingml3_area_high + ingml3_area_veryhigh + ingml3_area_extreme) + "\n")
149
```

```
150     dt = dt + timedelta(1)
151
152 #print "Average: " + str(ingml3_area_verylow_total + ingml3_area_low_total + ↩
        ingml3_area_moderate_total + ingml3_area_high_total + ↩
        ingml3_area_veryhigh_total + ingml3_area_extreme_total)
153
154 if index.lower() <> "fwi":
155     outcsv.write("\n")
156     outcsv.write(datasource + "\t" + str(ingml3_area_verylow_total/days) + "\t" +↩
           str(ingml3_area_low_total/days) + "\t" + str(ingml3_area_moderate_total/↩
          days) + "\t" + str(ingml3_area_high_total/days) + "\t" + str(↩
          ingml3_area_veryhigh_total/days + ingml3_area_extreme_total/days) + "\t" ↩
          + str(ingml3_area_verylow_total/days + ingml3_area_low_total/days + ↩
          ingml3_area_moderate_total/days + ingml3_area_high_total/days + ↩
          ingml3_area_veryhigh_total/days + ingml3_area_extreme_total/days) + "\n")
157 else:
158     outcsv.write("\n")
159     outcsv.write(datasource + "\t" + str(ingml3_area_verylow_total/days) + "\t" +↩
           str(ingml3_area_low_total/days) + "\t" + str(ingml3_area_moderate_total/↩
          days) + "\t" + str(ingml3_area_high_total/days) + "\t" + str(↩
          ingml3_area_veryhigh_total/days) + "\t" + str(ingml3_area_extreme_total/↩
          days) + "\t" + str(ingml3_area_verylow_total/days + ingml3_area_low_total↩
          /days + ingml3_area_moderate_total/days + ingml3_area_high_total/days + ↩
          ingml3_area_veryhigh_total/days + ingml3_area_extreme_total/days) + "\n")
```

A.1.9 wunderground_api_download.py

```
1  import urllib2
2  import json
3  import sys
4
5  import time as t
6  from datetime import datetime
7  from datetime import date
8  from datetime import timedelta
9  from datetime import *
10 import sys
11 import subprocess
12 import os
13
14
15 for j in range(1,len(sys.argv)):
16   if sys.argv[j][0:9].lower() == "stations=":
17     stations=sys.argv[j][9:].upper()
18   if sys.argv[j][0:7].lower() == "s_date=":
19     s_datestr=sys.argv[j][7:]
20   if sys.argv[j][0:7].lower() == "e_date=":
21     e_datestr=sys.argv[j][7:]
22   if sys.argv[j][0:9].lower() == "out_path=":
23     datapath=sys.argv[j][9:]
24   if sys.argv[j][0:10].lower() == "sleeptime=":
25     sleeptime=float(sys.argv[j][10:])
26   if sys.argv[j][0:8].lower() == "feature=":
27     feature=sys.argv[j][8:]
28   if sys.argv[j][0:4].lower() == "reg=":
29     reg=sys.argv[j][4:]
30
31 stationslist = stations.split(",")
32
33
34
35 #Startdate
```

```
36  s_year = int(s_datestr[0:4])
37  s_month = int(s_datestr[4:6])
38  s_day = int(s_datestr[6:8])
39
40  #Enddate
41  e_year = int(e_datestr[0:4])
42  e_month = int(e_datestr[4:6])
43  e_day = int(e_datestr[6:8])
44
45
46
47  today = datetime.now()
48  today_year_str = str(today.year)
49  today_month_str = str(today.month)
50  today_day_str = str(today.day)
51
52  if len(today_month_str) == 1:
53    today_month_str = "0" + today_month_str
54  if len(today_day_str) == 1:
55    today_day_str = "0" + today_day_str
56
57
58  start = datetime(s_year, s_month, s_day)
59  end = datetime(e_year, e_month, e_day)
60  dt = start
61
62
63  #LOOPS THROUGH EACH DAY IN THE GIVEN TIMESPAN
64  while dt <= end:
65
66    dt_year_str = str(dt.year)
67    dt_month_str = str(dt.month)
68    dt_day_str = str(dt.day)
69
70    if len(dt_month_str) == 1:
71      dt_month_str = "0" + dt_month_str
72    if len(dt_day_str) == 1:
```

```
73      dt_day_str = "0" + dt_day_str
74
75
76
77    dt = dt + timedelta(1)
78
79
80    if reg == "sard":
81
82      #Loop through all the input stations
83      for i in range(0, len(stationslist)):
84
85        if stationslist[i].upper() == "LIEA":
86          stationprefix = ""
87        if stationslist[i].upper() == "LIEB":
88          stationprefix = ""
89        if stationslist[i].upper() == "LIEC":
90          stationprefix = ""
91        if stationslist[i].upper() == "LIED":
92          stationprefix = ""
93        if stationslist[i].upper() == "LIEE":
94          stationprefix = ""
95        if stationslist[i].upper() == "LIEF":
96          stationprefix = ""
97        if stationslist[i].upper() == "LIEH":
98          stationprefix = ""
99        if stationslist[i].upper() == "LIEO":
100         stationprefix = ""
101       if stationslist[i].upper() == "LIER":
102         stationprefix = ""
103       if stationslist[i].upper() == "LIET":
104         stationprefix = ""
105       if stationslist[i].upper() == "IORISTAN2":
106         stationprefix = "pws:"
107       if stationslist[i].upper() == "IORISTAN4":
108         stationprefix = "pws:"
109       if stationslist[i].upper() == "IORISTAN5":
```

```
110         stationprefix = "pws:"
111       if stationslist[i].upper() == "ISARDEGN5":
112         stationprefix = "pws:"
113       if stationslist[i].upper() == "ISARDEGN6":
114         stationprefix = "pws:"
115       if stationslist[i].upper() == "ISARDEGN8":
116         stationprefix = "pws:"
117       if stationslist[i].upper() == "ISARDEGN9":
118         stationprefix = "pws:"
119       if stationslist[i].upper() == "ISARDEGN10":
120         stationprefix = "pws:"
121       if stationslist[i].upper() == "ISARDEGN15":
122         stationprefix = "pws:"
123       if stationslist[i].upper() == "ISARDEGN17":
124         stationprefix = "pws:"
125       if stationslist[i].upper() == "ISARDEGN22":
126         stationprefix = "pws:"
127       if stationslist[i].upper() == "ISARDEGN27":
128         stationprefix = "pws:"
129       if stationslist[i].upper() == "ISSPORTO1":
130         stationprefix = "pws:"
131
132       #if station_nr == 23:
133       # station = "LFKF"
134       # stationprefix = ""
135       #if station_nr == 24:
136       # station = "ICISANGI2"
137       # stationprefix = "pws:"
138
139
140       if feature == "history":
141         error = False
142         while True:
143           try:
144             datestr = dt_year_str + dt_month_str + dt_day_str
145             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/history_" + datestr + "/q/" + stationprefix + ↩
```

```
                stationslist[i].upper() + ".json", None, 60)
146             error = False
147             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
148             outfile = open(fw, 'w')
149             outfile.write(f.read())
150             outfile.close()
151             print fw
152             t.sleep(sleeptime)
153           except:
154             error = True
155             t.sleep(sleeptime)
156
157           if error == False:
158             break
159
160       if feature == "conditions":
161         error = False
162         while True:
163           try:
164             datestr = today_year_str + today_month_str + today_day_str
165             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/conditions/q/" + stationprefix + stationslist[i↩
                    ].upper() + ".json", None, 60)
166             error = False
167             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
168             outfile = open(fw, 'w')
169             outfile.write(f.read())
170             outfile.close()
171             print fw
172             t.sleep(sleeptime)
173           except:
174             error = True
175             t.sleep(sleeptime)
```

```
176
177           if error == False:
178             break
179
180
181
182       if feature == "forecast":
183         error = False
184         while True:
185           try:
186             datestr = today_year_str + today_month_str + today_day_str
187             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/forecast/q/" + stationprefix + stationslist[i].↩
                    upper() + ".json", None, 60)
188             error = False
189             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
190             outfile = open(fw, 'w')
191             outfile.write(f.read())
192             outfile.close()
193             print fw
194             t.sleep(sleeptime)
195           except:
196             error = True
197             t.sleep(sleeptime)
198
199           if error == False:
200             break
201
202
203       if feature == "forecast10day":
204         error = False
205         while True:
206           try:
207             datestr = today_year_str + today_month_str + today_day_str
208             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
```

```
                c0475dd9753a49a/forecast10day/q/" + stationprefix + ↩
                stationslist[i].upper() + ".json", None, 60)
209           error = False
210           fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                 "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                "
211           outfile = open(fw, 'w')
212           outfile.write(f.read())
213           outfile.close()
214           print fw
215           t.sleep(sleeptime)
216         except:
217           error = True
218           t.sleep(sleeptime)
219
220         if error == False:
221           break
222
223
224
225
226
227   if reg == "kreta":
228
229     #LOOPS THROUGH ALL THE STATIONS IN THE GIVEN COUNTRY
230     for i in range(0, len(stationslist)):
231
232
233       if stationslist[i].upper() == "LGIR":
234         stationprefix = ""
235       if stationslist[i].upper() == "LGSA":
236         stationprefix = ""
237       if stationslist[i].upper() == "LGST":
238         stationprefix = ""
239       if stationslist[i].upper() == "LGTL":
240         stationprefix = ""
241       if stationslist[i].upper() == "ICHANIAC2":
```

```
242         stationprefix = "pws:"
243       if stationslist[i].upper() == "IHERAKLI1":
244         stationprefix = "pws:"
245       if stationslist[i].upper() == "IHERAKLI3":
246         stationprefix = "pws:"
247       if stationslist[i].upper() == "IHERAKLI4":
248         stationprefix = "pws:"
249       if stationslist[i].upper() == "ILASITHI2":
250         stationprefix = "pws:"
251
252
253       if feature == "history":
254         error = False
255         while True:
256           try:
257             datestr = dt_year_str + dt_month_str + dt_day_str
258             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/history_" + datestr + "/q/" + stationprefix + ↩
                    stationslist[i].upper() + ".json", None, 60)
259             error = False
260             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
261             outfile = open(fw, 'w')
262             outfile.write(f.read())
263             outfile.close()
264             print fw
265             t.sleep(sleeptime)
266           except:
267             error = True
268             t.sleep(sleeptime)
269
270           if error == False:
271             break
272
273
274       if feature == "conditions":
```

```
275         error = False
276         while True:
277           try:
278             datestr = today_year_str + today_month_str + today_day_str
279             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/conditions/q/" + stationprefix + stationslist[i↩
                    ].upper() + ".json", None, 60)
280             error = False
281             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
282             outfile = open(fw, 'w')
283             outfile.write(f.read())
284             outfile.close()
285             print fw
286             t.sleep(sleeptime)
287           except:
288             error = True
289             t.sleep(sleeptime)
290
291           if error == False:
292             break
293
294
295       if feature == "forecast":
296         error = False
297         while True:
298           try:
299             datestr = today_year_str + today_month_str + today_day_str
300             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/forecast/q/" + stationprefix + stationslist[i].↩
                    upper() + ".json", None, 60)
301             error = False
302             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
303             outfile = open(fw, 'w')
```

```
304             outfile.write(f.read())
305             outfile.close()
306             print fw
307             t.sleep(sleeptime)
308           except:
309             error = True
310             t.sleep(sleeptime)
311
312           if error == False:
313             break
314
315
316       if feature == "forecast10day":
317         error = False
318         while True:
319           try:
320             datestr = today_year_str + today_month_str + today_day_str
321             f = urllib2.urlopen("http://api.wunderground.com/api/6↩
                    c0475dd9753a49a/forecast10day/q/" + stationprefix + ↩
                    stationslist[i].upper() + ".json", None, 60)
322             error = False
323             fw = datapath + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr +↩
                     "_" + stationslist[i].lower() + "_" + feature.lower() + ".json↩
                    "
324             outfile = open(fw, 'w')
325             outfile.write(f.read())
326             outfile.close()
327             print fw
328             t.sleep(sleeptime)
329           except:
330             error = True
331             t.sleep(sleeptime)
332
333           if error == False:
334             break
335
336
```

```
337 sys.exit()
```

A.1.10 wunderground_api_extract_history.py

```
import urllib2
import json
import sys
from datetime import datetime


observations_hour = []
observations_min = []
observations_tempm = []
observations_hum = []
observations_wspdm = []


for j in range(1,len(sys.argv)):
  if sys.argv[j][0:9].lower() == "stations=":
    stations=sys.argv[j][9:].upper()
  if sys.argv[j][0:5].lower() == "date=":
    datestr=sys.argv[j][5:]
  if sys.argv[j][0:8].lower() == "in_path=":
    in_path=sys.argv[j][8:]
  if sys.argv[j][0:9].lower() == "out_path=":
    out_path=sys.argv[j][9:]
  if sys.argv[j][0:12].lower() == "out_praefix=":
    out_praefix=sys.argv[j][12:]
  if sys.argv[j][0:11].lower() == "out_suffix=":
    out_suffix=sys.argv[j][11:]


stationslist = stations.split(",")
outfile = open(out_path + out_praefix + '_' + datestr + out_suffix + '.csv', 'w')
outfile.write("STATION,LAT,LON,HEIGHT,DATE,AVGTEMP_NN,PRECIP24ALL,TEMPNOON_NN,HUMIDNOON,WINDKMHNOON" + "\n")


```

```
34 for i in range(0, len(stationslist)):
35   #print stationslist[i]
36
37   if stationslist[i].upper() == "IORISTAN5":
38     x_coord = 495381.91
39     y_coord = 4413441.04
40     height_m = 233.17
41
42   if stationslist[i].upper() == "ISARDEGN15":
43     x_coord = 478230.50
44     y_coord = 4485728.46
45     height_m = 487.68
46
47   if stationslist[i].upper() == "ISSPORTO1":
48     x_coord = 449669.28
49     y_coord = 4521056.26
50     height_m = 2.74
51
52   if stationslist[i].upper() == "ISARDEGN6":
53     x_coord = 511566.46
54     y_coord = 4341754.11
55     height_m = 12.80
56
57   if stationslist[i].upper() == "IORISTAN2":
58     x_coord = 465213.99
59     y_coord = 4418069.43
60     height_m = 3.66
61
62   if stationslist[i].upper() == "ISARDEGN27":
63     x_coord = 500253.89
64     y_coord = 4492356.79
65     height_m = 402.95
66
67   if stationslist[i].upper() == "ISARDEGN8":
68     x_coord = 516138.96
69     y_coord = 4342871.97
70     height_m = 26.82
```

```
71
72    if stationslist[i].upper() == "ISARDEGN9":
73      x_coord = 546552.75
74      y_coord = 4366078.27
75      height_m = 14.94
76
77    if stationslist[i].upper() == "ISARDEGN5":
78      x_coord = 506552.44
79      y_coord = 4350848.36
80      height_m = 55.78
81
82    if stationslist[i].upper() == "IORISTAN4":
83      x_coord = 468147.33
84      y_coord = 4424605.12
85      height_m = 28.96
86
87    if stationslist[i].upper() == "ISARDEGN17":
88      x_coord = 458276.35
89      y_coord = 4335753.02
90      height_m = 99.97
91
92    if stationslist[i].upper() == "ISARDEGN22":
93      x_coord = 518948.20
94      y_coord = 4428446.76
95      height_m = 868.68
96
97    if stationslist[i].upper() == "ISARDEGN10":
98      x_coord = 469669.39
99      y_coord = 4377429.13
100     height_m = 99.06
101
102   if stationslist[i].upper() == "LIEA":
103     x_coord = 490359.17
104     y_coord = 4497813.22
105     height_m = 22.86
106
107   if stationslist[i].upper() == "LIEB":
```

```
108       x_coord = 560940.32
109       y_coord = 4420309.10
110       height_m = 149.96
111
112     if stationslist[i].upper() == "LIEC":
113       x_coord = 544399.88
114       y_coord = 4328432.07
115       height_m = 117.96
116
117     if stationslist[i].upper() == "LIED":
118       x_coord = 497199.53
119       y_coord = 4355185.32
120       height_m = 28.04
121
122     if stationslist[i].upper() == "LIEE":
123       x_coord = 505203.55
124       y_coord = 4343789.24
125       height_m = 3.96
126
127     if stationslist[i].upper() == "LIEF":
128       x_coord = 453688.38
129       y_coord = 4399028.96
130       height_m = 95.10
131
132     if stationslist[i].upper() == "LIEH":
133       x_coord = 429150.22
134       y_coord = 4490373.34
135       height_m = 203.91
136
137     if stationslist[i].upper() == "LIEO":
138       x_coord = 543614.53
139       y_coord = 4527629.83
140       height_m = 13.11
141
142     if stationslist[i].upper() == "LIER":
143       x_coord = 469488.48
144       y_coord = 4416164.30
```

```
145       height_m = 10.97
146
147     if stationslist[i].upper() == "LIET":
148       x_coord = 558369.92
149       y_coord = 4418968.13
150       height_m = 7.01
151
152
153     if stationslist[i].upper() == "ICHANIAC2":
154       x_coord = 234336.76
155       y_coord = 3935992.12
156       height_m = 121.92
157
158     if stationslist[i].upper() == "ILASITHI2":
159       x_coord = 406692.17
160       y_coord = 3877846.31
161       height_m = 0.00
162
163     if stationslist[i].upper() == "IHERAKLI1":
164       x_coord = 331592.91
165       y_coord = 3911769.63
166       height_m = 29.87
167
168     if stationslist[i].upper() == "IHERAKLI3":
169       x_coord = 327838.67
170       y_coord = 3910398.15
171       height_m = 10.97
172
173     if stationslist[i].upper() == "IHERAKLI4":
174       x_coord = 326492.13
175       y_coord = 3911311.70
176       height_m = 19.81
177
178     if stationslist[i].upper() == "LGSA":
179       x_coord = 238414.17
180       y_coord = 3930463.80
181       height_m = 150.88
```

```
182
183   if stationslist[i].upper() == "LGIR":
184     x_coord = 334629.25
185     y_coord = 3912234.84
186     height_m = 39.01
187
188   if stationslist[i].upper() == "LGTL":
189     x_coord = 347686.11
190     y_coord = 3895617.27
191     height_m = 335.89
192
193   if stationslist[i].upper() == "LGST":
194     x_coord = 418207.19
195     y_coord = 3897378.09
196     height_m = 28.04
197
198
199
200
201   f = open(in_path + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr + '_' + ↩
          stationslist[i].lower() + '_history.json', 'r')
202
203
204   json_string = f.read()
205   parsed_json = json.loads(json_string)
206
207
208   #Check if there's even data in the input file (otherwise everything is 'no ↩
          data')
209   data_available = True
210
211   try:
212     history_year = int(parsed_json['history']['date']['year'])
213     history_mon = int(parsed_json['history']['date']['mon'])
214     history_mday = int(parsed_json['history']['date']['mday'])
215     dailysummary_meantempm = float(parsed_json['history']['dailysummary'][0]['↩
            meantempm'])
```

```
216     dailysummary_precipm = float(parsed_json['history']['dailysummary'][0]['↩
            precipm'])
217   except:
218     data_available = False
219     dailysummary_meantempm = -999
220     dailysummary_meantempm_nn = -999
221     dailysummary_precipm = -999
222     noon_tempm = -999
223     noon_tempm_nn = -999
224     noon_hum = -999
225     noon_wspdm = -999
226
227
228   if data_available == True:
229     j = 0
230     #Read all measurements to lists (one list per column)
231     while True:
232
233       try:
234         parsed_json['history']['observations'][j]['tempm']
235       except:
236         break
237
238       observations_hour.append(parsed_json['history']['observations'][j]['↩
              utcdate']['hour'])
239       observations_min.append(parsed_json['history']['observations'][j]['↩
              utcdate']['min'])
240       observations_tempm.append(parsed_json['history']['observations'][j]['↩
              tempm'])
241       observations_hum.append(parsed_json['history']['observations'][j]['hum'])
242       observations_wspdm.append(parsed_json['history']['observations'][j]['↩
              wspdm'])
243       j = j + 1
244
245     f.close()
246
247
```

```
248     noon = '13:00:00'
249     #13:00:00 for Sardinia
250     #14:00:00 for Crete
251
252     min_beforenoon_tdelta_seconds = 86400
253     min_afternoon_tdelta_seconds = 0
254
255     #Loop through all the measurements, find the ones nearest before and after ↩
            noon, and determine their
256     #noon-time (UTC) values trough interpolation
257     for k in range(0,j):
258
259       #print str(observations_hour[j]) + ":" + str(observations_min[k]) + "    "↩
              + str(noon_tdelta.seconds) + "      tempm:" + str(observations_tempm[↩
             k]) + "   hum:" + str(observations_hum[k]) + "   wspdm:" + str(↩
             observations_wspdm[k])
260
261       obstime = observations_hour[k] + ":" + observations_min[k] + ':00'
262       noon_tdelta = datetime.strptime('13:00:00', '%H:%M:%S') - datetime.↩
              strptime(obstime, '%H:%M:%S')
263       noon_tdelta_seconds = int(noon_tdelta.seconds)
264
265       if noon_tdelta_seconds < min_beforenoon_tdelta_seconds:
266         min_beforenoon_tdelta_seconds = noon_tdelta_seconds
267         min_beforenoon_tempm = float(observations_tempm[k])
268         min_beforenoon_hum = float(observations_hum[k])
269         min_beforenoon_wspdm = float(observations_wspdm[k])
270
271       if noon_tdelta_seconds > min_afternoon_tdelta_seconds:
272         min_afternoon_tdelta_seconds = noon_tdelta_seconds
273         min_afternoon_tempm = float(observations_tempm[k])
274         min_afternoon_hum = float(observations_hum[k])
275         min_afternoon_wspdm = float(observations_wspdm[k])
276
277     #print "before noon: " + str(min_beforenoon_tdelta_seconds) + "   " + str(↩
            min_beforenoon_tempm) + "   " + str(min_beforenoon_hum) + "   " + str(↩
            min_beforenoon_wspdm)
```

```
278     #print "after noon: " + str(86400 - int(min_afternoon_tdelta_seconds)) + " "
            + str(min_afternoon_tempm) + "    " + str(min_afternoon_hum) + "    "
            + str(min_afternoon_wspdm)
279
280     dif_seconds = min_beforenoon_tdelta_seconds + (86400 - int(
            min_afternoon_tdelta_seconds))
281     noon_tempm = round(float(((abs(min_beforenoon_tempm - min_afternoon_tempm)
            / dif_seconds) * min_beforenoon_tdelta_seconds) + min_beforenoon_tempm)
            , 2)
282     noon_hum = round(float(((abs(min_beforenoon_hum - min_afternoon_hum) /
            dif_seconds) * min_beforenoon_tdelta_seconds) + min_beforenoon_hum), 2)
283     noon_wspdm = round(float(((abs(min_beforenoon_wspdm - min_afternoon_wspdm)
            / dif_seconds) * min_beforenoon_tdelta_seconds) + min_beforenoon_wspdm)
            , 2)
284
285
286
287     # Correct temperatures to equivalent sea level values
288     # Temperature lapse rate: 6.49 K / 1.000 m
289     dailysummary_meantempm_nn = -999
290     noon_tempm_nn = -999
291     lapserate_m = 0.00649
292
293     if str(dailysummary_meantempm) <> "" and str(dailysummary_meantempm) <> "
            -999":
294       dailysummary_meantempm_nn = round(dailysummary_meantempm + (height_m *
              lapserate_m), 2)
295     if str(noon_tempm) <> "" and str(noon_tempm) <> "-999":
296       noon_tempm_nn = round(noon_tempm + (height_m * lapserate_m), 2)
297
298   dailysummary_meantempm_nn_str = str(dailysummary_meantempm_nn)
299   noon_tempm_nn_str = str(noon_tempm_nn)
300   dailysummary_precipm_str = str(dailysummary_precipm)
301   noon_hum_str = str(noon_hum)
302   noon_wspdm_str = str(noon_wspdm)
303
304   #Check for empty values, replace by no data (-999)
```

```
305   if str(dailysummary_meantempm) == "":
306     dailysummary_meantempm_str = "-999"
307   if str(dailysummary_precipm) == "":
308     dailysummary_precipm_str = "-999"
309   if str(noon_tempm) == "":
310     noon_tempm_str = "-999"
311   if str(noon_hum) == "":
312     noon_hum_str = "-999"
313   if str(noon_wspdm) == "":
314     noon_wspdm_str = "-999"
315
316   outfile.write(stationslist[i].upper() + "," + str(y_coord) + "," + str(↩
          x_coord) + "," + str(height_m) + "," + datestr + "," + ↩
          dailysummary_meantempm_nn_str +"," + dailysummary_precipm_str + "," + ↩
          noon_tempm_nn_str + "," + noon_hum_str + "," + noon_wspdm_str + "\n")
317
318 outfile.close()
```

A.1.11 wunderground_api_extract_forecast10day.py

```
1  import urllib2
2  import json
3  import sys
4  import time
5  import datetime
6
7  observations_hour = []
8  observations_min = []
9  observations_tempm = []
10 observations_hum = []
11 observations_wspdm = []
12 outfile = []
13 datestr_forecast = []
14
15 for j in range(1,len(sys.argv)):
16   if sys.argv[j][0:9].lower() == "stations=":
17     stations=sys.argv[j][9:].upper()
18   if sys.argv[j][0:5].lower() == "date=":
19     datestr=sys.argv[j][5:]
20   if sys.argv[j][0:8].lower() == "in_path=":
21     in_path=sys.argv[j][8:]
22   if sys.argv[j][0:9].lower() == "out_path=":
23     out_path=sys.argv[j][9:]
24   if sys.argv[j][0:12].lower() == "out_praefix=":
25     out_praefix=sys.argv[j][12:]
26   if sys.argv[j][0:11].lower() == "out_suffix=":
27     out_suffix=sys.argv[j][11:]
28
29
30 #Determine the dates for the nect 10 days in save in list 'datestr_forecast'
31 dateobj = datetime.date(int(datestr[0:4]), int(datestr[4:6]), int(datestr[6:8])↩
       )
32 for j in range(0,10):
33   datestr_forecast.append(str(dateobj + datetime.timedelta(days=j)).replace("-"↩
         ,""))
```

```
34
35
36 #Open 10 output-files , each with a different date (the input date plus x days ↩
       in the future)
37 for j in range(0,10):
38   outfile.append(open(out_path + out_praefix + '_' + datestr_forecast[j] + ↩
         out_suffix + '.csv', 'w'))
39   outfile[j].write("STATION,LAT,LON,HEIGHT,DATE,AVGTEMP_NN,PRECIP24ALL,↩
         TEMPNOON_NN,HUMIDNOON,WINDKMHNOON" + "\n")
40
41
42 stationslist = stations.split(",")
43
44 #Loop through all the input stations
45 for i in range(0, len(stationslist)):
46   #print stationslist[i]
47
48   if stationslist[i].upper() == "IORISTAN5":
49     x_coord = 495381.91
50     y_coord = 4413441.04
51     height_m = 233.17
52
53   if stationslist[i].upper() == "ISARDEGN15":
54     x_coord = 478230.50
55     y_coord = 4485728.46
56     height_m = 487.68
57
58   if stationslist[i].upper() == "ISSPORTO1":
59     x_coord = 449669.28
60     y_coord = 4521056.26
61     height_m = 2.74
62
63   if stationslist[i].upper() == "ISARDEGN6":
64     x_coord = 511566.46
65     y_coord = 4341754.11
66     height_m = 12.80
67
```

```
68    if stationslist[i].upper() == "IORISTAN2":
69      x_coord = 465213.99
70      y_coord = 4418069.43
71      height_m = 3.66
72
73    if stationslist[i].upper() == "ISARDEGN27":
74      x_coord = 500253.89
75      y_coord = 4492356.79
76      height_m = 402.95
77
78    if stationslist[i].upper() == "ISARDEGN8":
79      x_coord = 516138.96
80      y_coord = 4342871.97
81      height_m = 26.82
82
83    if stationslist[i].upper() == "ISARDEGN9":
84      x_coord = 546552.75
85      y_coord = 4366078.27
86      height_m = 14.94
87
88    if stationslist[i].upper() == "ISARDEGN5":
89      x_coord = 506552.44
90      y_coord = 4350848.36
91      height_m = 55.78
92
93    if stationslist[i].upper() == "IORISTAN4":
94      x_coord = 468147.33
95      y_coord = 4424605.12
96      height_m = 28.96
97
98    if stationslist[i].upper() == "ISARDEGN17":
99      x_coord = 458276.35
100     y_coord = 4335753.02
101     height_m = 99.97
102
103   if stationslist[i].upper() == "ISARDEGN22":
104     x_coord = 518948.20
```

```
105       y_coord = 4428446.76
106       height_m = 868.68
107
108     if stationslist[i].upper() == "ISARDEGN10":
109       x_coord = 469669.39
110       y_coord = 4377429.13
111       height_m = 99.06
112
113     if stationslist[i].upper() == "LIEA":
114       x_coord = 490359.17
115       y_coord = 4497813.22
116       height_m = 22.86
117
118     if stationslist[i].upper() == "LIEB":
119       x_coord = 560940.32
120       y_coord = 4420309.10
121       height_m = 149.96
122
123     if stationslist[i].upper() == "LIEC":
124       x_coord = 544399.88
125       y_coord = 4328432.07
126       height_m = 117.96
127
128     if stationslist[i].upper() == "LIED":
129       x_coord = 497199.53
130       y_coord = 4355185.32
131       height_m = 28.04
132
133     if stationslist[i].upper() == "LIEE":
134       x_coord = 505203.55
135       y_coord = 4343789.24
136       height_m = 3.96
137
138     if stationslist[i].upper() == "LIEF":
139       x_coord = 453688.38
140       y_coord = 4399028.96
141       height_m = 95.10
```

```
142
143   if stationslist[i].upper() == "LIEH":
144     x_coord = 429150.22
145     y_coord = 4490373.34
146     height_m = 203.91
147
148   if stationslist[i].upper() == "LIEO":
149     x_coord = 543614.53
150     y_coord = 4527629.83
151     height_m = 13.11
152
153   if stationslist[i].upper() == "LIER":
154     x_coord = 469488.48
155     y_coord = 4416164.30
156     height_m = 10.97
157
158   if stationslist[i].upper() == "LIET":
159     x_coord = 558369.92
160     y_coord = 4418968.13
161     height_m = 7.01
162
163
164   if stationslist[i].upper() == "ICHANIAC2":
165     x_coord = 234336.76
166     y_coord = 3935992.12
167     height_m = 121.92
168
169   if stationslist[i].upper() == "ILASITHI2":
170     x_coord = 406692.17
171     y_coord = 3877846.31
172     height_m = 0.00
173
174   if stationslist[i].upper() == "IHERAKLI1":
175     x_coord = 331592.91
176     y_coord = 3911769.63
177     height_m = 29.87
178
```

```
179   if stationslist[i].upper() == "IHERAKLI3":
180     x_coord = 327838.67
181     y_coord = 3910398.15
182     height_m = 10.97
183
184   if stationslist[i].upper() == "IHERAKLI4":
185     x_coord = 326492.13
186     y_coord = 3911311.70
187     height_m = 19.81
188
189   if stationslist[i].upper() == "LGSA":
190     x_coord = 238414.17
191     y_coord = 3930463.80
192     height_m = 150.88
193
194   if stationslist[i].upper() == "LGIR":
195     x_coord = 334629.25
196     y_coord = 3912234.84
197     height_m = 39.01
198
199   if stationslist[i].upper() == "LGTL":
200     x_coord = 347686.11
201     y_coord = 3895617.27
202     height_m = 335.89
203
204   if stationslist[i].upper() == "LGST":
205     x_coord = 418207.19
206     y_coord = 3897378.09
207     height_m = 28.04
208
209
210
211   #open one 10day forecast for a specific input station
212   f = open(in_path + datestr[0:4] + "/" + datestr[4:6] + "/" + datestr + '_' + ↩
          stationslist[i].lower() + '_forecast10day.json', 'r')
213
214
```

```
215   json_string = f.read()
216   parsed_json = json.loads(json_string)
217
218   #read out all the values for the ten days and write each day's values into ↩
          one of the ten output files
219   for k in range(0,10):
220     #Check if there's even data in the input file (otherwise everything is 'no ↩
            data')
221     data_available = True
222
223     try:
224       forecast_year = int(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['date']['year'])
225       forecast_month = int(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['date']['month'])
226       forecast_day = int(parsed_json['forecast']['simpleforecast']['forecastday↩
              '][k]['date']['day'])
227     except:
228       data_available = False
229       dailysummary_meantempm = -999
230       dailysummary_precipm = -999
231       noon_tempm = -999
232       noon_hum = -999
233       noon_wspdm = -999
234
235     if data_available == True:
236       #Extremwerte statt Durchschnitt verwenden
237       forecast_high_celsius = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['high']['celsius'])
238       #forecast_low_celsius = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['low']['celsius'])
239       forecast_qpfallday_mm = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['qpf_allday']['mm'])
240       forecast_low_celsius = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['high']['celsius'])
241       #forecast_avewind_kph = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['avewind']['kph'])
```

```
242       #forecast_avehumidity = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['avehumidity'])
243       forecast_avewind_kph = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['maxwind']['kph'])
244       forecast_avehumidity = float(parsed_json['forecast']['simpleforecast']['↩
              forecastday'][k]['minhumidity'])
245
246
247
248
249       dailysummary_meantempm = (forecast_high_celsius + forecast_low_celsius) /↩
              2
250       dailysummary_precipm = forecast_qpfallday_mm
251       noon_tempm = (forecast_high_celsius + forecast_low_celsius) / 2
252       noon_hum = forecast_avehumidity
253       noon_wspdm = forecast_avewind_kph
254
255
256     # Correct temperatures to equivalent sea level values
257     # Temperature lapse rate: 6.49 K / 1.000 m
258     dailysummary_meantempm_nn = -999
259     noon_tempm_nn = -999
260     lapserate_m = 0.00649
261
262     if str(dailysummary_meantempm) <> "" and str(dailysummary_meantempm) <> "↩
            -999":
263       dailysummary_meantempm_nn = round(dailysummary_meantempm + (height_m * ↩
              lapserate_m), 2)
264     if str(noon_tempm) <> "" and str(noon_tempm) <> "-999":
265       noon_tempm_nn = round(noon_tempm + (height_m * lapserate_m), 2)
266
267     dailysummary_meantempm_nn_str = str(dailysummary_meantempm_nn)
268     noon_tempm_nn_str = str(noon_tempm_nn)
269     dailysummary_precipm_str = str(dailysummary_precipm)
270     noon_hum_str = str(noon_hum)
271     noon_wspdm_str = str(noon_wspdm)
272
```

```
273     #Check for empty values, replace by no data (-999)
274     if str(dailysummary_meantempm) == "":
275       dailysummary_meantempm_str = "-999"
276     if str(dailysummary_precipm) == "":
277       dailysummary_precipm_str = "-999"
278     if str(noon_tempm) == "":
279       noon_tempm_str = "-999"
280     if str(noon_hum) == "":
281       noon_hum_str = "-999"
282     if str(noon_wspdm) == "":
283       noon_wspdm_str = "-999"
284
285
286     #print dailysummary_meantempm
287     #print dailysummary_precipm
288     #print noon_tempm
289     #print noon_hum
290     #print noon_wspdm
291
292     #write the values into the actual one of the ten output files
293     outfile[k].write(stationslist[i].upper() + "," + str(y_coord) + "," + str(↩
            x_coord) + "," + str(height_m) + "," + datestr + "," + ↩
            dailysummary_meantempm_nn_str +"," + dailysummary_precipm_str + "," + ↩
            noon_tempm_nn_str + "," + noon_hum_str + "," + noon_wspdm_str + "\n")
294
295   #close the input file
296   f.close()
297
298 #close all the output files
299 for j in range(0,10):
300   outfile[j].close()
```

A.1.12 wunderground_comb.py

```
1  import sys
2  import time
3  from datetime import datetime
4  from datetime import date
5  from datetime import timedelta
6  import subprocess
7  import os
8  import os.path
9
10
11 in_praefix = ""
12 for j in range(1,len(sys.argv)):
13   if sys.argv[j][0:9].lower() == "stations=":
14     stations=sys.argv[j][9:].upper()
15   if sys.argv[j][0:7].lower() == "s_date=":
16     s_date_str=sys.argv[j][7:]
17   if sys.argv[j][0:7].lower() == "e_date=":
18     e_date_str=sys.argv[j][7:]
19   if sys.argv[j][0:8].lower() == "in_path=":
20     in_path=sys.argv[j][8:]
21   if sys.argv[j][0:11].lower() == "in_praefix=":
22     in_praefix=sys.argv[j][11:]
23   if sys.argv[j][0:9].lower() == "out_path=":
24     out_path=sys.argv[j][9:]
25   if sys.argv[j][0:12].lower() == "out_praefix=":
26     out_praefix=sys.argv[j][12:]
27   if sys.argv[j][0:11].lower() == "out_suffix=":
28     out_suffix=sys.argv[j][11:]
29
30
31 stationslist = stations.split(",")
32 infilelist = []
33 rlinelist = []
34 for i in range(0, len(stationslist)):
35   infilelist.append("")
```

```
36    rlinelist.append("")
37
38  for i in range(0, len(stationslist)):
39    infilelist[i] = open(in_path + in_praefix + stationslist[i] + "_" + ↩
          s_date_str + "_" + e_date_str + ".csv", 'r')
40
41
42  while True:
43
44    for i in range(0, len(stationslist)):
45      rlinelist[i] = infilelist[i].readline()
46
47      tmpentries = rlinelist[i].split(",")
48
49      if len(rlinelist[i]) == 0:
50        sys.exit()
51
52
53    outfile=open(out_path + out_praefix + "_" + tmpentries[0] + out_suffix + ".↩
          csv", 'w')
54    outfile.write("STATION,LAT,LON,HEIGHT,DATE,AVGTEMP_NN,PRECIP24ALL,TEMPNOON_NN↩
          ,HUMIDNOON,WINDKMHNOON" + "\n")
55
56    for i in range(0, len(stationslist)):
57
58      if stationslist[i].upper() == "IORISTAN5":
59        stationtype = "pws"
60        x_coord = 495381.91
61        y_coord = 4413441.04
62        height_m = 233.17
63
64      if stationslist[i].upper() == "ISARDEGN15":
65        stationtype = "pws"
66        x_coord = 478230.50
67        y_coord = 4485728.46
68        height_m = 487.68
69
```

```
70      if stationslist[i].upper() == "ISSPORTO1":
71        stationtype = "pws"
72        x_coord = 449669.28
73        y_coord = 4521056.26
74        height_m = 2.74
75
76      if stationslist[i].upper() == "ISARDEGN6":
77        stationtype = "pws"
78        x_coord = 511566.46
79        y_coord = 4341754.11
80        height_m = 12.80
81
82      if stationslist[i].upper() == "ISARDEGN32":
83        stationtype = "pws"
84        x_coord = 458873.55
85        y_coord = 4350732.09
86        height_m = 225.86
87
88      if stationslist[i].upper() == "ISARDEGN35":
89        stationtype = "pws"
90        x_coord = 458536.94
91        y_coord = 4352287.61
92        height_m = 294.13
93
94      if stationslist[i].upper() == "IORISTAN2":
95        stationtype = "pws"
96        x_coord = 465213.99
97        y_coord = 4418069.43
98        height_m = 3.66
99
100     if stationslist[i].upper() == "ISARDEGN39":
101       stationtype = "pws"
102       x_coord = 559165.71
103       y_coord = 4470057.04
104       height_m = 14.94
105
106     if stationslist[i].upper() == "ISARDEGN27":
```

```
107         stationtype = "pws"
108         x_coord = 500253.89
109         y_coord = 4492356.79
110         height_m = 402.95
111
112       if stationslist[i].upper() == "IOGLIAST2":
113         stationtype = "pws"
114         x_coord = 537647.85
115         y_coord = 4392222.44
116         height_m = 611.12
117
118       if stationslist[i].upper() == "ISARDEGN8":
119         stationtype = "pws"
120         x_coord = 516138.96
121         y_coord = 4342871.97
122         height_m = 26.82
123
124       if stationslist[i].upper() == "ICISANGI2":
125         stationtype = "pws"
126         x_coord = 456267.84
127         y_coord = 4331879.73
128         height_m = 10.36
129
130       if stationslist[i].upper() == "ISARDEGN9":
131         stationtype = "pws"
132         x_coord = 546552.75
133         y_coord = 4366078.27
134         height_m = 14.94
135
136       if stationslist[i].upper() == "ISARDEGN38":
137         stationtype = "pws"
138         x_coord = 546275.15
139         y_coord = 4340328.74
140         height_m = 49.07
141
142       if stationslist[i].upper() == "IITALIAS2":
143         stationtype = "pws"
```

```
144         x_coord = 461336.26
145         y_coord = 4511106.17
146         height_m = 142.04
147
148       if stationslist[i].upper() == "ISARDEGN5":
149         stationtype = "pws"
150         x_coord = 506552.44
151         y_coord = 4350848.36
152         height_m = 55.78
153
154       if stationslist[i].upper() == "ICAGLIAR7":
155         stationtype = "pws"
156         x_coord = 515436.68
157         y_coord = 4348863.29
158         height_m = 75.90
159
160       if stationslist[i].upper() == "ICAGLIAR8":
161         stationtype = "pws"
162         x_coord = 515608.93
163         y_coord = 4348974.61
164         height_m = 79.86
165
166       if stationslist[i].upper() == "ICAGLIAR6":
167         stationtype = "pws"
168         x_coord = 516811.95
169         y_coord = 4350974.70
170         height_m = 152.40
171
172       if stationslist[i].upper() == "IORISTAN4":
173         stationtype = "pws"
174         x_coord = 468147.33
175         y_coord = 4424605.12
176         height_m = 28.96
177
178       if stationslist[i].upper() == "ISARDEGN17":
179         stationtype = "pws"
180         x_coord = 458276.35
```

```
181         y_coord = 4335753.02
182         height_m = 99.97
183 
184       if stationslist[i].upper() == "ISARDEGN22":
185         stationtype = "pws"
186         x_coord = 518948.20
187         y_coord = 4428446.76
188         height_m = 868.68
189 
190       if stationslist[i].upper() == "ISARDEGN40":
191         stationtype = "pws"
192         x_coord = 500172.30
193         y_coord = 4356838.39
194         height_m = 42.06
195 
196       if stationslist[i].upper() == "ISARDEGN10":
197         stationtype = "pws"
198         x_coord = 469669.39
199         y_coord = 4377429.13
200         height_m = 99.06
201 
202       if stationslist[i].upper() == "LIEA":
203         stationtype = "airport"
204         x_coord = 490359.17
205         y_coord = 4497813.22
206         height_m = 22.86
207 
208       if stationslist[i].upper() == "LIEB":
209         stationtype = "airport"
210         x_coord = 560940.32
211         y_coord = 4420309.10
212         height_m = 149.96
213 
214       if stationslist[i].upper() == "LIEC":
215         stationtype = "airport"
216         x_coord = 544399.88
217         y_coord = 4328432.07
```

```
218         height_m = 117.96
219
220       if stationslist[i].upper() == "LIED":
221         stationtype = "airport"
222         x_coord = 497199.53
223         y_coord = 4355185.32
224         height_m = 28.04
225
226       if stationslist[i].upper() == "LIEE":
227         stationtype = "airport"
228         x_coord = 505203.55
229         y_coord = 4343789.24
230         height_m = 3.96
231
232       if stationslist[i].upper() == "LIEF":
233         stationtype = "airport"
234         x_coord = 453688.38
235         y_coord = 4399028.96
236         height_m = 95.10
237
238       if stationslist[i].upper() == "LIEG":
239         stationtype = "airport"
240         x_coord = 533530.23
241         y_coord = 4562890.39
242         height_m = 159.11
243
244       if stationslist[i].upper() == "LIEH":
245         stationtype = "airport"
246         x_coord = 429150.22
247         y_coord = 4490373.34
248         height_m = 203.91
249
250       if stationslist[i].upper() == "LIEL":
251         stationtype = "airport"
252         x_coord = 554116.87
253         y_coord = 4372242.71
254         height_m = 4.88
```

```
255
256     if stationslist[i].upper() == "LIEN":
257       stationtype = "airport"
258       x_coord = 521303.56
259       y_coord = 4440739.86
260       height_m = 1029.00
261
262     if stationslist[i].upper() == "LIEO":
263       stationtype = "airport"
264       x_coord = 543614.53
265       y_coord = 4527629.83
266       height_m = 13.11
267
268     if stationslist[i].upper() == "LIEP":
269       stationtype = "airport"
270       x_coord = 538001.82
271       y_coord = 4391746.93
272       height_m = 605.94
273
274     if stationslist[i].upper() == "LIER":
275       stationtype = "airport"
276       x_coord = 469488.48
277       y_coord = 4416164.30
278       height_m = 10.97
279
280     if stationslist[i].upper() == "LIET":
281       stationtype = "airport"
282       x_coord = 558369.92
283       y_coord = 4418968.13
284       height_m = 7.01
285
286     if stationslist[i].upper() == "LFKF":
287       stationtype = "airport"
288       x_coord = 508162.58
289       y_coord = 4594335.45
290       height_m = 22.86
291
```

```
292
293
294
295
296
297     if stationslist[i].upper() == "IRETHYMN4":
298       stationtype = "pws"
299       x_coord = 263714.58
300       y_coord = 3896865.62
301       height_m = 60.96
302
303     if stationslist[i].upper() == "ICHANIAC2":
304       stationtype = "pws"
305       x_coord = 234336.76
306       y_coord = 3935992.12
307       height_m = 121.92
308
309     if stationslist[i].upper() == "ILASITHI2":
310       stationtype = "pws"
311       x_coord = 406692.17
312       y_coord = 3877846.31
313       height_m = 0.00
314
315     if stationslist[i].upper() == "IHERAKLI1":
316       stationtype = "pws"
317       x_coord = 331592.91
318       y_coord = 3911769.63
319       height_m = 29.87
320
321     if stationslist[i].upper() == "IHERAKLI3":
322       stationtype = "pws"
323       x_coord = 327838.67
324       y_coord = 3910398.15
325       height_m = 10.97
326
327     if stationslist[i].upper() == "IHERAKLI4":
328       stationtype = "pws"
```

```
329         x_coord = 326492.13
330         y_coord = 3911311.70
331         height_m = 19.81
332
333       if stationslist[i].upper() == "LGSA":
334         stationtype = "airport"
335         x_coord = 238414.17
336         y_coord = 3930463.80
337         height_m = 150.88
338
339       if stationslist[i].upper() == "LGIR":
340         stationtype = "airport"
341         x_coord = 334629.25
342         y_coord = 3912234.84
343         height_m = 39.01
344
345       if stationslist[i].upper() == "LGTL":
346         stationtype = "airport"
347         x_coord = 347686.11
348         y_coord = 3895617.27
349         height_m = 335.89
350
351       if stationslist[i].upper() == "LGST":
352         stationtype = "airport"
353         x_coord = 418207.19
354         y_coord = 3897378.09
355         height_m = 28.04
356
357       if stationslist[i].upper() == "LGKP":
358         stationtype = "airport"
359         x_coord = 513254.08
360         y_coord = 3919786.14
361         height_m = 20.12
362
363       if stationslist[i].upper() == "LGKS":
364         stationtype = "airport"
365         x_coord = 491829.68
```

```
366         y_coord = 3919780.07
367         height_m = 10.67
368
369       if stationslist[i].upper() == "LGSR":
370         stationtype = "airport"
371         x_coord = 363632.22
372         y_coord = 4029301.07
373         height_m = 38.10
374
375
376
377       tmpline = rlinelist[i]
378       tmplinelist = tmpline.split(",")
379
380       # Korrigieren der Temperaturwerte auf Meeresspiegelniveau
381       # Temperature lapse rate: 6.49 K / 1.000 m
382       lapserate_m = 0.00649
383
384       if tmplinelist[1] <> "--":
385         temp_nn = float(tmplinelist[1]) + (height_m * lapserate_m)
386         tmplinelist[1] = str(round(temp_nn, 2))
387
388       if tmplinelist[3] <> "--":
389         temp_nn = float(tmplinelist[3]) + (height_m * lapserate_m)
390         tmplinelist[3] = str(round(temp_nn, 2))
391
392       # Da die Windmessungen bei den Personal Weather Stations fast ↩
              ausschliesslich
393       # den Wert 0.0, wird dieser Wert auf nodata (-999) gesetzt
394
395       if tmplinelist[5] <> "--" and stationtype == "pws":
396         tmplinelist[5] = "-999"
397
398       # Entfernen des letzten Eintrags jeder Zeile (Wind in m/s), wird nicht ↩
              benoetigt
399       # Ersetzen des nodata-Zeichens "--" durch "-999"
400       outline = ""
```

```
401     for j in range(0, len(tmplinelist) -1):
402
403       if tmplinelist[j] == "--":
404         tmplinelist[j] = "-999"
405
406       outline = outline + tmplinelist[j] + ","
407
408     outline = outline[:-1]
409     outfile.write(stationslist[i].upper() + "," + str(y_coord) + "," + str(↩
          x_coord) + "," + str(height_m) + "," + outline + "\n")
410
411
412   outfile.close()
413
414 for i in range(0, len(stationslist)):
415   infilelist[i].close()
```

A.1.13 wunderground_corr.py

```
1  import sys
2  import time
3  from datetime import datetime
4  from datetime import date
5  from datetime import timedelta
6  import subprocess
7  import os
8  import os.path
9
10
11 for j in range(1,len(sys.argv)):
12   if sys.argv[j][0:7].lower() == "s_date=":
13     s_date_str=sys.argv[j][7:]
14   if sys.argv[j][0:7].lower() == "e_date=":
15     e_date_str=sys.argv[j][7:]
16   if sys.argv[j][0:8].lower() == "in_path=":
17     in_path=sys.argv[j][8:]
18   if sys.argv[j][0:9].lower() == "out_path=":
19     out_path=sys.argv[j][9:]
20   if sys.argv[j][0:11].lower() == "in_praefix=":
21     in_praefix=sys.argv[j][11:]
22   if sys.argv[j][0:10].lower() == "in_suffix=":
23     in_suffix=sys.argv[j][10:]
24
25
26 s_year = int(s_date_str[0:4])
27 s_month = int(s_date_str[4:6])
28 s_day = int(s_date_str[6:8])
29
30 e_year = int(e_date_str[0:4])
31 e_month = int(e_date_str[4:6])
32 e_day = int(e_date_str[6:8])
33
34
35
```

```
36 start = datetime(s_year, s_month, s_day)
37 end = datetime(e_year, e_month, e_day)
38 dt = start
39
40
41 infile_station = []
42 infile_lat = []
43 infile_lon = []
44 infile_height = []
45 infile_date = []
46 infile_avgtemp_nn = []
47 infile_precip24all = []
48 infile_tempnoon_nn = []
49 infile_humidnoon = []
50 infile_windkmhnoon = []
51
52
53 i = 0
54
55 while dt <= end:
56
57   dt_year_str = str(dt.year)
58   dt_month_str = str(dt.month)
59   dt_day_str = str(dt.day)
60
61   dt_month_out_str = dt_month_str
62   dt_day_out_str = dt_day_str
63
64   if len(dt_month_str) == 1:
65     dt_month_out_str = "0" + dt_month_out_str
66   if len(dt_day_str) == 1:
67     dt_day_out_str = "0" + dt_day_out_str
68
69
70   infile=open(in_path + in_praefix + dt_year_str + dt_month_out_str + ↩
         dt_day_out_str + in_suffix + ".csv", 'r')
71   titel = infile.readline()
```

```
72
73    infile_avgtemp_nn_val = 0
74    infile_precip24all_val = 0
75    infile_tempnoon_nn_val = 0
76    infile_humidnoon_val = 0
77    infile_windkmhnoon_val = 0
78
79    del infile_station[:]
80    del infile_lat[:]
81    del infile_lon[:]
82    del infile_height[:]
83    del infile_date[:]
84    del infile_avgtemp_nn[:]
85    del infile_precip24all[:]
86    del infile_tempnoon_nn[:]
87    del infile_humidnoon[:]
88    del infile_windkmhnoon[:]
89
90    while True:
91
92      rline = infile.readline()
93
94      if len(rline) == 0:
95        break
96
97      rline = rline.strip()
98
99      rlinelist = rline.split(",")
100
101     infile_station.append(rlinelist[0])
102     infile_lat.append(rlinelist[1])
103     infile_lon.append(rlinelist[2])
104     infile_height.append(rlinelist[3])
105     infile_date.append(rlinelist[4])
106     infile_avgtemp_nn.append(rlinelist[5])
107     infile_precip24all.append(rlinelist[6])
108     infile_tempnoon_nn.append(rlinelist[7])
```

```
109       infile_humidnoon.append(rlinelist[8])
110       infile_windkmhnoon.append(rlinelist[9])
111
112     infile.close()
113
114
115     for h in range(0, len(infile_station)):
116       if infile_avgtemp_nn[h] <> "0" and infile_avgtemp_nn[h] <> "0.0" and ↩
              infile_avgtemp_nn[h] <> "-999":
117         infile_avgtemp_nn_val = 1
118       if infile_precip24all[h] <> "0" and infile_precip24all[h] <> "0.0" and ↩
              infile_precip24all[h] <> "-999":
119         infile_precip24all_val = 1
120       if infile_tempnoon_nn[h] <> "0" and infile_tempnoon_nn[h] <> "0.0" and ↩
              infile_tempnoon_nn[h] <> "-999":
121         infile_tempnoon_nn_val = 1
122       if infile_humidnoon[h] <> "0" and infile_humidnoon[h] <> "0.0" and ↩
              infile_humidnoon[h] <> "-999":
123         infile_humidnoon_val = 1
124       if infile_windkmhnoon[h] <> "0" and infile_windkmhnoon[h] <> "0.0" and ↩
              infile_windkmhnoon[h] <> "-999":
125         infile_windkmhnoon_val = 1
126
127
128     for h in range(0, len(infile_station)):
129       if infile_avgtemp_nn[h] == "0" or infile_avgtemp_nn[h] == "0.0":
130         if infile_avgtemp_nn_val == 1:
131           infile_avgtemp_nn[h] = "-999"
132
133       if infile_precip24all[h] == "0" or infile_precip24all[h] == "0.0":
134         if infile_precip24all_val == 1:
135           infile_precip24all[h] = "-999"
136
137       if infile_tempnoon_nn[h] == "0" or infile_tempnoon_nn[h] == "0.0":
138         if infile_tempnoon_nn_val == 1:
139           infile_tempnoon_nn[h] = "-999"
140
```

```
141     if infile_humidnoon[h] == "0" or infile_humidnoon[h] == "0.0":
142       if infile_humidnoon_val == 1:
143         infile_humidnoon[h] = "-999"
144
145     if infile_windkmhnoon[h] == "0" or infile_windkmhnoon[h] == "0.0":
146       if infile_windkmhnoon_val == 1:
147         infile_windkmhnoon[h] = "-999"
148
149
150   outfile=open(out_path + in_praefix + dt_year_str + dt_month_out_str + ↩
          dt_day_out_str + in_suffix + "_corr.csv", 'w')
151   outfile.write(titel)
152
153   for h in range(0, len(infile_station)):
154
155     outfile.write(infile_station[h] + "," + infile_lat[h] + "," + infile_lon[h]↩
            + "," + infile_height[h] + "," + infile_date[h] + "," + ↩
          infile_avgtemp_nn[h] + "," + infile_precip24all[h] + "," + ↩
          infile_tempnoon_nn[h] + "," + infile_humidnoon[h] + "," + ↩
          infile_windkmhnoon[h] + "\n")
156
157
158   dt = dt + timedelta(1)
159
160   i = i + 1
```

A.1.14 wunderground_idw.py

```
from datetime import datetime
from datetime import date
from datetime import timedelta
from datetime import *
import sys
import subprocess
import os


for j in range(1,len(sys.argv)):
  if sys.argv[j][0:14].lower() == "incsv_praefix=":
    incsv_praefix=sys.argv[j][14:]
  if sys.argv[j][0:13].lower() == "incsv_suffix=":
    incsv_suffix=sys.argv[j][13:]
  if sys.argv[j][0:7].lower() == "s_date=":
    s_date_str=sys.argv[j][7:]
  if sys.argv[j][0:7].lower() == "e_date=":
    e_date_str=sys.argv[j][7:]
  if sys.argv[j][0:9].lower() == "out_path=":
    out_path=sys.argv[j][9:]
  if sys.argv[j][0:12].lower() == "out_praefix=":
    out_praefix=sys.argv[j][12:]
  if sys.argv[j][0:11].lower() == "incsv_path=":
    incsv_path=sys.argv[j][11:]
  if sys.argv[j][0:11].lower() == "indem_path=":
    indem_path=sys.argv[j][11:]
  if sys.argv[j][0:6].lower() == "indem=":
    indem=sys.argv[j][6:]
  if sys.argv[j][0:7].lower() == "nodata=":
    nodata=sys.argv[j][7:]
  if sys.argv[j][0:11].lower() == "projstring=":
    projstring=sys.argv[j][11:]
  if sys.argv[j][0:4].lower() == "reg=":
    reg=sys.argv[j][4:]
  if sys.argv[j][0:4].lower() == "res=":
```

```
36      res=int(sys.argv[j][4:])
37
38  s_year = int(s_date_str[0:4])
39  s_month = int(s_date_str[4:6])
40  s_day = int(s_date_str[6:8])
41
42  e_year = int(e_date_str[0:4])
43  e_month = int(e_date_str[4:6])
44  e_day = int(e_date_str[6:8])
45
46
47
48  start = datetime(s_year, s_month, s_day)
49  end = datetime(e_year, e_month, e_day)
50  dt = start
51
52
53  i = 0
54
55  while dt <= end:
56
57    dt_year_str = str(dt.year)
58    dt_month_str = str(dt.month)
59    dt_day_str = str(dt.day)
60
61    dt_month_out_str = dt_month_str
62    dt_day_out_str = dt_day_str
63
64    if len(dt_month_str) == 1:
65      dt_month_out_str = "0" + dt_month_out_str
66    if len(dt_day_str) == 1:
67      dt_day_out_str = "0" + dt_day_out_str
68
69
70    incsv = incsv_path + incsv_praefix + dt_year_str + dt_month_out_str + ↩
          dt_day_out_str + incsv_suffix + ".csv"
71
```

```
72  # IDW-Interpolation der fuenf benoetigten Werte aus der csv-Detei mittels 'R'
73  # Als Ergebnis werden fuenf csv-Dateien erstellt
74  return_code = subprocess.call("Rscript idw.R " + indem_path + indem + " " + ↩
        incsv + " " + "output_" + " " + nodata + " " + projstring, shell=True)
75
76
77  infile = open(incsv, 'r')
78  outfile = open(out_path + out_praefix + dt_year_str + dt_month_out_str + ↩
        dt_day_out_str + ".csv", 'w')
79
80  rline = infile.readline()
81  outfile.write("STATION,LAT,LON,HEIGHT,DATE,AVGTEMP_HGT,PRECIP24ALL,↩
        TEMPNOON_HGT,HUMIDNOON,WINDKMHNOON" + "\n")
82
83  while True:
84
85    rline = infile.readline()
86    rlinelist = rline.split(",")
87
88    if len(rline) == 0:
89      break
90
91    # Bestimmen der Zeile in der output-csv-Tabelle, die zu einer bestimmten
92    # Wetterstation gehoert (anhand ihrer Koordinaten und der raeumlichen ↩
          Aufloesung
93    # des Grids
94
95    ycoord = float(rlinelist[1])
96    xcoord = float(rlinelist[2])
97    height_m = float(rlinelist[3])
98
99    #ycoord = 4575336.27894572
100   #xcoord = 423347.2353972
101   #ycoord =  4300325.85267772
102   #xcoord = 574362.7860662
103
104   #res = 3000  # Aufloesung des Input / Output - Grids
```

```
105
106         #N: 4575212.46772316     S: 4300325.85267772    Res: 2987.89798962
107         #E: 574362.78615826     W: 423347.2353972    Res: 3020.31101522
108
109     if reg.lower() == "sard":
110       bb_north = 4575212.46772316
111       bb_west = 423347.2353972
112       if res == 3000:
113         reg_cols = 50
114       if res == 300:
115         reg_cols = 503
116
117     #N: 3958027.69765364     S: 3848879.23393273    Res: 3031.90177003
118     #E: 443961.12770907     W: 175121.01962377
119
120     if reg.lower() == "kreta":
121       bb_north = 3958027.69765364
122       bb_west = 175121.01962377
123       if res == 3000:
124         reg_cols = 90
125       if res == 300:
126         reg_cols = 896
127
128
129     row = int(round(((bb_north - ycoord) / res) - 0.5))
130     col = int(round(((xcoord - bb_west) / res) - 0.5))
131
132
133     if row < 1:
134       row = 1
135     if col < 1:
136       col = 1
137
138     cell = ((row -1) * reg_cols) + col
139
140
141     infile_avgtemp_nn = open("output_avgtemp_nn.csv", 'r')
```

```
142     infile_precip24all = open("output_precip24all.csv", 'r')
143     infile_tempnoon_nn = open("output_tempnoon_nn.csv", 'r')
144     infile_humidnoon = open("output_humidnoon.csv", 'r')
145     infile_windkmhnoon = open("output_windkmhnoon.csv", 'r')
146
147     i = 0
148
149     while True:
150
151       avgtemp_nn_rline = infile_avgtemp_nn.readline()
152       precip24all_rline = infile_precip24all.readline()
153       tempnoon_nn_rline = infile_tempnoon_nn.readline()
154       humidnoon_rline = infile_humidnoon.readline()
155       windkmhnoon_rline = infile_windkmhnoon.readline()
156       i = i + 1
157
158       if i -1 == cell:
159         avgtemp_nn_rline_list = avgtemp_nn_rline.split(",")
160         precip24all_rline_list = precip24all_rline.split(",")
161         tempnoon_nn_rline_list = tempnoon_nn_rline.split(",")
162         humidnoon_rline_list = humidnoon_rline.split(",")
163         windkmhnoon_rline_list = windkmhnoon_rline.split(",")
164
165
166
167         avgtemp_nn_f = float(avgtemp_nn_rline_list[1].strip())
168
169         #print avgtemp_nn_rline_list[1].strip()
170         #sys.exit()
171
172         precip24all_out = str(round(float(precip24all_rline_list[1].strip()), ↩
              2))
173         tempnoon_nn_f = float(tempnoon_nn_rline_list[1].strip())
174         humidnoon_out = str(round(float(humidnoon_rline_list[1].strip()), 2))
175         windkmhnoon_out = str(round(float(windkmhnoon_rline_list[1].strip()), ↩
              2))
176
```

```
177
178         # Hochrechnen der Normalnull-Temperatur auf die tatsaechliche Hoehe
179         lapserate_m = 0.00649
180
181         avgtemp_hgt_out = str(round(avgtemp_nn_f - (height_m * lapserate_m), 2)↩
                )
182         tempnoon_hgt_out = str(round(tempnoon_nn_f - (height_m * lapserate_m), ↩
                2))
183
184
185         outfile.write(rlinelist[0] + "," + rlinelist[1] + "," + rlinelist[2] + ↩
                "," + rlinelist[3] + "," + rlinelist[4] + "," + avgtemp_hgt_out + "↩
                ," + precip24all_out + "," + tempnoon_hgt_out + "," + humidnoon_out↩
                 + "," + windkmhnoon_out + "\n")
186         break
187
188
189       infile_avgtemp_nn.close()
190       infile_precip24all.close()
191       infile_tempnoon_nn.close()
192       infile_humidnoon.close()
193       infile_windkmhnoon.close()
194
195     infile.close()
196     outfile.close()
197
198     dt = dt + timedelta(1)
199
200     i = i + 1
201
202     #sys.exit()
```

A.1.15 wunderground_strt.py

```
1  from datetime import datetime
2  from datetime import date
3  from datetime import timedelta
4  from datetime import *
5  import sys
6  import subprocess
7  import os
8
9
10 for j in range(1,len(sys.argv)):
11   if sys.argv[j][0:14].lower() == "incsv_praefix=":
12     incsv_praefix=sys.argv[j][14:]
13   if sys.argv[j][0:7].lower() == "s_date=":
14     s_date_str=sys.argv[j][7:]
15   if sys.argv[j][0:7].lower() == "e_date=":
16     e_date_str=sys.argv[j][7:]
17   if sys.argv[j][0:9].lower() == "out_path=":
18     out_path=sys.argv[j][9:]
19   if sys.argv[j][0:4].lower() == "out=":
20     outfilename=sys.argv[j][4:]
21   if sys.argv[j][0:11].lower() == "incsv_path=":
22     incsv_path=sys.argv[j][11:]
23
24
25 # Oeffnen einer Beispieldatei, um die Anzahl der Stationen und damit dir Laenge
26 # der Listen fuer Stationsnamen, Durchschnittstemperatur, relative ↩
       Luftfeuchtigkeit
27 # und zu ermittelndes Startdatum zu bestimmen
28
29
30 infile = open(incsv_path + incsv_praefix + s_date_str + ".csv", 'r')
31 rline = infile.readline()
32
33 avgtemp_days = []
34 precip24all_days = []
```

```
35 stations = []
36 startdate = []
37 precip = []
38
39 while True:
40
41   rline = infile.readline()
42
43   if len(rline) == 0:
44     break
45
46   rlinelist = rline.split(",")
47   stations.append(rlinelist[0])
48   avgtemp_days.append(0)
49   precip24all_days.append(0)
50   precip.append(0)
51   startdate.append("")
52
53 infile.close()
54
55
56
57 s_year = int(s_date_str[0:4])
58 s_month = int(s_date_str[4:6])
59 s_day = int(s_date_str[6:8])
60
61 e_year = int(e_date_str[0:4])
62 e_month = int(e_date_str[4:6])
63 e_day = int(e_date_str[6:8])
64
65
66
67 start = datetime(s_year, s_month, s_day)
68 end = datetime(e_year, e_month, e_day)
69 dt = start
70
71
```

```
72  i = 0
73
74  while dt <= end:
75
76    dt_year_str = str(dt.year)
77    dt_month_str = str(dt.month)
78    dt_day_str = str(dt.day)
79
80    dt_month_out_str = dt_month_str
81    dt_day_out_str = dt_day_str
82
83    if len(dt_month_str) == 1:
84      dt_month_out_str = "0" + dt_month_out_str
85    if len(dt_day_str) == 1:
86      dt_day_out_str = "0" + dt_day_out_str
87
88
89    infile = open(incsv_path + incsv_praefix + dt_year_str + dt_month_out_str + ↩
          dt_day_out_str + ".csv", 'r')
90
91    # Headerzeile auslesen
92    rline = infile.readline()
93
94
95
96    for j in range(0, len(stations)):
97
98      rline = infile.readline()
99
100     if len(rline) == 0:
101       break
102
103     rlinelist = rline.split(",")
104     avgtemp_hgt = float(rlinelist[5])
105     precip24all = float(rlinelist[6])
106
107     #
```

```
108     #if  rlinelist[6] <> "0.0":
109     # print rlinelist[0] + ", " + dt_year_str + dt_month_out_str + ↩
            dt_day_out_str
110     # sys.exit()
111     #
112
113
114     # if startdate[j] == "":
115     # Sobald es an einer Station 3 Tage nicht geregnet hat, wird dieses
116     #   Datum als Ergebnis uebernommen und die Anzahl der regenfreien Tage
117     # gespeichert. Das gilt auch, wenn es im untersuchten Zeitraum noch gar
118     # nicht geregnet hat bzw. keine Messwerte vorliegen -> Fehlerquelle
119
120     # if startdate[j] == "" or precip[j] == 0:
121     # Ein gefundenes Datum wird nur dann als Ergebnis uebernommen, wenn
122     #   es im untersuchten Zeitraum schon einen Regentag gab.
123
124     #if precip24all <> 0.0:
125     # print dt_year_str + dt_month_out_str + dt_day_out_str
126     # sys.exit()
127
128     if startdate[j] == "":
129       if avgtemp_hgt >= 6:
130         avgtemp_days[j] = avgtemp_days[j] + 1
131       else:
132         avgtemp_days[j] = 0
133
134       if precip24all == 0.0:
135         precip24all_days[j] = precip24all_days[j] + 1
136       else:
137         precip24all_days[j] = 0
138         precip[j] = precip[j] + 1
139         if precip[j] == 1:
140           avgtemp_days[j] = 0
141
142       if avgtemp_days[j] >= 3 and precip[j] >= 1:
143         startdate[j] = dt_year_str + dt_month_out_str + dt_day_out_str
```

```
144
145
146     infile.close()
147
148     dt = dt + timedelta(1)
149
150     i = i + 1
151
152     #sys.exit()
153
154
155 outfile = open(out_path + outfilename, 'w')
156 outfile.write("STATION,STARTDATE,AVGTEMP_DAYS,PRECIP24ALL_DAYS" + "\n")
157
158
159 for k in range(0, len(stations)):
160
161     outfile.write(str(stations[k]) + "," + str(startdate[k]) + "," + str(↩
            avgtemp_days[k]) + "," + str(precip24all_days[k]) + "\n")
162
163 outfile.close()
```

A.1.16 wunderground_splt_append.py

```
1  import sys
2  import time
3  from datetime import datetime
4  from datetime import date
5  from datetime import timedelta
6  import subprocess
7  import os
8  import os.path
9
10
11 in_praefix = ""
12 for j in range(1,len(sys.argv)):
13   if sys.argv[j][0:9].lower() == "stations=":
14     stations=sys.argv[j][9:].upper()
15   if sys.argv[j][0:7].lower() == "s_date=":
16     s_date_str=sys.argv[j][7:]
17   if sys.argv[j][0:7].lower() == "e_date=":
18     e_date_str=sys.argv[j][7:]
19   if sys.argv[j][0:8].lower() == "in_path=":
20     in_path=sys.argv[j][8:]
21   if sys.argv[j][0:11].lower() == "in_praefix=":
22     in_praefix=sys.argv[j][11:]
23   if sys.argv[j][0:9].lower() == "out_path=":
24     out_path=sys.argv[j][9:]
25   if sys.argv[j][0:12].lower() == "out_praefix=":
26     out_praefix=sys.argv[j][12:]
27   if sys.argv[j][0:11].lower() == "out_suffix=":
28     out_suffix=sys.argv[j][11:]
29
30
31 stationslist = stations.split(",")
32
33 for i in range(0, len(stationslist)):
34   infile=open(in_path + in_praefix + stationslist[i] + "_" + s_date_str + "_" +↩
         e_date_str + ".csv", 'r')
```

```
35    outfile=open(out_path + out_praefix + stationslist[i] + out_suffix + ".csv", ↩
          'a')
36
37    rline1 = infile.readline()
38    rline2 = infile.readline()
39
40    outfile.write(rline2)
41
42    infile.close()
43    outfile.close()
```

A.1.17 wunderground_splt.py

```
from datetime import datetime
from datetime import date
from datetime import timedelta
from datetime import *
import sys
import subprocess
import os


for j in range(1,len(sys.argv)):
  if sys.argv[j][0:11].lower() == "incsv_path=":
    incsv_path=sys.argv[j][11:]
  if sys.argv[j][0:14].lower() == "incsv_praefix=":
    incsv_praefix=sys.argv[j][14:]
  if sys.argv[j][0:8].lower() == "instart=":
    instart_filename=sys.argv[j][8:]
  if sys.argv[j][0:13].lower() == "instart_path=":
    instart_path=sys.argv[j][13:]
  if sys.argv[j][0:7].lower() == "s_date=":
    s_date_str=sys.argv[j][7:]
  if sys.argv[j][0:7].lower() == "e_date=":
    e_date_str=sys.argv[j][7:]
  if sys.argv[j][0:9].lower() == "out_path=":
    out_path=sys.argv[j][9:]
  if sys.argv[j][0:12].lower() == "out_praefix=":
    out_praefix=sys.argv[j][12:]
  if sys.argv[j][0:11].lower() == "out_suffix=":
    out_suffix=sys.argv[j][11:]


# Einmaliges Oeffnen der csv-Datei mit dem Startdatum, um Anzahl der Stationen zu ermitteln

infile = open(incsv_path + incsv_praefix + s_date_str + ".csv", 'r')
rline = infile.readline()
```

```
35 outfile_station = []
36 outfile_station_startdate = []
37
38
39 k = 0
40 outfile_station_max_startdate_int = 0
41
42 while True:
43
44   rline = infile.readline()
45   rlinelist = rline.split(",")
46
47   if len(rline) == 0:
48     break
49
50   outfile_station.append("")
51   outfile_station_startdate.append("")
52
53   # Fuer jede in der Beispieldatei enthaltene Station eine neue Datei anlegen ↩
       und
54   # diese geoeffnet halten
55
56   outfile_station[k] = open(out_path + out_praefix + rlinelist[0] + "_" + ↩
       s_date_str + "_" + e_date_str + ".csv", 'w')
57
58   # Fuer jede Zeile in der Beispieldatei (heisst: jede Station) einmal die csv-↩
       Datei
59   # mit den Startdaten oeffnen, und das betreffende Startdatum sowie regenfreie↩
        Tage
60   # und Tage mit Durchschnittstemperatur ueber 6 Grad heraussuchen (anhand des ↩
       Stationsnamens)
61
62   instart = open(instart_path + instart_filename, 'r')
63   rline_start = instart.readline()
64
65
66   while True:
```

```
67
68      rline_start = instart.readline()
69
70      if len(rline_start) == 0:
71        break
72
73      rlinelist_start = rline_start.split(",")
74
75      if rlinelist_start[0] == rlinelist[0].upper():
76        stationname = rlinelist_start[0]
77        startdate = rlinelist_start[1]
78        days_over_6_degrees = int(rlinelist_start[2])
79        rainfree_days = int(rlinelist_start[3])
80
81
82    outfile_station_startdate[k] = startdate
83
84    # spaetestes Startdatum der verschiedenen Wetterstationen ermitteln
85    if int(outfile_station_startdate[k]) > outfile_station_max_startdate_int:
86      outfile_station_max_startdate_int = int(outfile_station_startdate[k])
87
88
89    # FFMC set to 85
90      # DMC set to 2 times the number of days since precipitation
91      # DC set to 5 times the number of days since precipitation*
92
93    # Setzten der Init-Werte in der Header-Zeile der Stations-csv-Dateien
94
95    ffmc_init = "85"
96    dmc_init = str(2 * int(rainfree_days))
97    dc_init = str(5 * int(rainfree_days))
98
99
100   if stationname.upper() == "IORISTAN5":
101     lat = "39.871"
102
103   if stationname.upper() == "ISARDEGN15":
```

```
104       lat = "40.522"
105
106     if stationname.upper() == "ISSPORTO1":
107       lat = "40.839"
108
109     if stationname.upper() == "ISARDEGN6":
110       lat = "39.225"
111
112     if stationname.upper() == "ISARDEGN32":
113       lat = "39.305"
114
115     if stationname.upper() == "ISARDEGN35":
116       lat = "39.319"
117
118     if stationname.upper() == "IORISTAN2":
119       lat = "39.912"
120
121     if stationname.upper() == "ISARDEGN39":
122       lat = "40.379"
123
124     if stationname.upper() == "ISARDEGN27":
125       lat = "40.582"
126
127     if stationname.upper() == "IOGLIAST2":
128       lat = "39.679"
129
130     if stationname.upper() == "ISARDEGN8":
131       lat = "39.235"
132
133     if stationname.upper() == "ICISANGI2":
134       lat = "39.135"
135
136     if stationname.upper() == "ISARDEGN9":
137       lat = "39.443"
138
139     if stationname.upper() == "ISARDEGN38":
140       lat = "39.211"
```

```
141
142   if stationname.upper() == "IITALIAS2":
143     lat = "40.750"
144
145   if stationname.upper() == "ISARDEGN5":
146     lat = "39.307"
147
148   if stationname.upper() == "ICAGLIAR7":
149     lat = "39.289"
150
151   if stationname.upper() == "ICAGLIAR8":
152     lat = "39.290"
153
154   if stationname.upper() == "ICAGLIAR6":
155     lat = "39.308"
156
157   if stationname.upper() == "IORISTAN4":
158     lat = "39.971"
159
160   if stationname.upper() == "ISARDEGN17":
161     lat = "39.170"
162
163   if stationname.upper() == "ISARDEGN22":
164     lat = "40.006"
165
166   if stationname.upper() == "ISARDEGN40":
167     lat = "39.361"
168
169   if stationname.upper() == "ISARDEGN10":
170     lat = "39.546"
171
172   if stationname.upper() == "LIEA":
173     lat = "40.6311"
174
175   if stationname.upper() == "LIEB":
176     lat = "39.9307"
177
```

```
178   if stationname.upper() == "LIEC":
179     lat = "39.1039"
180
181   if stationname.upper() == "LIED":
182     lat = "39.3461"
183
184   if stationname.upper() == "LIEE":
185     lat = "39.2434"
186
187   if stationname.upper() == "LIEF":
188     lat = "39.7399"
189
190   if stationname.upper() == "LIEG":
191     lat = "41.2167"
192
193   if stationname.upper() == "LIEH":
194     lat = "40.5611"
195
196   if stationname.upper() == "LIEL":
197     lat = "39.4981"
198
199   if stationname.upper() == "LIEN":
200     lat = "40.1167"
201
202   if stationname.upper() == "LIEO":
203     lat = "40.8986"
204
205   if stationname.upper() == "LIEP":
206     lat = "39.6747"
207
208   if stationname.upper() == "LIER":
209     lat = "39.8950"
210
211   if stationname.upper() == "LIET":
212     lat = "39.9188"
213
214   if stationname.upper() == "LFKF":
```

```
215       lat = "41.5006"
216
217
218
219     if stationname.upper() == "IRETHYMN4":
220       lat = "35.187"
221
222     if stationname.upper() == "ICHANIAC2":
223       lat = "35.532"
224
225     if stationname.upper() == "ILASITHI2":
226       lat = "35.039"
227
228     if stationname.upper() == "IHERAKLI1":
229       lat = "35.335"
230
231     if stationname.upper() == "IHERAKLI3":
232       lat = "35.322"
233
234     if stationname.upper() == "IHERAKLI4":
235       lat = "35.330"
236
237     if stationname.upper() == "LGSA":
238       lat = "35.4833"
239
240     if stationname.upper() == "LGIR":
241       lat = "35.3397"
242
243     if stationname.upper() == "LGTL":
244       lat = "35.1920"
245
246     if stationname.upper() == "LGST":
247       lat = "35.2161"
248
249     if stationname.upper() == "LGKP":
250       lat = "35.4214"
251
```

```
252     if stationname.upper() == "LGKS":
253       lat = "35.4214"
254
255     if stationname.upper() == "LGSR":
256       lat = "36.3992"
257
258
259
260
261
262
263
264     outfile_station[k].write("%m/%d/%Y" + "," + lat + "," + ffmc_init + "," + ↩
            dmc_init + "," + dc_init + "\n")
265     k = k + 1
266
267
268
269   infile.close()
270
271
272   # Durchlaufen aller Dateien im angegebenen Zeitraum und umsortieren der Daten
273   # war: sortiert nach Datum, alle Stationen in einer Datei pro Tag ↩
        zusammengefasst
274   # wird: sortiert nach Station, saemtliche Daten des Zeitraumes in einer Datei ↩
        pro Station zusammengefasst
275
276   s_year = int(s_date_str[0:4])
277   s_month = int(s_date_str[4:6])
278   s_day = int(s_date_str[6:8])
279
280   e_year = int(e_date_str[0:4])
281   e_month = int(e_date_str[4:6])
282   e_day = int(e_date_str[6:8])
283
284
285
```

```
286 start = datetime(s_year, s_month, s_day)
287 end = datetime(e_year, e_month, e_day)
288 dt = start
289
290
291 i = 0
292
293 while dt <= end:
294
295   dt_year_str = str(dt.year)
296   dt_month_str = str(dt.month)
297   dt_day_str = str(dt.day)
298
299   dt_month_out_str = dt_month_str
300   dt_day_out_str = dt_day_str
301
302   if len(dt_month_str) == 1:
303     dt_month_out_str = "0" + dt_month_out_str
304   if len(dt_day_str) == 1:
305     dt_day_out_str = "0" + dt_day_out_str
306
307
308   incsv = incsv_path + incsv_praefix + dt_year_str + dt_month_out_str + ↩
          dt_day_out_str + ".csv"
309
310   infile = open(incsv, 'r')
311   rline = infile.readline().strip()
312
313
314   # Die Eintraege fuer den aktuellen Tag in die jeweilige geoeffnete ↩
          Stationsdatei schreiben
315   # Sollte der aktuelle Tag vor dem Startdatum der jeweiligen Station liegen,
316   # diesen Eintrag ueberspringen
317
318   for h in range(0, len(outfile_station)):
319
320     rline = infile.readline().strip()
```

```
321     rlinelist = rline.split(",")
322
323     # avgtemp_hgt: rlinelist[5]
324     # tempnoon_hgt: rlinelist[7]
325
326     #date format, latitude, FFMC init value, DMC init value, DC init value
327     #date, temperature, relative humidity, wind speed, and rain
328     #-> rlinelist[4], rlinelist[7], rlinelist[8], rlinelist[9], rlinelist[6]
329
330     # pyFWI stuerzt bei negativem Vorzeichen ab, daher Eingabegroessen auf ↩
            Minuszeichen pruefen und ggf. den Wert auf 0.0 setzen
331     if rlinelist[6][:1] == "-":
332       rlinelist[6] = "0.0"
333     if rlinelist[7][:1] == "-":
334       rlinelist[7] = "0.0"
335     if rlinelist[8][:1] == "-":
336       rlinelist[8] = "0.0"
337     if rlinelist[9][:1] == "-":
338       rlinelist[9] = "0.0"
339
340
341     #if int(rlinelist[4]) >= int(outfile_station_startdate[h]):
342     if int(rlinelist[4]) >= outfile_station_max_startdate_int:
343       #wline = rlinelist[4][4:6] + "/" + rlinelist[4][6:8] + "/" + rlinelist↩
              [4][0:4] + "," + rlinelist[7] + "," + rlinelist[8] + "," + rlinelist↩
              [9] + "," + rlinelist[6] + "\n"
344       wline = dt_month_out_str + "/" + dt_day_out_str + "/" + dt_year_str + ","↩
               + rlinelist[7] + "," + rlinelist[8] + "," + rlinelist[9] + "," + ↩
              rlinelist[6] + "\n"
345     else:
346       wline = ""
347       #wline = rlinelist[4][4:6] + "/" + rlinelist[4][6:8] + "/" + rlinelist↩
              [4][0:4] + "," + "-999" + "," + "-999" + "," + "-999" + "," + "-999" ↩
             + "\n"
348
349     if len(rline) == 0:
350       break
```

```
351
352       if wline <> "":
353         outfile_station[h].write(wline)
354
355
356     infile.close()
357
358     dt = dt + timedelta(1)
359
360     i = i + 1
361
362     #sys.exit()
363
364 for h in range(0, len(outfile_station)):
365     outfile_station[h].close()
```

A.1.18 wunderground_fwicomb.py

```
import sys
import time
from datetime import datetime
from datetime import date
from datetime import timedelta
import subprocess
import os
import os.path


in_praefix = ""
for j in range(1,len(sys.argv)):
  if sys.argv[j][0:9].lower() == "stations=":
    stations=sys.argv[j][9:].upper()
  if sys.argv[j][0:7].lower() == "s_date=":
    s_date_str=sys.argv[j][7:]
  if sys.argv[j][0:7].lower() == "e_date=":
    e_date_str=sys.argv[j][7:]
  if sys.argv[j][0:8].lower() == "in_path=":
    in_path=sys.argv[j][8:]
  if sys.argv[j][0:11].lower() == "in_praefix=":
    in_praefix=sys.argv[j][11:]
  if sys.argv[j][0:9].lower() == "out_path=":
    out_path=sys.argv[j][9:]
  if sys.argv[j][0:12].lower() == "out_praefix=":
    out_praefix=sys.argv[j][12:]
  if sys.argv[j][0:11].lower() == "out_suffix=":
    out_suffix=sys.argv[j][11:]


stationslist = stations.split(",")
infilelist = []
rlinelist = []
for i in range(0, len(stationslist)):
  infilelist.append("")
```

```
36    rlinelist.append("")
37
38  for i in range(0, len(stationslist)):
39    infilelist[i] = open(in_path + in_praefix + stationslist[i] + "_" + ↩
          s_date_str + "_" + e_date_str + ".csv", 'r')
40    tmpentry = infilelist[i].readline()
41
42
43  while True:
44
45    for i in range(0, len(stationslist)):
46
47      #print stationslist[i]
48      rlinelist[i] = infilelist[i].readline().strip()
49      tmpentries = rlinelist[i].split(",")
50      tmpentries_month = tmpentries[0][0:2]
51      tmpentries_day = tmpentries[0][3:5]
52      tmpentries_year = tmpentries[0][6:10]
53      tmpentries_date = tmpentries_year + tmpentries_month + tmpentries_day
54      #print tmpentries
55
56      if len(rlinelist[i]) == 0:
57        sys.exit()
58
59
60    #date format, latitude, FFMC init value, DMC init value, DC init value
61    #date, temperature, relative humidity, wind speed, and rain, FFMC, DMC, DC, ↩
          ISI, BUI, FWI
62
63
64
65    outfile=open(out_path + out_praefix + tmpentries_date + out_suffix + ".csv", ↩
          'w')
66    #print out_path + out_praefix + tmpentries_date + out_suffix + ".csv"
67
68    outfile.write("STATION,LAT,LON,HEIGHT,DATE,AVGTEMP_HGT,HUMIDNOON,WINKMHNOON,↩
          PRECIP24ALL,FFMC_CLASS,DMC_CLASS,DC_CLASS,ISI_CLASS,BUI_CLASS,FWI_CLASS" ↩
```

```
        + "\n")
69
70
71  for i in range(0, len(stationslist)):
72
73    if stationslist[i].upper() == "IORISTAN5":
74      stationtype = "pws"
75      x_coord = 495381.91
76      y_coord = 4413441.04
77      height_m = 233.17
78
79    if stationslist[i].upper() == "ISARDEGN15":
80      stationtype = "pws"
81      x_coord = 478230.50
82      y_coord = 4485728.46
83      height_m = 487.68
84
85    if stationslist[i].upper() == "ISSPORTO1":
86      stationtype = "pws"
87      x_coord = 449669.28
88      y_coord = 4521056.26
89      height_m = 2.74
90
91    if stationslist[i].upper() == "ISARDEGN6":
92      stationtype = "pws"
93      x_coord = 511566.46
94      y_coord = 4341754.11
95      height_m = 12.80
96
97    if stationslist[i].upper() == "ISARDEGN32":
98      stationtype = "pws"
99      x_coord = 458873.55
100     y_coord = 4350732.09
101     height_m = 225.86
102
103   if stationslist[i].upper() == "ISARDEGN35":
104     stationtype = "pws"
```

```
105         x_coord = 458536.94
106         y_coord = 4352287.61
107         height_m = 294.13
108
109       if stationslist[i].upper() == "IORISTAN2":
110         stationtype = "pws"
111         x_coord = 465213.99
112         y_coord = 4418069.43
113         height_m = 3.66
114
115       if stationslist[i].upper() == "ISARDEGN39":
116         stationtype = "pws"
117         x_coord = 559165.71
118         y_coord = 4470057.04
119         height_m = 14.94
120
121       if stationslist[i].upper() == "ISARDEGN27":
122         stationtype = "pws"
123         x_coord = 500253.89
124         y_coord = 4492356.79
125         height_m = 402.95
126
127       if stationslist[i].upper() == "IOGLIAST2":
128         stationtype = "pws"
129         x_coord = 537647.85
130         y_coord = 4392222.44
131         height_m = 611.12
132
133       if stationslist[i].upper() == "ISARDEGN8":
134         stationtype = "pws"
135         x_coord = 516138.96
136         y_coord = 4342871.97
137         height_m = 26.82
138
139       if stationslist[i].upper() == "ICISANGI2":
140         stationtype = "pws"
141         x_coord = 456267.84
```

```
142         y_coord = 4331879.73
143         height_m = 10.36
144
145       if stationslist[i].upper() == "ISARDEGN9":
146         stationtype = "pws"
147         x_coord = 546552.75
148         y_coord = 4366078.27
149         height_m = 14.94
150
151       if stationslist[i].upper() == "ISARDEGN38":
152         stationtype = "pws"
153         x_coord = 546275.15
154         y_coord = 4340328.74
155         height_m = 49.07
156
157       if stationslist[i].upper() == "IITALIAS2":
158         stationtype = "pws"
159         x_coord = 461336.26
160         y_coord = 4511106.17
161         height_m = 142.04
162
163       if stationslist[i].upper() == "ISARDEGN5":
164         stationtype = "pws"
165         x_coord = 506552.44
166         y_coord = 4350848.36
167         height_m = 55.78
168
169       if stationslist[i].upper() == "ICAGLIAR7":
170         stationtype = "pws"
171         x_coord = 515436.68
172         y_coord = 4348863.29
173         height_m = 75.90
174
175       if stationslist[i].upper() == "ICAGLIAR8":
176         stationtype = "pws"
177         x_coord = 515608.93
178         y_coord = 4348974.61
```

```
179         height_m = 79.86
180
181       if stationslist[i].upper() == "ICAGLIAR6":
182         stationtype = "pws"
183         x_coord = 516811.95
184         y_coord = 4350974.70
185         height_m = 152.40
186
187       if stationslist[i].upper() == "IORISTAN4":
188         stationtype = "pws"
189         x_coord = 468147.33
190         y_coord = 4424605.12
191         height_m = 28.96
192
193       if stationslist[i].upper() == "ISARDEGN17":
194         stationtype = "pws"
195         x_coord = 458276.35
196         y_coord = 4335753.02
197         height_m = 99.97
198
199       if stationslist[i].upper() == "ISARDEGN22":
200         stationtype = "pws"
201         x_coord = 518948.20
202         y_coord = 4428446.76
203         height_m = 868.68
204
205       if stationslist[i].upper() == "ISARDEGN40":
206         stationtype = "pws"
207         x_coord = 500172.30
208         y_coord = 4356838.39
209         height_m = 42.06
210
211       if stationslist[i].upper() == "ISARDEGN10":
212         stationtype = "pws"
213         x_coord = 469669.39
214         y_coord = 4377429.13
215         height_m = 99.06
```

```
216
217     if stationslist[i].upper() == "LIEA":
218       stationtype = "airport"
219       x_coord = 490359.17
220       y_coord = 4497813.22
221       height_m = 22.86
222
223     if stationslist[i].upper() == "LIEB":
224       stationtype = "airport"
225       x_coord = 560940.32
226       y_coord = 4420309.10
227       height_m = 149.96
228
229     if stationslist[i].upper() == "LIEC":
230       stationtype = "airport"
231       x_coord = 544399.88
232       y_coord = 4328432.07
233       height_m = 117.96
234
235     if stationslist[i].upper() == "LIED":
236       stationtype = "airport"
237       x_coord = 497199.53
238       y_coord = 4355185.32
239       height_m = 28.04
240
241     if stationslist[i].upper() == "LIEE":
242       stationtype = "airport"
243       x_coord = 505203.55
244       y_coord = 4343789.24
245       height_m = 3.96
246
247     if stationslist[i].upper() == "LIEF":
248       stationtype = "airport"
249       x_coord = 453688.38
250       y_coord = 4399028.96
251       height_m = 95.10
252
```

```
253     if stationslist[i].upper() == "LIEG":
254       stationtype = "airport"
255       x_coord = 533530.23
256       y_coord = 4562890.39
257       height_m = 159.11
258
259     if stationslist[i].upper() == "LIEH":
260       stationtype = "airport"
261       x_coord = 429150.22
262       y_coord = 4490373.34
263       height_m = 203.91
264
265     if stationslist[i].upper() == "LIEL":
266       stationtype = "airport"
267       x_coord = 554116.87
268       y_coord = 4372242.71
269       height_m = 4.88
270
271     if stationslist[i].upper() == "LIEN":
272       stationtype = "airport"
273       x_coord = 521303.56
274       y_coord = 4440739.86
275       height_m = 1029.00
276
277     if stationslist[i].upper() == "LIEO":
278       stationtype = "airport"
279       x_coord = 543614.53
280       y_coord = 4527629.83
281       height_m = 13.11
282
283     if stationslist[i].upper() == "LIEP":
284       stationtype = "airport"
285       x_coord = 538001.82
286       y_coord = 4391746.93
287       height_m = 605.94
288
289     if stationslist[i].upper() == "LIER":
```

```
290         stationtype = "airport"
291         x_coord = 469488.48
292         y_coord = 4416164.30
293         height_m = 10.97
294
295       if stationslist[i].upper() == "LIET":
296         stationtype = "airport"
297         x_coord = 558369.92
298         y_coord = 4418968.13
299         height_m = 7.01
300
301       if stationslist[i].upper() == "LFKF":
302         stationtype = "airport"
303         x_coord = 508162.58
304         y_coord = 4594335.45
305         height_m = 22.86
306
307
308
309
310
311
312       if stationslist[i].upper() == "IRETHYMN4":
313         stationtype = "pws"
314         x_coord = 263714.58
315         y_coord = 3896865.62
316         height_m = 60.96
317
318       if stationslist[i].upper() == "ICHANIAC2":
319         stationtype = "pws"
320         x_coord = 234336.76
321         y_coord = 3935992.12
322         height_m = 121.92
323
324       if stationslist[i].upper() == "ILASITHI2":
325         stationtype = "pws"
326         x_coord = 406692.17
```

```
327         y_coord = 3877846.31
328         height_m = 0.00
329 
330       if stationslist[i].upper() == "IHERAKLI1":
331         stationtype = "pws"
332         x_coord = 331592.91
333         y_coord = 3911769.63
334         height_m = 29.87
335 
336       if stationslist[i].upper() == "IHERAKLI3":
337         stationtype = "pws"
338         x_coord = 327838.67
339         y_coord = 3910398.15
340         height_m = 10.97
341 
342       if stationslist[i].upper() == "IHERAKLI4":
343         stationtype = "pws"
344         x_coord = 326492.13
345         y_coord = 3911311.70
346         height_m = 19.81
347 
348       if stationslist[i].upper() == "LGSA":
349         stationtype = "airport"
350         x_coord = 238414.17
351         y_coord = 3930463.80
352         height_m = 150.88
353 
354       if stationslist[i].upper() == "LGIR":
355         stationtype = "airport"
356         x_coord = 334629.25
357         y_coord = 3912234.84
358         height_m = 39.01
359 
360       if stationslist[i].upper() == "LGTL":
361         stationtype = "airport"
362         x_coord = 347686.11
363         y_coord = 3895617.27
```

```
364         height_m = 335.89
365
366       if stationslist[i].upper() == "LGST":
367         stationtype = "airport"
368         x_coord = 418207.19
369         y_coord = 3897378.09
370         height_m = 28.04
371
372       if stationslist[i].upper() == "LGKP":
373         stationtype = "airport"
374         x_coord = 513254.08
375         y_coord = 3919786.14
376         height_m = 20.12
377
378       if stationslist[i].upper() == "LGKS":
379         stationtype = "airport"
380         x_coord = 491829.68
381         y_coord = 3919780.07
382         height_m = 10.67
383
384       if stationslist[i].upper() == "LGSR":
385         stationtype = "airport"
386         x_coord = 363632.22
387         y_coord = 4029301.07
388         height_m = 38.10
389
390
391
392       tmpline = rlinelist[i].strip()
393       tmplinelist = tmpline.split(",")
394
395
396       tmpline_month = tmplinelist[0][0:2]
397       tmpline_day = tmplinelist[0][3:5]
398       tmpline_year = tmplinelist[0][6:10]
399       tmpline_date = tmpline_year + tmpline_month + tmpline_day
400
```

```
401     # Berechnungsergebnisse von pyFWI auf 3 Nachkommastellen kuerzen
402     for tmp in range(1,11):
403       #print tmplinelist
404
405       tmpval = round(float(tmplinelist[tmp]),3)
406       tmplinelist[tmp] = str(tmpval)
407
408
409     ffmc_float = float(tmplinelist[5])
410     dmc_float = float(tmplinelist[6])
411     dc_float = float(tmplinelist[7])
412     isi_float = float(tmplinelist[8])
413     bui_float = float(tmplinelist[9])
414     fwi_float = float(tmplinelist[10])
415
416
417     outfile.write(stationslist[i].upper() + "," + str(y_coord) + "," + str(↩
            x_coord) + "," + str(height_m) + "," + tmpline_date + "," + tmplinelist↩
            [1] + "," + tmplinelist[2] + "," + tmplinelist[3] + "," + tmplinelist↩
            [4] + "," + str(ffmc_float) + "," + str(dmc_float) + "," + str(dc_float↩
            ) + "," + str(isi_float) + "," + str(bui_float) + "," + str(fwi_float) ↩
            + "\n")
418
419
420   outfile.close()
421
422 for i in range(0, len(stationslist)):
423   infilelist[i].close()
```

A.1.19 wunderground_fwicomb_cut.py

```
1  import sys
2  import time
3  from datetime import datetime
4  from datetime import date
5  from datetime import timedelta
6  import subprocess
7  import os
8  import os.path
9
10
11 in_praefix = ""
12 for j in range(1,len(sys.argv)):
13   if sys.argv[j][0:9].lower() == "stations=":
14     stations=sys.argv[j][9:].upper()
15   if sys.argv[j][0:7].lower() == "s_date=":
16     s_date_str=sys.argv[j][7:]
17   if sys.argv[j][0:7].lower() == "e_date=":
18     e_date_str=sys.argv[j][7:]
19   if sys.argv[j][0:8].lower() == "in_path=":
20     in_path=sys.argv[j][8:]
21   if sys.argv[j][0:11].lower() == "in_praefix=":
22     in_praefix=sys.argv[j][11:]
23   if sys.argv[j][0:9].lower() == "out_path=":
24     out_path=sys.argv[j][9:]
25   if sys.argv[j][0:12].lower() == "out_praefix=":
26     out_praefix=sys.argv[j][12:]
27   if sys.argv[j][0:10].lower() == "in_suffix=":
28     in_suffix=sys.argv[j][10:]
29
30
31 stationslist = stations.split(",")
32
33 for i in range(0, len(stationslist)):
34   infile=open(in_path + in_praefix + stationslist[i] + in_suffix + ".csv", 'r')
35   outfile=open(out_path + out_praefix + stationslist[i] + "_" + s_date_str + "_↩
```

```
        " + e_date_str + ".csv", 'w')
36    outfile.write("%m/%d/%Y" + "\n")
37
38    rline = ""
39    while True:
40      oldrline = rline
41      rline = infile.readline()
42
43      if len(rline) == 0:
44        outfile.write(oldrline + "\n")
45        break
46
47    infile.close()
48    outfile.close()
```

A.1.20 wunderground_fwiidw.py

```
from datetime import datetime
from datetime import date
from datetime import timedelta
from datetime import *
import sys
import subprocess
import os


for j in range(1,len(sys.argv)):
  if sys.argv[j][0:14].lower() == "incsv_praefix=":
    incsv_praefix=sys.argv[j][14:]
  if sys.argv[j][0:13].lower() == "incsv_suffix=":
    incsv_suffix=sys.argv[j][13:]
  if sys.argv[j][0:11].lower() == "incsv_path=":
    incsv_path=sys.argv[j][11:]
  if sys.argv[j][0:7].lower() == "s_date=":
    s_date_str=sys.argv[j][7:]
  if sys.argv[j][0:7].lower() == "e_date=":
    e_date_str=sys.argv[j][7:]
  if sys.argv[j][0:9].lower() == "out_path=":
    out_path=sys.argv[j][9:]
  if sys.argv[j][0:14].lower() == "out_path_merc=":
    out_path_merc=sys.argv[j][14:]
  if sys.argv[j][0:12].lower() == "out_praefix=":
    out_praefix=sys.argv[j][12:]
  if sys.argv[j][0:11].lower() == "out_suffix=":
    out_suffix=sys.argv[j][11:]
  if sys.argv[j][0:6].lower() == "indem=":
    indem=sys.argv[j][6:]
  if sys.argv[j][0:11].lower() == "indem_path=":
    indem_path=sys.argv[j][11:]
  if sys.argv[j][0:7].lower() == "nodata=":
    nodata=sys.argv[j][7:]
  if sys.argv[j][0:11].lower() == "projstring=":
```

```
36      projstring=sys.argv[j][11:]
37
38
39
40  s_year = int(s_date_str[0:4])
41  s_month = int(s_date_str[4:6])
42  s_day = int(s_date_str[6:8])
43
44  e_year = int(e_date_str[0:4])
45  e_month = int(e_date_str[4:6])
46  e_day = int(e_date_str[6:8])
47
48
49
50  start = datetime(s_year, s_month, s_day)
51  end = datetime(e_year, e_month, e_day)
52  dt = start
53
54
55  i = 0
56
57  while dt <= end:
58
59    dt_year_str = str(dt.year)
60    dt_month_str = str(dt.month)
61    dt_day_str = str(dt.day)
62
63    dt_month_out_str = dt_month_str
64    dt_day_out_str = dt_day_str
65
66    if len(dt_month_str) == 1:
67      dt_month_out_str = "0" + dt_month_out_str
68    if len(dt_day_str) == 1:
69      dt_day_out_str = "0" + dt_day_out_str
70
71    dt_date = dt_year_str + dt_month_out_str + dt_day_out_str
72
```

```
73    return_code = subprocess.call("Rscript fwiidw.R " + indem_path + indem + " " ↩
          + incsv_path + incsv_praefix  + dt_date + incsv_suffix + ".csv" + " " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val" + " " + nodata + " " + projstring, shell=↩
          True)
74
75    return_code = subprocess.call("python wunderground_class.py infile=" + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val_dc.asc" + " outfile=" + out_path + ↩
          dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
          out_suffix + "_dc.asc" + " index=dc", shell=True)
76    return_code = subprocess.call("python wunderground_class.py infile=" + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val_dmc.asc" + " outfile=" + out_path + ↩
          dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
          out_suffix + "_dmc.asc" + " index=dmc", shell=True)
77    return_code = subprocess.call("python wunderground_class.py infile=" + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val_ffmc.asc" + " outfile=" + out_path + ↩
          dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
          out_suffix + "_ffmc.asc" + " index=ffmc", shell=True)
78    return_code = subprocess.call("python wunderground_class.py infile=" + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val_fwi.asc" + " outfile=" + out_path + ↩
          dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
          out_suffix + "_fwi.asc" + " index=fwi", shell=True)
79    return_code = subprocess.call("python wunderground_class.py infile=" + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val_bui.asc" + " outfile=" + out_path + ↩
          dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
          out_suffix + "_bui.asc" + " index=bui", shell=True)
80    return_code = subprocess.call("python wunderground_class.py infile=" + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_val_isi.asc" + " outfile=" + out_path + ↩
          dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
          out_suffix + "_isi.asc" + " index=isi", shell=True)
81
```

```
82
83    return_code = subprocess.call("python /usr/bin/gdal_polygonize.py " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_dc" + ".asc" + " " + out_path + dt_year_str + "/↩
          " + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_dc" +↩
           ".gml", shell=True)
84    return_code = subprocess.call("python /usr/bin/gdal_polygonize.py " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_dmc" + ".asc" + " " + out_path + dt_year_str + "↩
          /" + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_dmc"↩
           + ".gml", shell=True)
85    return_code = subprocess.call("python /usr/bin/gdal_polygonize.py " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_ffmc" + ".asc" + " " + out_path + dt_year_str + ↩
          "/" + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
          _ffmc" + ".gml", shell=True)
86    return_code = subprocess.call("python /usr/bin/gdal_polygonize.py " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_fwi" + ".asc" + " " + out_path + dt_year_str + "↩
          /" + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_fwi"↩
           + ".gml", shell=True)
87    return_code = subprocess.call("python /usr/bin/gdal_polygonize.py " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_bui" + ".asc" + " " + out_path + dt_year_str + "↩
          /" + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_bui"↩
           + ".gml", shell=True)
88    return_code = subprocess.call("python /usr/bin/gdal_polygonize.py " + ↩
          out_path + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + ↩
          dt_date + out_suffix + "_isi" + ".asc" + " " + out_path + dt_year_str + "↩
          /" + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_isi"↩
           + ".gml", shell=True)
89
90
91
92    # Project to Spherical Mercator and extract single day danger classes
93    return_code = subprocess.call("ogr2ogr " + out_path_merc + dt_year_str + "/" ↩
          + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
```

```
        _dc_900913" + ".gml " + " -f GML " + out_path + dt_year_str + "/" + ↩
        dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_dc" + ".↩
        gml " + "-s_srs EPSG:32632 -t_srs EPSG:900913", shell=True)
94      return_code = subprocess.call('ogr2ogr -where "DN=1" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dc_900913_verylow" + ".gml " + out_path_merc +↩
         dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dc_900913" + ".gml", shell=True)
95      return_code = subprocess.call('ogr2ogr -where "DN=2" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dc_900913_low" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dc_900913" + ".gml", shell=True)
96      return_code = subprocess.call('ogr2ogr -where "DN=3" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dc_900913_moderate" + ".gml " + out_path_merc ↩
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dc_900913" + ".gml", shell=True)
97      return_code = subprocess.call('ogr2ogr -where "DN=4" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dc_900913_high" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dc_900913" + ".gml", shell=True)
98      return_code = subprocess.call('ogr2ogr -where "DN=5 or DN=6" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dc_900913_extreme" + ".gml " + out_path_merc +↩
         dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dc_900913" + ".gml", shell=True)
99
100     return_code = subprocess.call("ogr2ogr " + out_path_merc + dt_year_str + "/" ↩
        + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
        _dmc_900913" + ".gml " + " -f GML " + out_path + dt_year_str + "/" + ↩
        dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_dmc" + ".↩
        gml " + "-s_srs EPSG:32632 -t_srs EPSG:900913", shell=True)
101     return_code = subprocess.call('ogr2ogr -where "DN=1" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dmc_900913_verylow" + ".gml " + out_path_merc ↩
```

```
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dmc_900913" + ".gml", shell=True)
102     return_code = subprocess.call('ogr2ogr -where "DN=2" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dmc_900913_low" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dmc_900913" + ".gml", shell=True)
103     return_code = subprocess.call('ogr2ogr -where "DN=3" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dmc_900913_moderate" + ".gml " + out_path_merc↩
         + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dmc_900913" + ".gml", shell=True)
104     return_code = subprocess.call('ogr2ogr -where "DN=4" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dmc_900913_high" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dmc_900913" + ".gml", shell=True)
105     return_code = subprocess.call('ogr2ogr -where "DN=5 or DN=6" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_dmc_900913_extreme" + ".gml " + out_path_merc ↩
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_dmc_900913" + ".gml", shell=True)
106
107     return_code = subprocess.call("ogr2ogr " + out_path_merc + dt_year_str + "/" ↩
        + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
        _ffmc_900913" + ".gml " + " -f GML " + out_path + dt_year_str + "/" + ↩
        dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_ffmc" + "↩
        .gml " + "-s_srs EPSG:32632 -t_srs EPSG:900913", shell=True)
108     return_code = subprocess.call('ogr2ogr -where "DN=1" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_ffmc_900913_verylow" + ".gml " + out_path_merc↩
         + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_ffmc_900913" + ".gml", shell=True)
109     return_code = subprocess.call('ogr2ogr -where "DN=2" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_ffmc_900913_low" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
```

```
        out_suffix + "_ffmc_900913" + ".gml", shell=True)
110     return_code = subprocess.call('ogr2ogr -where "DN=3" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_ffmc_900913_moderate" + ".gml " + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_ffmc_900913" + ".gml", shell=True)
111     return_code = subprocess.call('ogr2ogr -where "DN=4" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_ffmc_900913_high" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_ffmc_900913" + ".gml", shell=True)
112     return_code = subprocess.call('ogr2ogr -where "DN=5 or DN=6" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_ffmc_900913_extreme" + ".gml " + out_path_merc↩
         + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_ffmc_900913" + ".gml", shell=True)
113
114     return_code = subprocess.call("ogr2ogr " + out_path_merc + dt_year_str + "/" ↩
        + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
        _fwi_900913" + ".gml " + " -f GML " + out_path + dt_year_str + "/" + ↩
        dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_fwi" + ".↩
        gml " + "-s_srs EPSG:32632 -t_srs EPSG:900913", shell=True)
115     return_code = subprocess.call('ogr2ogr -where "DN=1" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_fwi_900913_verylow" + ".gml " + out_path_merc ↩
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_fwi_900913" + ".gml", shell=True)
116     return_code = subprocess.call('ogr2ogr -where "DN=2" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_fwi_900913_low" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_fwi_900913" + ".gml", shell=True)
117     return_code = subprocess.call('ogr2ogr -where "DN=3" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_fwi_900913_moderate" + ".gml " + out_path_merc↩
         + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_fwi_900913" + ".gml", shell=True)
```

```
118    return_code = subprocess.call('ogr2ogr -where "DN=4" -f "GML" ' + ↩
           out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
           + dt_date + out_suffix + "_fwi_900913_high" + ".gml " + out_path_merc + ↩
           dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
           out_suffix + "_fwi_900913" + ".gml", shell=True)
119    return_code = subprocess.call('ogr2ogr -where "DN=5" -f "GML" ' + ↩
           out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
           + dt_date + out_suffix + "_fwi_900913_veryhigh" + ".gml " + out_path_merc↩
            + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
           out_suffix + "_fwi_900913" + ".gml", shell=True)
120    return_code = subprocess.call('ogr2ogr -where "DN=6" -f "GML" ' + ↩
           out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
           + dt_date + out_suffix + "_fwi_900913_extreme" + ".gml " + out_path_merc ↩
           + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
           out_suffix + "_fwi_900913" + ".gml", shell=True)
121
122    return_code = subprocess.call("ogr2ogr " + out_path_merc + dt_year_str + "/" ↩
           + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
           _bui_900913" + ".gml " + " -f GML " + out_path + dt_year_str + "/" + ↩
           dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_bui" + ".↩
           gml " + "-s_srs EPSG:32632 -t_srs EPSG:900913", shell=True)
123    return_code = subprocess.call('ogr2ogr -where "DN=1" -f "GML" ' + ↩
           out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
           + dt_date + out_suffix + "_bui_900913_verylow" + ".gml " + out_path_merc ↩
           + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
           out_suffix + "_bui_900913" + ".gml", shell=True)
124    return_code = subprocess.call('ogr2ogr -where "DN=2" -f "GML" ' + ↩
           out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
           + dt_date + out_suffix + "_bui_900913_low" + ".gml " + out_path_merc + ↩
           dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
           out_suffix + "_bui_900913" + ".gml", shell=True)
125    return_code = subprocess.call('ogr2ogr -where "DN=3" -f "GML" ' + ↩
           out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
           + dt_date + out_suffix + "_bui_900913_moderate" + ".gml " + out_path_merc↩
            + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
           out_suffix + "_bui_900913" + ".gml", shell=True)
126    return_code = subprocess.call('ogr2ogr -where "DN=4" -f "GML" ' + ↩
```

```
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_bui_900913_high" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_bui_900913" + ".gml", shell=True)
127     return_code = subprocess.call('ogr2ogr -where "DN=5 or DN=6" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_bui_900913_extreme" + ".gml " + out_path_merc ↩
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_bui_900913" + ".gml", shell=True)
128
129     return_code = subprocess.call("ogr2ogr " + out_path_merc + dt_year_str + "/" ↩
        + dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "↩
        _isi_900913" + ".gml " + " -f GML " + out_path + dt_year_str + "/" + ↩
        dt_month_out_str + "/" + out_praefix + dt_date + out_suffix + "_isi" + ".↩
        gml " + "-s_srs EPSG:32632 -t_srs EPSG:900913", shell=True)
130     return_code = subprocess.call('ogr2ogr -where "DN=1" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_isi_900913_verylow" + ".gml " + out_path_merc ↩
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_isi_900913" + ".gml", shell=True)
131     return_code = subprocess.call('ogr2ogr -where "DN=2" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_isi_900913_low" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_isi_900913" + ".gml", shell=True)
132     return_code = subprocess.call('ogr2ogr -where "DN=3" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_isi_900913_moderate" + ".gml " + out_path_merc↩
         + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_isi_900913" + ".gml", shell=True)
133     return_code = subprocess.call('ogr2ogr -where "DN=4" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
        + dt_date + out_suffix + "_isi_900913_high" + ".gml " + out_path_merc + ↩
        dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_isi_900913" + ".gml", shell=True)
134     return_code = subprocess.call('ogr2ogr -where "DN=5 or DN=6" -f "GML" ' + ↩
        out_path_merc + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix ↩
```

```
        + dt_date + out_suffix + "_isi_900913_extreme" + ".gml " + out_path_merc ↩
        + dt_year_str + "/" + dt_month_out_str + "/" + out_praefix + dt_date + ↩
        out_suffix + "_isi_900913" + ".gml", shell=True)
135
136     #print "Rscript idw.R " + indem_path + indem + " " + incsv_path + ↩
        incsv_praefix  + dt_date + incsv_suffix + " " + out_path + out_praefix + ↩
        out_suffix +  " " + nodata + " " + projstring
137
138
139     dt = dt + timedelta(1)
140
141     i = i + 1
142
143     #sys.exit()
```

A.1.21 wunderground_class.py

```
1  import sys
2
3
4  for j in range(1,len(sys.argv)):
5    if sys.argv[j][0:7].lower() == "infile=":
6      infilename=sys.argv[j][7:]
7    if sys.argv[j][0:8].lower() == "outfile=":
8      outfilename=sys.argv[j][8:]
9    if sys.argv[j][0:6].lower() == "index=":
10     index=sys.argv[j][6:].lower()
11
12
13 infile = open(infilename, 'r')
14 outfile = open(outfilename, 'w')
15
16 for i in range(0,6):
17   header = infile.readline()
18   outfile.write(header)
19   if "NODATA_VALUE" in header.upper():
20     nodata = int(header.split(" ")[1])
21
22
23 while True:
24
25   rline = infile.readline()
26   rlinelist = rline.split(" ")
27
28   if len(rline) == 0:
29     break
30
31   outline=""
32   for i in range(0, len(rlinelist)):
33     if int(float(rlinelist[i])) <> nodata:
34       val_float = float(rlinelist[i])
35
```

```
36        if index == "ffmc":
37          if val_float < 39.19718:
38            val_class = 1 #VERY LOW
39          if val_float >= 39.19718 and val_float < 56.13564:
40            val_class = 2 #LOW
41          if val_float >= 56.13564 and val_float < 68.89025:
42            val_class = 3 #MODERATE
43          if val_float >= 68.89025 and val_float < 79.05454:
44            val_class = 4 #HIGH
45          if val_float >= 79.05454 and val_float < 87.33479:
46            val_class = 5 #VERY HIGH
47          if val_float >= 87.33479:
48            val_class = 6 #EXTREME
49
50
51        if index == "dmc":
52          if val_float < 42.48784:
53            val_class = 1 #VERY LOW
54          if val_float >= 42.48784 and val_float < 115.99162:
55            val_class = 2 #LOW
56          if val_float >= 115.99162 and val_float < 212.60339:
57            val_class = 3 #MODERATE
58          if val_float >= 212.60339 and val_float < 346.08450:
59            val_class = 4 #HIGH
60          if val_float >= 346.08450 and val_float < 515.85506:
61            val_class = 5 #HIGH
62          if val_float >= 515.85506:
63            val_class = 6 #EXTREME
64
65
66        if index == "dc":
67          if val_float < 280.5417:
68            val_class = 1 #VERY LOW
69          if val_float >= 280.5417 and val_float < 701.1608:
70            val_class = 2 #LOW
71          if val_float >= 701.1608 and val_float < 1499.7300:
72            val_class = 3 #MODERATE
```

```
73          if val_float >= 1499.7300 and val_float < 2868.0294:
74            val_class = 4 #HIGH
75          if val_float >= 2868.0294 and val_float < 4548.2494:
76            val_class = 5 #VERY HIGH
77          if val_float >= 4548.2494:
78            val_class = 6 #EXTREME
79
80
81        if index == "isi":
82          if val_float < 5.979628:
83            val_class = 1 #VERY LOW
84          if val_float >= 5.979628 and val_float < 15.047063:
85            val_class = 2 #LOW
86          if val_float >= 15.047063 and val_float < 30.153462:
87            val_class = 3 #MODERATE
88          if val_float >= 30.153462 and val_float < 61.239854:
89            val_class = 4 #HIGH
90          if val_float >= 61.239854 and val_float < 123.619883:
91            val_class = 5 #VERY HIGH
92          if val_float >= 123.619883:
93            val_class = 6 #EXTREME
94
95
96        if index == "bui":
97          if val_float < 55.67652:
98            val_class = 1 #VERY LOW
99          if val_float >= 55.67652 and val_float < 151.20549:
100           val_class = 2 #LOW
101         if val_float >= 151.20549 and val_float < 279.70854:
102           val_class = 3 #MODERATE
103         if val_float >= 279.70854 and val_float < 470.72905:
104           val_class = 4 #HIGH
105         if val_float >= 470.72905 and val_float < 709.86928:
106           val_class = 5 #VERY HIGH
107         if val_float >= 709.86928:
108           val_class = 6 #EXTREME
109
```

```
110
111         if index == "fwi":
112           if val_float < 9.176579:
113             val_class = 1 #Very LOW
114           if val_float >= 9.176579 and val_float < 24.981299:
115             val_class = 2 #LOW
116           if val_float >= 24.981299 and val_float < 43.375469:
117             val_class = 3 #MODERATE
118           if val_float >= 43.375469 and val_float < 66.643181:
119             val_class = 4 #HIGH
120           if val_float >= 66.643181 and val_float < 109.716633:
121             val_class = 5 #VERY HIGH
122           if val_float >= 109.716633:
123             val_class = 6 #EXTREME
124
125
126
127
128
129
130         """
131         # EFFIS (Juli 2012)
132         #
133         if index == "ffmc":
134           if val_float < 82.7:
135             val_class = 1 #VERY LOW
136           if val_float >= 82.7 and val_float < 86.1:
137             val_class = 2 #LOW
138           if val_float >= 86.1 and val_float < 89.2:
139             val_class = 3 #MODERATE
140           if val_float >= 89.2 and val_float < 93:
141             val_class = 4 #HIGH
142           if val_float >= 93:
143             val_class = 5 #EXTREME
144
145
146         if index == "dmc":
```

```
147           if val_float < 15.7:
148             val_class = 1 #VERY LOW
149           if val_float >= 15.7 and val_float < 27.9:
150             val_class = 2 #LOW
151           if val_float >= 27.9 and val_float < 53.1:
152             val_class = 3 #MODERATE
153           if val_float >= 53.1 and val_float < 140.7:
154             val_class = 4 #HIGH
155           if val_float >= 140.7:
156             val_class = 5 #EXTREME
157
158
159         if index == "dc":
160           if val_float < 256.1:
161             val_class = 1 #VERY LOW
162           if val_float >= 256.1 and val_float < 334.1:
163             val_class = 2 #LOW
164           if val_float >= 334.1 and val_float < 450.6:
165             val_class = 3 #MODERATE
166           if val_float >= 450.6 and val_float < 749.4:
167             val_class = 4 #HIGH
168           if val_float >= 749.4:
169             val_class = 5 #EXTREME
170
171
172         if index == "ffmc":
173           if val_float < 3.2:
174             val_class = 1 #VERY LOW
175           if val_float >= 3.2 and val_float < 5:
176             val_class = 2 #LOW
177           if val_float >= 5 and val_float < 7.5:
178             val_class = 3 #MODERATE
179           if val_float >= 7.5 and val_float < 13.5:
180             val_class = 4 #HIGH
181           if val_float >= 13.4:
182             val_class = 5 #EXTREME
183
```

```
184
185
186         if index == "bui":
187           if val_float < 24.2:
188             val_class = 1 #VERY LOW
189           if val_float >= 24.2 and val_float < 40.7:
190             val_class = 2 #LOW
191           if val_float >= 40.7 and val_float < 73.3:
192             val_class = 3 #MODERATE
193           if val_float >= 73.3 and val_float < 178.1:
194             val_class = 4 #HIGH
195           if val_float >= 178.1:
196             val_class = 5 #EXTREME
197
198
199         if index == "fwi":
200           if val_float < 5.2:
201             val_class = 1 #Very LOW
202           if val_float >= 5.2 and val_float < 11.2:
203             val_class = 2 #LOW
204           if val_float >= 11.2 and val_float < 21.3:
205             val_class = 3 #MODERATE
206           if val_float >= 21.3 and val_float < 38.0:
207             val_class = 4 #HIGH
208           if val_float >= 38.0 and val_float < 50:
209             val_class = 5 #VERY HIGH
210           if val_float >= 50:
211             val_class = 6 #EXTREME
212         """
213
214
215         """
216         # Dimitrakopoulos
217         #
218         if index == "fwi":
219           if val_float < 24.47:
220             val_class = 1 #Very LOW
```

```
221           if val_float >= 24.47 and val_float < 39.30:
222             val_class = 2 #LOW
223           if val_float >= 39.30 and val_float < 48.15:
224             val_class = 3 #MODERATE
225           if val_float >= 48.15 and val_float < 51.28:
226             val_class = 4 #HIGH
227           if val_float >= 51.28 and val_float < 60.82:
228             val_class = 5 #VERY HIGH
229           if val_float >= 60.82:
230             val_class = 6 #EXTREME
231         """
232
233
234
235         """
236         # Van Wagner (1987)
237         #
238         if index == "fwi":
239           if val_float <= 1:
240             val_class = 1 #Very LOW
241           if val_float >= 2 and val_float <= 4:
242             val_class = 2 #LOW
243           if val_float >= 5 and val_float <= 8:
244             val_class = 3 #MODERATE
245           if val_float >= 9 and val_float <= 16:
246             val_class = 4 #HIGH
247           if val_float >= 17 and val_float <= 29:
248             val_class = 5 #VERY HIGH
249           if val_float >= 30:
250             val_class = 6 #EXTREME
251         """
252
253
254
255
256
257       else:
```

```
258         val_class = nodata
259
260
261       outline = outline + str(val_class) + " "
262
263     outfile.write(outline + "\n")
264
265 infile.close()
266 outfile.close()
```

A.1.22 platform.py

```
1  import sys
2  import datetime
3  import subprocess
4
5  datestr_forecast = []
6  weekday_forecast = []
7  month_forecast = []
8  info = False
9
10 for j in range(1,len(sys.argv)):
11   if sys.argv[j][0:5].lower() == "date=":
12     datestr=sys.argv[j][5:]
13   if sys.argv[j][0:9].lower() == "out_path=":
14     out_path=sys.argv[j][9:]
15   if sys.argv[j][0:11].lower() == "in_praefix=":
16     in_praefix=sys.argv[j][11:]
17   if sys.argv[j][0:10].lower() == "in_suffix=":
18     in_suffix=sys.argv[j][10:]
19   if sys.argv[j][0:11].lower() == "gmlin_path=":
20     gmlin_path=sys.argv[j][11:]
21   if sys.argv[j][0:15].lower() == "gmlmercin_path=":
22     gmlmercin_path=sys.argv[j][15:]
23   if sys.argv[j][0:14].lower() == "wmsgmlin_path=":
24     wmsgmlin_path = sys.argv[j][14:]
25
26 #Determine the dates for the next 10 days in save in list 'datestr_forecast'
27
28 dateobj = datetime.date(int(datestr[0:4]), int(datestr[4:6]), int(datestr[6:8])↩
       )
29
30 for j in range(0,10):
31   datestr_forecast.append(str(dateobj + datetime.timedelta(days=j)).replace("-"↩
       ,""))
32   weekday = datetime.date(int(datestr_forecast[j][0:4]), int(datestr_forecast[j↩
       ][4:6]), int(datestr_forecast[j][6:8])).weekday()
```

```
33
34    if int(datestr_forecast[j][4:6]) == 1:
35      monthstr = "Jan"
36    if int(datestr_forecast[j][4:6]) == 2:
37      monthstr = "Feb"
38    if int(datestr_forecast[j][4:6]) == 3:
39      monthstr = "Mar"
40    if int(datestr_forecast[j][4:6]) == 4:
41      monthstr = "Apr"
42    if int(datestr_forecast[j][4:6]) == 5:
43      monthstr = "May"
44    if int(datestr_forecast[j][4:6]) == 6:
45      monthstr = "Jun"
46    if int(datestr_forecast[j][4:6]) == 7:
47      monthstr = "Jul"
48    if int(datestr_forecast[j][4:6]) == 8:
49      monthstr = "Aug"
50    if int(datestr_forecast[j][4:6]) == 9:
51      monthstr = "Sep"
52    if int(datestr_forecast[j][4:6]) == 10:
53      monthstr = "Oct"
54    if int(datestr_forecast[j][4:6]) == 11:
55      monthstr = "Nov"
56    if int(datestr_forecast[j][4:6]) == 12:
57      monthstr = "Dec"
58
59
60    if weekday == 0:
61      weekdaystr = "Monday"
62    if weekday == 1:
63      weekdaystr = "Tuesday"
64    if weekday == 2:
65      weekdaystr = "Wednesday"
66    if weekday == 3:
67      weekdaystr = "Thursday"
68    if weekday == 4:
69      weekdaystr = "Friday"
```

```
70    if weekday == 5:
71      weekdaystr = "Saturday"
72    if weekday == 6:
73      weekdaystr = "Sunday"
74
75
76
77    month_forecast.append(monthstr)
78    weekday_forecast.append(weekdaystr)
79    #print str(datestr_forecast[j]) + " " + weekday_forecast[j]
80
81
82
83  for i in range(0, 10):
84    for j in range(0, 6):
85
86      outfilename_day = "forecastday" + str(i)
87
88      if j == 0:
89        outfilename_index_short = "fwi"
90        outfile_index = "Fire Weather Index"
91      if j == 1:
92        outfilename_index_short = "isi"
93        outfile_index = "Initial Spread Speed"
94      if j == 2:
95        outfilename_index_short = "bui"
96        outfile_index = "Build Up Index"
97      if j == 3:
98        outfilename_index_short = "ffmc"
99        outfile_index = "Fine Fuel Moisture Code"
100     if j == 4:
101       outfilename_index_short = "dmc"
102       outfile_index = "Duff Moisture Code"
103     if j == 5:
104       outfilename_index_short = "dc"
105       outfile_index = "Drought Code"
106
```

```
107     outfilename = outfilename_day + "_" + outfilename_index_short
108
109
110     #if i == 0 and j == 0:
111     # outfilename = "index"
112
113
114     outfile_html = open(out_path + outfilename + ".html", 'w')
115
116     outfile_html.write('<HTML>' + '\n')
117     outfile_html.write('<HEAD>' + '\n')
118     outfile_html.write('<META http-equiv="Content-Type" content="text/html; ↩
            charset=utf-8">' + '\n')
119     outfile_html.write('<META name="viewport" content="width=device-width, ↩
            initial-scale=1.0, maximum-scale=1.0, user-scalable=0">' + '\n')
120     outfile_html.write('<META name="apple-mobile-web-app-capable" content="yes↩
            ">' + '\n')
121     outfile_html.write('<TITLE>Sardinia Fireweather</TITLE>' + '\n')
122     outfile_html.write('<LINK rel="stylesheet" href="theme/default/google.css" ↩
            type="text/css">' + '\n')
123     outfile_html.write('<LINK rel="stylesheet" href="style.css" type="text/css↩
            ">' + '\n')
124     outfile_html.write('<SCRIPT src="http://maps.google.com/maps/api/js?v=3&amp↩
            ;sensor=false"></script>' + '\n')
125     outfile_html.write('<SCRIPT src="lib/OpenLayers.js"></script>' + '\n')
126     outfile_html.write('<SCRIPT src="' + outfilename + '.js"></script>' + '\n')
127     outfile_html.write('<STYLE type="text/css">' + '\n')
128     outfile_html.write('a:link { text-decoration:none }' + '\n')
129     outfile_html.write('a:visited { text-decoration:none }' + '\n')
130     outfile_html.write('a:hover { text-decoration:underline }' + '\n')
131     outfile_html.write('a:active { text-decoration:none }' + '\n')
132     outfile_html.write('a:focus { text-decoration:none }' + '\n')
133     outfile_html.write('</STYLE>' + '\n')
134     outfile_html.write('</HEAD>' + '\n')
135     outfile_html.write('' + '\n')
136     outfile_html.write('<BODY leftmargin="0" topmargin="0" marginwidth="0" ↩
            marginheight="0" onload="init()">' + '\n')
```

```
137     outfile_html.write('  ' + '\n')
138     outfile_html.write('  <!-- GESAMT-TABELLE -->' + '\n')
139     outfile_html.write('  <TABLE border=0 align="left" valign="top" width↩
            ="100%" height="100%" cellpadding="0" cellspacing="0">' + '\n')
140     outfile_html.write('  <TR>' + '\n')
141     outfile_html.write('  <TD>' + '\n')
142     outfile_html.write('    ' + '\n')
143     outfile_html.write('    <!-- TITELLEISTE -->' + '\n')
144     outfile_html.write('    <TABLE align="left" width="100%" cellspacing="0" ↩
            cellpadding="5">' + '\n')
145     outfile_html.write('    <TR bgcolor="#133f52" height="30">' + '\n')
146     outfile_html.write('    <TD align="left">' + '\n')
147     outfile_html.write('    <FONT face="Lucida Grande,Verdana,Geneva,Lucida,↩
            Arial,Helvetica" size="3" color=#ffffff> <STRONG>Sardinia ↩
            Fireweather</STRONG></FONT>' + '\n')
148     outfile_html.write('    </TD>' + '\n')
149     outfile_html.write('    <TD align="right">' + '\n')
150     outfile_html.write('    <FONT face="Lucida Grande,Verdana,Geneva,Lucida,↩
            Arial,Helvetica" size="3"><A HREF="info.html" style="color:#ffffff">↩
            information</A></FONT>' + '\n')
151     outfile_html.write('    </TD>' + '\n')
152     outfile_html.write('    </TR>' + '\n')
153     outfile_html.write('    </TABLE>' + '\n')
154     outfile_html.write('    ' + '\n')
155     outfile_html.write('    ' + '\n')
156     outfile_html.write('  </TD>' + '\n')
157     outfile_html.write('  </TR>' + '\n')
158     outfile_html.write('  <TR height=100%>' + '\n')
159     outfile_html.write('  <TD>' + '\n')
160     outfile_html.write('    ' + '\n')
161     outfile_html.write('    <!-- INHALT-TABELLE -->' + '\n')
162     outfile_html.write('    <TABLE width="100%" height="100%" cellspacing="0" ↩
            cellpadding="0">' + '\n')
163     outfile_html.write('    <TR>' + '\n')
164     outfile_html.write('    <TD bgcolor="#1c4e63" width="260" valign="top">' + ↩
            '\n')
165     outfile_html.write('    ' + '\n')
```

```
166     outfile_html.write('      <!-- MENULEISTE -->' + '\n')
167     outfile_html.write('      <TABLE width="100%" cellspacing="0" cellpadding↩
            ="5">' + '\n')
168     outfile_html.write('      <TR>' + '\n')
169     outfile_html.write('      <TD>' + '\n')
170     outfile_html.write('       ' + '\n')
171     outfile_html.write('      </TD>' + '\n')
172     outfile_html.write('      </TR>' + '\n')
173
174     for k in range(0, 10):
175       outfile_html.write('      <TR>' + '\n')
176       outfile_html.write('      <TD>' + '\n')
177       outfile_html.write('      <FONT face="Lucida Grande,Verdana,Geneva,Lucida↩
              ,Arial,Helvetica" size="3"> <A HREF="forecastday' + str(k) + '↩
              _fwi.html"' + ' style="color:#ffffff"><STRONG>' + weekday_forecast[k]↩
               + ', ' + datestr_forecast[k][6:8] + "-" + month_forecast[k] + "-" + ↩
              datestr_forecast[k][0:4] + '</STRONG></A></FONT>' + '\n')
178       outfile_html.write('      </TD>' + '\n')
179       outfile_html.write('      </TR>' + '\n')
180
181       if k == i:
182         outfile_html.write('      <TR>' + '\n')
183         outfile_html.write('      <TD>' + '\n')
184         outfile_html.write('        <TABLE>' + '\n')
185
186         for l in range(0, 6):
187           if l == 0:
188             index = "Fire Weather Index"
189             index_short = "fwi"
190           if l == 1:
191             index = "Initial Spread Speed"
192             index_short = "isi"
193           if l == 2:
194             index = "Build Up Index"
195             index_short = "bui"
196           if l == 3:
197             index = "Fine Fuel Moisture Code"
```

```
198             index_short = "ffmc"
199           if l == 4:
200             index = "Duff Moisture Code"
201             index_short = "dmc"
202           if l == 5:
203             index = "Drought Code"
204             index_short = "dc"
205
206
207           outfile_html.write('        <TR>' + '\n')
208           outfile_html.write('        <TD width="15">' + '\n')
209           outfile_html.write('        </TD>' + '\n')
210           outfile_html.write('        <TD>' + '\n')
211           outfile_html.write('        <FONT face="Lucida Grande,Verdana,Geneva,↩
                  Lucida,Arial,Helvetica" size="3"><A HREF="forecastday' + str(k) +↩
                   '_' + index_short + '.html"' + ' style="color:#6ca9a6">' + index↩
                   + '</A></FONT>' + '\n')
212           outfile_html.write('        </TD>' + '\n')
213           outfile_html.write('        </TR>' + '\n')
214
215         outfile_html.write('        </TABLE>' + '\n')
216         outfile_html.write('      </TD>' + '\n')
217         outfile_html.write('      </TR>' + '\n')
218
219
220     outfile_html.write('      </TABLE>' + '\n')
221     outfile_html.write('    ' + '\n')
222     outfile_html.write('    </TD>' + '\n')
223     outfile_html.write('    <TD valign="top">' + '\n')
224     outfile_html.write('    ' + '\n')
225     outfile_html.write('      <!-- CANVAS -->' + '\n')
226     outfile_html.write('      <TABLE width="100%" height="100%" cellspacing="0"↩
            cellpadding="0">' + '\n')
227     outfile_html.write('      <TR bgcolor="f2f2f2" height="40">' + '\n')
228     outfile_html.write('      <TD>' + '\n')
229     outfile_html.write('        ' + '\n')
230     outfile_html.write('        <!-- UNTER-UEBERSCHRIFT -->' + '\n')
```

```
231     outfile_html.write('        <TABLE width="100%">' + '\n')
232     outfile_html.write('        <TR>' + '\n')
233     outfile_html.write('        <TD align="left">' + '\n')
234     outfile_html.write('        <FONT face="Lucida Grande,Verdana,Geneva,Lucida↩
            ,Arial,Helvetica" color="#20435c" size="5"><STRONG>' + outfile_index + ↩
            '</STRONG></FONT>   ' + '\n')
235     outfile_html.write('        </TD>' + '\n')
236     outfile_html.write('        <TD align="right">' + '\n')
237     outfile_html.write('        <FONT face="Lucida Grande,Verdana,Geneva,Lucida↩
            ,Arial,Helvetica" size="2"><A HREF="' + gmlin_path + datestr_forecast[i↩
            ][0:4] + '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
            datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + datestr_forecast[↩
            i][6:8] + '_airports_pws_' + outfilename_index_short +  '.gml' + '" ↩
            style="color:#20435c">download as GML file (UTM projection)</A></FONT>'↩
             + '\n')
238     outfile_html.write('        </TD>' + '\n')
239     outfile_html.write('        </TABLE>' + '\n')
240     outfile_html.write('        ' + '\n')
241     outfile_html.write('      </TD>' + '\n')
242     outfile_html.write('      </TR>' + '\n')
243     outfile_html.write('      <TR>' + '\n')
244     outfile_html.write('      <TD align="center">' + '\n')
245     outfile_html.write('      <DIV id="map" class="smallmap"></DIV>' + '\n')
246     outfile_html.write('      </TD>' + '\n')
247     outfile_html.write('      </TR>' + '\n')
248     outfile_html.write('' + '\n')
249     outfile_html.write('      </TABLE>' + '\n')
250     outfile_html.write('' + '\n')
251     outfile_html.write('    </TD>' + '\n')
252     outfile_html.write('    </TR>' + '\n')
253     outfile_html.write('    </TABLE>' + '\n')
254     outfile_html.write('  ' + '\n')
255     outfile_html.write('  </TD>' + '\n')
256     outfile_html.write('  </TR>' + '\n')
257     outfile_html.write('  ' + '\n')
258     outfile_html.write('  <TR>' + '\n')
259     outfile_html.write('  <TD>' + '\n')
```

```
260     outfile_html.write('    ' + '\n')
261     outfile_html.write('    <!-- IMPRESSUM -->' + '\n')
262     outfile_html.write('    <TABLE width="100%" cellspacing="0" cellpadding↩
            ="5">' + '\n')
263     outfile_html.write('    <TR bgcolor="#11303d" height="78">' + '\n')
264     outfile_html.write('    <TD align="center" width="100%">' + '\n')
265     outfile_html.write('      <TABLE width="100%" cellspacing="0" cellpadding↩
            ="0">' + '\n')
266     outfile_html.write('      <TR bgcolor="#11303d" height="78">' + '\n')
267     outfile_html.write('      <TD align="left">' + '\n')
268     outfile_html.write('      <A HREF="http://www.wunderground.com"><img src↩
            ="./pic/wunderground_logo/wundergroundLogo_white_horz_small.png" border↩
            ="0" alt="wunderground.com"></A>' + '\n')
269     outfile_html.write('      </TD>' + '\n')
270     outfile_html.write('      <TD align="right">' + '\n')
271     outfile_html.write('      <FONT face="Lucida Grande,Verdana,Geneva,Lucida,↩
            Arial,Helvetica" color="#ffffff" size="2">Impressum:<BR />Michael Nolde↩
            <BR />Christian Albrechts - Universit&auml;t zu Kiel<BR />↩
            Geographisches Institut<BR />Ludewig-Meyn-Str. 14<BR />24118 Kiel, ↩
            Germany</FONT>' + '\n')
272     outfile_html.write('      </TD>' + '\n')
273     outfile_html.write('      </TR>' + '\n')
274     outfile_html.write('      </TABLE' + '\n')
275     outfile_html.write('    </TD>' + '\n')
276     outfile_html.write('    </TR>' + '\n')
277     outfile_html.write('    </TABLE>' + '\n')
278     outfile_html.write('    ' + '\n')
279     outfile_html.write('    ' + '\n')
280     outfile_html.write('  </TD>' + '\n')
281     outfile_html.write('  </TR>' + '\n')
282     outfile_html.write('  </TABLE>' + '\n')
283     outfile_html.write('  ' + '\n')
284     outfile_html.write('</BODY>' + '\n')
285     outfile_html.write('</HTML>' + '\n')
286
287     outfile_html.close()
288
```

```
289
290
291
292
293     #INFO.HTML
294     if info == False:
295         outfileinfo_html = open(out_path + "info.html", 'w')
296
297         outfileinfo_html.write('<HTML>' + '\n')
298         outfileinfo_html.write('<HEAD>' + '\n')
299         outfileinfo_html.write('<META http-equiv="Content-Type" content="text/↩
                html; charset=utf-8">' + '\n')
300         outfileinfo_html.write('<META name="viewport" content="width=device-↩
                width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">' + '↩
                \n')
301         outfileinfo_html.write('<META name="apple-mobile-web-app-capable" ↩
                content="yes">' + '\n')
302         outfileinfo_html.write('<TITLE>Sardinia Fireweather</TITLE>' + '\n')
303         outfileinfo_html.write('<LINK rel="stylesheet" href="theme/default/↩
                google.css" type="text/css">' + '\n')
304         outfileinfo_html.write('<LINK rel="stylesheet" href="style.css" type="↩
                text/css">' + '\n')
305         outfileinfo_html.write('<SCRIPT src="http://maps.google.com/maps/api/js↩
                ?v=3&amp;sensor=false"></script>' + '\n')
306         outfileinfo_html.write('<SCRIPT src="lib/OpenLayers.js"></script>' + '\↩
                n')
307         outfileinfo_html.write('<SCRIPT src="' + outfilename + '.js"></script>'↩
                 + '\n')
308         outfileinfo_html.write('<STYLE type="text/css">' + '\n')
309         outfileinfo_html.write('a:link { text-decoration:none }' + '\n')
310         outfileinfo_html.write('a:visited { text-decoration:none }' + '\n')
311         outfileinfo_html.write('a:hover { text-decoration:underline }' + '\n')
312         outfileinfo_html.write('a:active { text-decoration:none }' + '\n')
313         outfileinfo_html.write('a:focus { text-decoration:none }' + '\n')
314         outfileinfo_html.write('</STYLE>' + '\n')
315         outfileinfo_html.write('</HEAD>' + '\n')
316         outfileinfo_html.write('' + '\n')
```

```
317         outfileinfo_html.write('<BODY leftmargin="0" topmargin="0" marginwidth↩
                ="0" marginheight="0" onload="init()">' + '\n')
318         outfileinfo_html.write('  ' + '\n')
319         outfileinfo_html.write('  <!-- GESAMT-TABELLE -->' + '\n')
320         outfileinfo_html.write('  <TABLE border=0 align="left" valign="top" ↩
                width="100%" height="100%" cellpadding="0" cellspacing="0">' + '\n'↩
                )
321         outfileinfo_html.write('  <TR>' + '\n')
322         outfileinfo_html.write('  <TD>' + '\n')
323         outfileinfo_html.write('    ' + '\n')
324         outfileinfo_html.write('    <!-- TITELLEISTE -->' + '\n')
325         outfileinfo_html.write('    <TABLE align="left" width="100%" ↩
                cellspacing="0" cellpadding="5">' + '\n')
326         outfileinfo_html.write('    <TR bgcolor="#133f52" height="30">' + '\n')
327         outfileinfo_html.write('    <TD align="left">' + '\n')
328         outfileinfo_html.write('    <FONT face="Lucida Grande,Verdana,Geneva,↩
                Lucida,Arial,Helvetica" size="3" color=#ffffff> <STRONG>↩
                Sardinia Fireweather</STRONG></FONT>' + '\n')
329         outfileinfo_html.write('    </TD>' + '\n')
330         outfileinfo_html.write('    <TD align="right">' + '\n')
331         outfileinfo_html.write('    <FONT face="Lucida Grande,Verdana,Geneva,↩
                Lucida,Arial,Helvetica" size="3"><A HREF="info.html" style="color:#↩
                ffffff">information</A></FONT>' + '\n')
332         outfileinfo_html.write('    </TD>' + '\n')
333         outfileinfo_html.write('    </TR>' + '\n')
334         outfileinfo_html.write('    </TABLE>' + '\n')
335         outfileinfo_html.write('    ' + '\n')
336         outfileinfo_html.write('    ' + '\n')
337         outfileinfo_html.write('  </TD>' + '\n')
338         outfileinfo_html.write('  </TR>' + '\n')
339         outfileinfo_html.write('  <TR height=100%>' + '\n')
340         outfileinfo_html.write('  <TD>' + '\n')
341         outfileinfo_html.write('    ' + '\n')
342         outfileinfo_html.write('    <!-- INHALT-TABELLE -->' + '\n')
343         outfileinfo_html.write('    <TABLE width="100%" height="100%" ↩
                cellspacing="0" cellpadding="0">' + '\n')
344         outfileinfo_html.write('    <TR>' + '\n')
```

```
345         outfileinfo_html.write('    <TD bgcolor="#1c4e63" width="260" valign="↩
                top">' + '\n')
346         outfileinfo_html.write('    ' + '\n')
347         outfileinfo_html.write('      <!-- MENULEISTE -->' + '\n')
348         outfileinfo_html.write('      <TABLE width="100%" cellspacing="0" ↩
                cellpadding="5">' + '\n')
349         outfileinfo_html.write('      <TR>' + '\n')
350         outfileinfo_html.write('      <TD>' + '\n')
351         outfileinfo_html.write('       ' + '\n')
352         outfileinfo_html.write('      </TD>' + '\n')
353         outfileinfo_html.write('      </TR>' + '\n')
354
355         for k in range(0, 10):
356           outfileinfo_html.write('      <TR>' + '\n')
357           outfileinfo_html.write('      <TD>' + '\n')
358           outfileinfo_html.write('      <FONT face="Lucida Grande,Verdana,↩
                  Geneva,Lucida,Arial,Helvetica" size="3"> <A HREF="↩
                  forecastday' + str(k) + '_fwi.html"' + ' style="color:#ffffff"><↩
                  STRONG>' + weekday_forecast[k] + ', ' + datestr_forecast[k][6:8] ↩
                  + "-" + month_forecast[k] + "-" + datestr_forecast[k][0:4] + '</↩
                  STRONG></A></FONT>' + '\n')
359           outfileinfo_html.write('      </TD>' + '\n')
360           outfileinfo_html.write('      </TR>' + '\n')
361
362           if k == i:
363             outfileinfo_html.write('      <TR>' + '\n')
364             outfileinfo_html.write('      <TD>' + '\n')
365             outfileinfo_html.write('        <TABLE>' + '\n')
366
367             for l in range(0, 6):
368               if l == 0:
369                 index = "Fire Weather Index"
370                 index_short = "fwi"
371               if l == 1:
372                 index = "Initial Spread Speed"
373                 index_short = "isi"
374               if l == 2:
```

```
375                 index = "Build Up Index"
376                 index_short = "bui"
377               if l == 3:
378                 index = "Fine Fuel Moisture Code"
379                 index_short = "ffmc"
380               if l == 4:
381                 index = "Duff Moisture Code"
382                 index_short = "dmc"
383               if l == 5:
384                 index = "Drought Code"
385                 index_short = "dc"
386
387
388               outfileinfo_html.write('        <TR>' + '\n')
389               outfileinfo_html.write('        <TD width="15">' + '\n')
390               outfileinfo_html.write('        </TD>' + '\n')
391               outfileinfo_html.write('        <TD>' + '\n')
392               outfileinfo_html.write('        <FONT face="Lucida Grande,Verdana↩
                      ,Geneva,Lucida,Arial,Helvetica" size="3"><A HREF="forecastday↩
                      ' + str(k) + '_' + index_short + '.html"' + ' style="color:#6↩
                      ca9a6">' + index + '</A></FONT>' + '\n')
393               outfileinfo_html.write('        </TD>' + '\n')
394               outfileinfo_html.write('        </TR>' + '\n')
395
396             outfileinfo_html.write('        </TABLE>' + '\n')
397             outfileinfo_html.write('      </TD>' + '\n')
398             outfileinfo_html.write('      </TR>' + '\n')
399
400
401         outfileinfo_html.write('      </TABLE>' + '\n')
402         outfileinfo_html.write('    ' + '\n')
403         outfileinfo_html.write('    </TD>' + '\n')
404         outfileinfo_html.write('    <TD valign="top">' + '\n')
405         outfileinfo_html.write('    ' + '\n')
406         outfileinfo_html.write('      <!-- CANVAS -->' + '\n')
407         outfileinfo_html.write('      <TABLE width="100%" height="100%" ↩
                cellspacing="0" cellpadding="0">' + '\n')
```

```
408         outfileinfo_html.write('      <TR bgcolor="f2f2f2" height="40">' + '\n'↩
                )
409         outfileinfo_html.write('      <TD>' + '\n')
410         outfileinfo_html.write('        ' + '\n')
411         outfileinfo_html.write('        <!-- UNTER-UEBERSCHRIFT -->' + '\n')
412         outfileinfo_html.write('        <TABLE width="100%">' + '\n')
413         outfileinfo_html.write('        <TR>' + '\n')
414         outfileinfo_html.write('        <TD align="left">' + '\n')
415         outfileinfo_html.write('        <FONT face="Lucida Grande,Verdana,↩
                Geneva,Lucida,Arial,Helvetica" color="#20435c" size="5"><STRONG>↩
                Information</STRONG></FONT>   ' + '\n')
416         outfileinfo_html.write('        </TD>' + '\n')
417         outfileinfo_html.write('        <TD align="right">' + '\n')
418         outfileinfo_html.write('        </TD>' + '\n')
419         outfileinfo_html.write('        </TABLE>' + '\n')
420         outfileinfo_html.write('        ' + '\n')
421         outfileinfo_html.write('      </TD>' + '\n')
422         outfileinfo_html.write('      </TR>' + '\n')
423         outfileinfo_html.write('      <TR>' + '\n')
424         outfileinfo_html.write('      <TD align="left" valign="top">' + '\n')
425         outfileinfo_html.write('      <FONT face="Lucida Grande,Verdana,Geneva,↩
                Lucida,Arial,Helvetica" color="#000000" size="3">' + '\n')
426         outfileinfo_html.write('      <br />' + '\n')
427         outfileinfo_html.write('           <b>This ↩
                platform was created in the frame of a Ph.D. at Kiel University, ↩
                Germany</b><br />' + '\n')
428         outfileinfo_html.write('           Institute ↩
                of Geography, Chair of Physical Geography - Landscape Ecology and ↩
                Geoinformation<br />' + '\n')
429         outfileinfo_html.write('           <br /><br ↩
                />' + '\n')
430         outfileinfo_html.write('           <b>Title: ↩
                </b>Development of a web platform for wildfire danger forecasting ↩
                on Sardinia using free accessible data and open source applications↩
                <br />' + '\n')
431         outfileinfo_html.write('           <b>Author: ↩
                </b><A HREF="mailto:mnolde@gis.uni-kiel.de" style="color:#000000">↩
```

```
            Michael Nolde</A>, 2012<br />' + '\n')
432         outfileinfo_html.write('           <b>Ph.D. ↩
            supervisor: </b>Prof. Dr. Rainer Duttmann<br /><br /><br />' + '\n'↩
            )
433         outfileinfo_html.write('           <b>This ↩
            platform makes use of several open source components: </b><br />' +↩
             '\n')
434         outfileinfo_html.write('           <A HREF="↩
            http://www.r-project.org/" style="color:#000000">R programming ↩
            language</A> (with sp, gstat and rgdal packages): IDW-Interpolation↩
             (used resolution: 3000m)<br />' + '\n')
435         outfileinfo_html.write('           <A HREF="↩
            http://www.openlayers.org/" style="color:#000000">Open Layers</A>: ↩
            visualizing the danger areas in Dynamic Maps<br />' + '\n')
436         outfileinfo_html.write('           <A HREF="↩
            http://www.python.org/" style="color:#000000">Python Programming ↩
            Language</A>: creation of content<br />' + '\n')
437         outfileinfo_html.write('           <A HREF="↩
            http://www.gdal.org/" style="color:#000000">GDAL/OGR</A>, <A HREF="↩
            http://proj.osgeo.org/" style="color:#000000">PROJ.4</A>: ↩
            Conversion between different coordinate systems and geographic file↩
             formats<br />' + '\n')
438         outfileinfo_html.write('           <br /><br ↩
            />' + '\n')
439         outfileinfo_html.write('           <b>Glossary↩
            </b><br />' + '\n')
440         outfileinfo_html.write('           Fine Fuel ↩
            Moisture Code: numeric rating of the moisture content of litter and↩
             other cured fine fuels<br />' + '\n')
441         outfileinfo_html.write('           Duff ↩
            Moisture Code: numeric rating of the average moisture content of ↩
            loosely compacted organic layers of moderate depth<br />' + '\n')
442         outfileinfo_html.write('           Drought ↩
            Code: numeric rating of the average moisture content of deep, ↩
            compact organic layers<br />' + '\n')
443         outfileinfo_html.write('           Initial ↩
            Spread Index: numeric rating of the expected rate of fire spread<br↩
```

```
        />' + '\n')
444     outfileinfo_html.write('           Buildup Index: numeric rating of the total amount of fuel available for combustion<br />' + '\n')
445     outfileinfo_html.write('           Fire Weather Index: numeric rating of fire intensity<br />' + '\n')
446     outfileinfo_html.write('           (definitions taken from: http://cwfis.cfs.nrcan.gc.ca/background/summary/fwi)<br />' + '\n')
447     outfileinfo_html.write('           <br /><br />' + '\n')
448     outfileinfo_html.write('           University of Kiel: <A HREF="http://www.uni-kiel.de" style="color:#000000">http://www.uni-kiel.de</A><br />' + '\n')
449     outfileinfo_html.write('           Chair of Physical Geography - Landscape Ecology and Geoinformation, Institute of Geography: <A HREF="http://www.lgi.geographie.uni-kiel.de" style="color:#000000">http://www.lgi.geographie.uni-kiel.de</A><br />' + '\n')
450     outfileinfo_html.write('           <br /><br />' + '\n')
451     outfileinfo_html.write("           <b>There's also a mobile version showing today's fire weather: </b><br /><br/>" + "\n")
452     outfileinfo_html.write('           <img src="./pic/mobileversion.png" height="100" width="100">' + '\n')
453     outfileinfo_html.write('           <br /><br /><br />' + '\n')
454     outfileinfo_html.write('           <b>Updating cycle</b><br />' + '\n')
455     outfileinfo_html.write('           The forecasts on this website are calculated every day between 8.00 and 9.30 a.m.<br />' + '\n')
456     outfileinfo_html.write('           <br /><br />      ' + '\n')
457     outfileinfo_html.write('           <b>Disclaimer</b><br />' + '\n')
```

```
458         outfileinfo_html.write('           This website is a work in progress and the forecasting results still need to be validated.<br />' + '\n')
459         outfileinfo_html.write("           The Service is provided 'as is' without warranty of any kind, either express or implied, including, but not limited to, <br />" + "\n")
460         outfileinfo_html.write('           the implied warranties of fitness for a purpose, or the warranty of non-infringement.<br />' + '\n')
461         outfileinfo_html.write('           <br /><br /><br />' + '\n')
462         outfileinfo_html.write('      </FONT>' + '\n')
463         outfileinfo_html.write('      </TD>' + '\n')
464         outfileinfo_html.write('      </TR>' + '\n')
465         outfileinfo_html.write('' + '\n')
466         outfileinfo_html.write('      </TABLE>' + '\n')
467         outfileinfo_html.write('' + '\n')
468         outfileinfo_html.write('    </TD>' + '\n')
469         outfileinfo_html.write('    </TR>' + '\n')
470         outfileinfo_html.write('    </TABLE>' + '\n')
471         outfileinfo_html.write('  ' + '\n')
472         outfileinfo_html.write('  </TD>' + '\n')
473         outfileinfo_html.write('  </TR>' + '\n')
474         outfileinfo_html.write('  ' + '\n')
475         outfileinfo_html.write('  <TR>' + '\n')
476         outfileinfo_html.write('  <TD>' + '\n')
477         outfileinfo_html.write('    ' + '\n')
478         outfileinfo_html.write('    <!-- IMPRESSUM -->' + '\n')
479         outfileinfo_html.write('    <TABLE width="100%" cellspacing="0" cellpadding="5">' + '\n')
480         outfileinfo_html.write('    <TR bgcolor="#11303d" height="78">' + '\n')
481         outfileinfo_html.write('    <TD align="center" width="100%">' + '\n')
482         outfileinfo_html.write('      <TABLE width="100%" cellspacing="0" cellpadding="0">' + '\n')
483         outfileinfo_html.write('      <TR bgcolor="#11303d" height="78">' + '\n')
484         outfileinfo_html.write('      <TD align="left">' + '\n')
```

```
485         outfileinfo_html.write('      <A HREF="http://www.wunderground.com"><↩
                img src="./pic/wunderground_logo/wundergroundLogo_white_horz_small.↩
                png" border="0" alt="wunderground.com"></A>' + '\n')
486         outfileinfo_html.write('      </TD>' + '\n')
487         outfileinfo_html.write('      <TD align="right">' + '\n')
488         outfileinfo_html.write('      <FONT face="Lucida Grande,Verdana,Geneva,↩
                Lucida,Arial,Helvetica" color="#ffffff" size="2">Impressum:<BR />↩
                Michael Nolde<BR />Christian Albrechts - Universit&auml;t zu Kiel<↩
                BR />Geographisches Institut<BR />Ludewig-Meyn-Str. 14<BR />24118 ↩
                Kiel, Germany</FONT>' + '\n')
489         outfileinfo_html.write('      </TD>' + '\n')
490         outfileinfo_html.write('      </TR>' + '\n')
491         outfileinfo_html.write('      </TABLE' + '\n')
492         outfileinfo_html.write('    </TD>' + '\n')
493         outfileinfo_html.write('    </TR>' + '\n')
494         outfileinfo_html.write('    </TABLE>' + '\n')
495         outfileinfo_html.write('    ' + '\n')
496         outfileinfo_html.write('    ' + '\n')
497         outfileinfo_html.write('  </TD>' + '\n')
498         outfileinfo_html.write('  </TR>' + '\n')
499         outfileinfo_html.write('  </TABLE>' + '\n')
500         outfileinfo_html.write('  ' + '\n')
501         outfileinfo_html.write('</BODY>' + '\n')
502         outfileinfo_html.write('</HTML>' + '\n')
503
504         outfileinfo_html.close()
505         info = True
506
507
508
509
510     outfile_js = open(out_path + outfilename + ".js", 'w')
511
512     outfile_js.write('var map;' + '\n')
513     outfile_js.write('' + '\n')
514     outfile_js.write('function init() {' + '\n')
515     outfile_js.write('    map = new OpenLayers.Map({' + '\n')
```

```
516     outfile_js.write('        div: "map",' + '\n')
517     outfile_js.write('        projection: new OpenLayers.Projection("EPSG:900913")' + '\n')
518     outfile_js.write('    });' + '\n')
519     outfile_js.write('    ' + '\n')
520     outfile_js.write('    var osm = new OpenLayers.Layer.OSM();            ' + '\n')
521     outfile_js.write('    //var gmapreets = new OpenLayers.Layer.Google("Google Streets");' + '\n')
522     outfile_js.write('  //var gmap_satellite = new OpenLayers.Layer.Google("Google Satellite", {type: G_SATELLITE_MAP});' + '\n')
523     outfile_js.write('' + '\n')
524     outfile_js.write('  var gphy = new OpenLayers.Layer.Google("Google Physical", {type: google.maps.MapTypeId.TERRAIN});' + '\n')
525     outfile_js.write('  var gmap = new OpenLayers.Layer.Google("Google Streets", {numZoomLevels: 20});' + '\n')
526     outfile_js.write('  var ghyb = new OpenLayers.Layer.Google("Google Hybrid",{type: google.maps.MapTypeId.HYBRID, numZoomLevels: 20});' + '\n')
527     outfile_js.write('  var gsat = new OpenLayers.Layer.Google("Google Satellite", {type: google.maps.MapTypeId.SATELLITE, numZoomLevels: 22});' + '\n')
528     outfile_js.write('' + '\n')
529     outfile_js.write('  var attrib = new OpenLayers.Layer.Vector("Attribution",{"attribution":"<IMG src=./legend.png border=0 alt=legend>", displayInLayerSwitcher: false});' + '\n')
530     outfile_js.write('' + '\n')
531     outfile_js.write('  var dangerclass1 = new OpenLayers.Layer.Vector("very low", {' + '\n')
532     outfile_js.write('  visibility: true, ' + '\n')
533     outfile_js.write('  projection: map.displayProjection, ' + '\n')
534     outfile_js.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n')
535     outfile_js.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
536     outfile_js.write('        strokeColor: "#99ff66",' + '\n')
537     outfile_js.write('        strokeOpacity: 1,' + '\n')
538     outfile_js.write('        strokeWidth: 1,' + '\n')
539     outfile_js.write('        fillColor: "#99ff66",' + '\n')
```

```
540     outfile_js.write('        fillOpacity: 0.5,' + '\n')
541     outfile_js.write('        }}),' + '\n')
542     outfile_js.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
543     outfile_js.write('    url: "' + gmlmercin_path + datestr_forecast[i][0:4] +↩
            '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
        datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + datestr_forecast[↩
        i][6:8] + '_airports_pws_' + outfilename_index_short + '_900913_verylow↩
        .gml", ' + '\n')
544     outfile_js.write('    format: new OpenLayers.Format.GML({' + '\n')
545     outfile_js.write('      extractStyles: true, ' + '\n')
546     outfile_js.write('      extractAttributes: true})})});    ' + '\n')
547     outfile_js.write('' + '\n')
548     outfile_js.write('' + '\n')
549     outfile_js.write('  var dangerclass2 = new OpenLayers.Layer.Vector("low", {↩
        ' + '\n')
550     outfile_js.write('  visibility: true, ' + '\n')
551     outfile_js.write('  projection: map.displayProjection, ' + '\n')
552     outfile_js.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
        )
553     outfile_js.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
554     outfile_js.write('        strokeColor: "#ffff33",' + '\n')
555     outfile_js.write('        strokeOpacity: 1,' + '\n')
556     outfile_js.write('        strokeWidth: 1,' + '\n')
557     outfile_js.write('        fillColor: "#ffff33",' + '\n')
558     outfile_js.write('        fillOpacity: 0.5,' + '\n')
559     outfile_js.write('        }}),' + '\n')
560     outfile_js.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
561     outfile_js.write('    url: "' + gmlmercin_path + datestr_forecast[i][0:4] +↩
            '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
        datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + datestr_forecast[↩
        i][6:8] + '_airports_pws_' + outfilename_index_short + '_900913_low.gml↩
        ", ' + '\n')
562     outfile_js.write('    format: new OpenLayers.Format.GML({' + '\n')
563     outfile_js.write('      extractStyles: true, ' + '\n')
564     outfile_js.write('      extractAttributes: true})})});    ' + '\n')
565     outfile_js.write('' + '\n')
566     outfile_js.write('' + '\n')
```

```
567     outfile_js.write('  var dangerclass3 = new OpenLayers.Layer.Vector("↩
            moderate", {' + '\n')
568     outfile_js.write('  visibility: true, ' + '\n')
569     outfile_js.write('  projection: map.displayProjection, ' + '\n')
570     outfile_js.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
            )
571     outfile_js.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
572     outfile_js.write('        strokeColor: "#ffcc33",' + '\n')
573     outfile_js.write('        strokeOpacity: 1,' + '\n')
574     outfile_js.write('        strokeWidth: 1,' + '\n')
575     outfile_js.write('        fillColor: "#ffcc33",' + '\n')
576     outfile_js.write('        fillOpacity: 0.5,' + '\n')
577     outfile_js.write('        }}),' + '\n')
578     outfile_js.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
579     outfile_js.write('    url: "' + gmlmercin_path + datestr_forecast[i][0:4] +↩
             '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
            datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + datestr_forecast[↩
            i][6:8] + '_airports_pws_' + outfilename_index_short + '↩
            _900913_moderate.gml", ' + '\n')
580     outfile_js.write('    format: new OpenLayers.Format.GML({' + '\n')
581     outfile_js.write('      extractStyles: true, ' + '\n')
582     outfile_js.write('      extractAttributes: true})})});    ' + '\n')
583     outfile_js.write('' + '\n')
584     outfile_js.write('' + '\n')
585     outfile_js.write('  var dangerclass4 = new OpenLayers.Layer.Vector("high", ↩
            {' + '\n')
586     outfile_js.write('  visibility: true, ' + '\n')
587     outfile_js.write('  projection: map.displayProjection, ' + '\n')
588     outfile_js.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
            )
589     outfile_js.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
590     outfile_js.write('        strokeColor: "#fa5000",' + '\n')
591     outfile_js.write('        strokeOpacity: 1,' + '\n')
592     outfile_js.write('        strokeWidth: 1,' + '\n')
593     outfile_js.write('        fillColor: "#fa5000",' + '\n')
594     outfile_js.write('        fillOpacity: 0.5,' + '\n')
595     outfile_js.write('        }}),' + '\n')
```

```
596     outfile_js.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
597     outfile_js.write('    url: "' + gmlmercin_path + datestr_forecast[i][0:4] +↩
            '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
            datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + datestr_forecast[↩
            i][6:8] + '_airports_pws_' + outfilename_index_short + '_900913_high.↩
            gml", ' + '\n')
598     outfile_js.write('    format: new OpenLayers.Format.GML({' + '\n')
599     outfile_js.write('      extractStyles: true, ' + '\n')
600     outfile_js.write('      extractAttributes: true})})});    ' + '\n')
601     outfile_js.write('' + '\n')
602     outfile_js.write('' + '\n')
603
604     if outfilename_index_short.lower() == "fwi":
605       outfile_js.write('  var dangerclass5 = new OpenLayers.Layer.Vector("very ↩
              high", {' + '\n')
606       outfile_js.write('  visibility: true, ' + '\n')
607       outfile_js.write('  projection: map.displayProjection, ' + '\n')
608       outfile_js.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\↩
              n')
609       outfile_js.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n↩
              ')
610       outfile_js.write('        strokeColor: "#b40000",' + '\n')
611       outfile_js.write('        strokeOpacity: 1,' + '\n')
612       outfile_js.write('        strokeWidth: 1,' + '\n')
613       outfile_js.write('        fillColor: "#b40000",' + '\n')
614       outfile_js.write('        fillOpacity: 0.5,' + '\n')
615       outfile_js.write('        }}),' + '\n')
616       outfile_js.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
617       outfile_js.write('    url: "' + gmlmercin_path + datestr_forecast[i][0:4]↩
              + '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
              datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + ↩
              datestr_forecast[i][6:8] + '_airports_pws_' + outfilename_index_short↩
              + '_900913_veryhigh.gml", ' + '\n')
618       outfile_js.write('    format: new OpenLayers.Format.GML({' + '\n')
619       outfile_js.write('      extractStyles: true, ' + '\n')
620       outfile_js.write('      extractAttributes: true})})});    ' + '\n')
621       outfile_js.write('' + '\n')
```

```
622
623     outfile_js.write('' + '\n')
624     outfile_js.write('  var dangerclass6 = new OpenLayers.Layer.Vector("extreme↩
            ", {' + '\n')
625     outfile_js.write('  visibility: true, ' + '\n')
626     outfile_js.write('  projection: map.displayProjection, ' + '\n')
627     outfile_js.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
            )
628     outfile_js.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
629     outfile_js.write('        strokeColor: "#280523",' + '\n')
630     outfile_js.write('        strokeOpacity: 1,' + '\n')
631     outfile_js.write('        strokeWidth: 1,' + '\n')
632     outfile_js.write('        fillColor: "#280523",' + '\n')
633     outfile_js.write('        fillOpacity: 0.5,' + '\n')
634     outfile_js.write('        }}),' + '\n')
635     outfile_js.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
636     outfile_js.write('    url: "' + gmlmercin_path + datestr_forecast[i][0:4] +↩
             '/' + datestr_forecast[i][4:6] + '/' + 'sard_fwi_idw_' + ↩
            datestr_forecast[i][0:4] + datestr_forecast[i][4:6] + datestr_forecast[↩
            i][6:8] + '_airports_pws_' + outfilename_index_short + '_900913_extreme↩
            .gml", ' + '\n')
637     outfile_js.write('    format: new OpenLayers.Format.GML({' + '\n')
638     outfile_js.write('      extractStyles: true, ' + '\n')
639     outfile_js.write('      extractAttributes: true})})});    ' + '\n')
640     outfile_js.write('' + '\n')
641
642     if outfilename_index_short.lower() == "fwi":
643       outfile_js.write('    map.addLayers([osm, gmap, gsat, gphy, ghyb, attrib,↩
              dangerclass1, dangerclass2, dangerclass3, dangerclass4, dangerclass5↩
            , dangerclass6]);' + '\n')
644     else:
645       outfile_js.write('    map.addLayers([osm, gmap, gsat, gphy, ghyb, attrib,↩
              dangerclass1, dangerclass2, dangerclass3, dangerclass4, dangerclass6↩
            ]);' + '\n')
646
647     outfile_js.write('' + '\n')
648     outfile_js.write('    map.addControl(new OpenLayers.Control.LayerSwitcher()↩
```

```
        );' + '\n')
649     outfile_js.write('' + '\n')
650     outfile_js.write('    map.setCenter(new OpenLayers.LonLat↩
            (1000406.618302607,4880203.739580242), 8);' + '\n')
651     outfile_js.write('' + '\n')
652     outfile_js.write('}' + '\n')
653
654     outfile_js.close()
655
656
657
658 #WEB-APP
659
660 outfile_js_app = open(out_path + "applayers.js", 'w')
661
662 outfile_js_app.write('// API key for http://openlayers.org. Please get your own↩
        at' + '\n')
663 outfile_js_app.write('// http://bingmapsportal.com/ and use that instead.' + '\↩
        n')
664 outfile_js_app.write('var apiKey = "↩
        AqTGBsziZHIJYYxgivLBf0hVdrAk9mWO5cQcb8Yux8sW5M8c8opEC2lZqKR1ZZXf";' + '\n')
665 outfile_js_app.write('' + '\n')
666 outfile_js_app.write('var map;' + '\n')
667 outfile_js_app.write('' + '\n')
668 outfile_js_app.write('' + '\n')
669 outfile_js_app.write('var init = function () {' + '\n')
670 outfile_js_app.write('' + '\n')
671 outfile_js_app.write('' + '\n')
672 outfile_js_app.write('  var vector = new OpenLayers.Layer.Vector("Position", {↩
        displayInLayerSwitcher: false});' + '\n')
673 outfile_js_app.write('  var geolocate = new OpenLayers.Control.Geolocate({id: "↩
        locate-control", geolocationOptions: {enableHighAccuracy: false, maximumAge↩
        : 0, timeout: 7000}});' + '\n')
674 outfile_js_app.write('' + '\n')
675 outfile_js_app.write('  map = new OpenLayers.Map({' + '\n')
676 outfile_js_app.write('' + '\n')
677 outfile_js_app.write('    projection: new OpenLayers.Projection("EPSG:900913"),↩
```

```
    ' + '\n')
678 outfile_js_app.write('    displayProjection: new OpenLayers.Projection("EPSG:900913"),' + '\n')
679 outfile_js_app.write('    units: "m",' + '\n')
680 outfile_js_app.write('    //maxExtent: new OpenLayers.Bounds(1126728.964466007, 7234260.369663073, 1127018.977872739, 7234647.801024904),' + '\n')
681 outfile_js_app.write('    maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34, 20037508.34, 20037508.34),' + '\n')
682 outfile_js_app.write('        ' + '\n')
683 outfile_js_app.write('    div: "map",' + '\n')
684 outfile_js_app.write('    theme: null,' + '\n')
685 outfile_js_app.write('    numZoomLevels: 22,' + '\n')
686 outfile_js_app.write('    maxToomLevel: 22,' + '\n')
687 outfile_js_app.write('' + '\n')
688 outfile_js_app.write('' + '\n')
689 outfile_js_app.write('    controls: [' + '\n')
690 outfile_js_app.write('      new OpenLayers.Control.Attribution(),' + '\n')
691 outfile_js_app.write('      new OpenLayers.Control.TouchNavigation({dragPanOptions: {interval: 100, enableKinetic: true}}),' + '\n')
692 outfile_js_app.write('      geolocate' + '\n')
693 outfile_js_app.write('    ],' + '\n')
694 outfile_js_app.write('    ' + '\n')
695 outfile_js_app.write('    layers: [' + '\n')
696 outfile_js_app.write('             ' + '\n')
697 outfile_js_app.write('      //BASELAYER:' + '\n')
698 outfile_js_app.write('      //new OpenLayers.Layer.WMS("VMAP0", "http://vmap0.tiles.osgeo.org/wms/vmap0", {layers: "basic"}),' + '\n')
699 outfile_js_app.write('      //new OpenLayers.Layer.Google("Google Hybrid", {sphericalMercator:true, type: G_HYBRID_MAP, numZoomLevels: 22}),' + '\n')
700 outfile_js_app.write('      //new OpenLayers.Layer.Google("Google Physical", {sphericalMercator:true, type: G_PHYSICAL_MAP, numZoomLevels: 22}),' + '\n')
701 outfile_js_app.write('      //new OpenLayers.Layer.Google("Google Streets", {sphericalMercator:true, numZoomLevels: 22}),' + '\n')
702 outfile_js_app.write('      //new OpenLayers.Layer.Google("Google Satellite", {visibility: false, sphericalMercator:true, type: G_SATELLITE_MAP, numZoomLevels: 35}),      ' + '\n')
703 outfile_js_app.write('' + '\n')
```

```
704 outfile_js_app.write('      //new OpenLayers.Layer.Bing({key: apiKey, type: "↩
        Road", metadataParams: {mapVersion: "v1"}, name: "Bing Road", ↩
        transitionEffect: "resize"}),' + '\n')
705 outfile_js_app.write('      //new OpenLayers.Layer.Bing({key: apiKey, type: "↩
        AerialWithLabels", name: "Bing Aerial + Labels", transitionEffect: "resize↩
        "}), ' + '\n')
706 outfile_js_app.write('' + '\n')
707 outfile_js_app.write('      new OpenLayers.Layer.OSM("OpenStreetMap", null, {↩
        transitionEffect: "resize"}),' + '\n')
708 outfile_js_app.write('      new OpenLayers.Layer.Google("Google Satellite", {↩
        type: google.maps.MapTypeId.SATELLITE, numZoomLevels: 22}),' + '\n')
709 outfile_js_app.write('      new OpenLayers.Layer("Kein Hintergrund",{↩
        isBaseLayer:true, sphericalMercator:true, numZoomLevels: 35}),' + '\n')
710 outfile_js_app.write('      new OpenLayers.Layer.Bing({key: apiKey, type: "↩
        Aerial", name: "Bing Aerial", transitionEffect: "resize"}),' + '\n')
711 outfile_js_app.write('' + '\n')
712 outfile_js_app.write('      vector ' + '\n')
713 outfile_js_app.write('' + '\n')
714 outfile_js_app.write('    ],' + '\n')
715 outfile_js_app.write('' + '\n')
716 outfile_js_app.write('' + '\n')
717 outfile_js_app.write('    center: new OpenLayers.LonLat(1000406.618302607, ↩
        4880203.739580242),' + '\n')
718 outfile_js_app.write('    zoom: 8 //fuer GoogleMaps: zoom 20' + '\n')
719 outfile_js_app.write('    ' + '\n')
720 outfile_js_app.write('        ' + '\n')
721 outfile_js_app.write('    ' + '\n')
722 outfile_js_app.write('  });' + '\n')
723 outfile_js_app.write('' + '\n')
724 outfile_js_app.write('  ' + '\n')
725 outfile_js_app.write('  var attrib = new OpenLayers.Layer.Vector("Attribution",↩
       {"attribution":"<IMG src=./legend.png><A HREF=http://www.wunderground.com↩
       ><IMG src=./pic/wunderground_logo/wundergroundLogo_black_mini.png border=0 ↩
       alt=wunderground.com></A>", displayInLayerSwitcher: false});' + '\n')
726 outfile_js_app.write('  ' + '\n')
727 outfile_js_app.write('  var dangerclass1 = new OpenLayers.Layer.Vector("very ↩
       low", {' + '\n')
```

```
728 outfile_js_app.write('  visibility: true, ' + '\n')
729 outfile_js_app.write('  projection: map.displayProjection, ' + '\n')
730 outfile_js_app.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
        )
731 outfile_js_app.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
732 outfile_js_app.write('        strokeColor: "#99ff66",' + '\n')
733 outfile_js_app.write('        strokeOpacity: 1,' + '\n')
734 outfile_js_app.write('        strokeWidth: 1,' + '\n')
735 outfile_js_app.write('        fillColor: "#99ff66",' + '\n')
736 outfile_js_app.write('        fillOpacity: 0.5,' + '\n')
737 outfile_js_app.write('        }}),' + '\n')
738 outfile_js_app.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
739 outfile_js_app.write('    url: "' + gmlmercin_path + datestr_forecast[0][0:4] +↩
        '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_verylow.gml", ' + '\n')
740 outfile_js_app.write('    format: new OpenLayers.Format.GML({' + '\n')
741 outfile_js_app.write('      extractStyles: true, ' + '\n')
742 outfile_js_app.write('      extractAttributes: true})})});    ' + '\n')
743 outfile_js_app.write('' + '\n')
744 outfile_js_app.write('' + '\n')
745 outfile_js_app.write('  var dangerclass2 = new OpenLayers.Layer.Vector("low", {↩
        ' + '\n')
746 outfile_js_app.write('  visibility: true, ' + '\n')
747 outfile_js_app.write('  projection: map.displayProjection, ' + '\n')
748 outfile_js_app.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
        )
749 outfile_js_app.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
750 outfile_js_app.write('        strokeColor: "#ffff33",' + '\n')
751 outfile_js_app.write('        strokeOpacity: 1,' + '\n')
752 outfile_js_app.write('        strokeWidth: 1,' + '\n')
753 outfile_js_app.write('        fillColor: "#ffff33",' + '\n')
754 outfile_js_app.write('        fillOpacity: 0.5,' + '\n')
755 outfile_js_app.write('        }}),' + '\n')
756 outfile_js_app.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
757 outfile_js_app.write('    url: "' + gmlmercin_path + datestr_forecast[0][0:4] +↩
        '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
```

```
    [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
    _airports_pws_fwi_900913_low.gml", ' + '\n')
outfile_js_app.write('    format: new OpenLayers.Format.GML({' + '\n')
outfile_js_app.write('      extractStyles: true, ' + '\n')
outfile_js_app.write('      extractAttributes: true})})});    ' + '\n')
outfile_js_app.write('' + '\n')
outfile_js_app.write('' + '\n')
outfile_js_app.write('  var dangerclass3 = new OpenLayers.Layer.Vector("↩
    moderate", {' + '\n')
outfile_js_app.write('  visibility: true, ' + '\n')
outfile_js_app.write('  projection: map.displayProjection, ' + '\n')
outfile_js_app.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
    )
outfile_js_app.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
outfile_js_app.write('        strokeColor: "#ffcc33",' + '\n')
outfile_js_app.write('        strokeOpacity: 1,' + '\n')
outfile_js_app.write('        strokeWidth: 1,' + '\n')
outfile_js_app.write('        fillColor: "#ffcc33",' + '\n')
outfile_js_app.write('        fillOpacity: 0.5,' + '\n')
outfile_js_app.write('        }}),' + '\n')
outfile_js_app.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
outfile_js_app.write('    url: "' + gmlmercin_path + datestr_forecast[0][0:4] +↩
     '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
    [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
    _airports_pws_fwi_900913_moderate.gml", ' + '\n')
outfile_js_app.write('    format: new OpenLayers.Format.GML({' + '\n')
outfile_js_app.write('      extractStyles: true, ' + '\n')
outfile_js_app.write('      extractAttributes: true})})});    ' + '\n')
outfile_js_app.write('' + '\n')
outfile_js_app.write('' + '\n')
outfile_js_app.write('  var dangerclass4 = new OpenLayers.Layer.Vector("high", ↩
    {' + '\n')
outfile_js_app.write('  visibility: true, ' + '\n')
outfile_js_app.write('  projection: map.displayProjection, ' + '\n')
outfile_js_app.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
    )
outfile_js_app.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
```

```
786 outfile_js_app.write('        strokeColor: "#fa5000",' + '\n')
787 outfile_js_app.write('        strokeOpacity: 1,' + '\n')
788 outfile_js_app.write('        strokeWidth: 1,' + '\n')
789 outfile_js_app.write('        fillColor: "#fa5000",' + '\n')
790 outfile_js_app.write('        fillOpacity: 0.5,' + '\n')
791 outfile_js_app.write('        }}),' + '\n')
792 outfile_js_app.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
793 outfile_js_app.write('    url: "' + gmlmercin_path + datestr_forecast[0][0:4] +↩
        '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
    [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
    _airports_pws_fwi_900913_high.gml", ' + '\n')
794 outfile_js_app.write('    format: new OpenLayers.Format.GML({' + '\n')
795 outfile_js_app.write('      extractStyles: true, ' + '\n')
796 outfile_js_app.write('      extractAttributes: true})})});    ' + '\n')
797 outfile_js_app.write('' + '\n')
798 outfile_js_app.write('' + '\n')
799 outfile_js_app.write('  var dangerclass5 = new OpenLayers.Layer.Vector("very ↩
    high", {' + '\n')
800 outfile_js_app.write('  visibility: true, ' + '\n')
801 outfile_js_app.write('  projection: map.displayProjection, ' + '\n')
802 outfile_js_app.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n'↩
    )
803 outfile_js_app.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
804 outfile_js_app.write('        strokeColor: "#b40000",' + '\n')
805 outfile_js_app.write('        strokeOpacity: 1,' + '\n')
806 outfile_js_app.write('        strokeWidth: 1,' + '\n')
807 outfile_js_app.write('        fillColor: "#b40000",' + '\n')
808 outfile_js_app.write('        fillOpacity: 0.5,' + '\n')
809 outfile_js_app.write('        }}),' + '\n')
810 outfile_js_app.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
811 outfile_js_app.write('    url: "' + gmlmercin_path + datestr_forecast[0][0:4] +↩
        '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
    [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
    _airports_pws_fwi_900913_veryhigh.gml", ' + '\n')
812 outfile_js_app.write('    format: new OpenLayers.Format.GML({' + '\n')
813 outfile_js_app.write('      extractStyles: true, ' + '\n')
814 outfile_js_app.write('      extractAttributes: true})})});    ' + '\n')
```

```
815 outfile_js_app.write('' + '\n')
816 outfile_js_app.write('' + '\n')
817 outfile_js_app.write('  var dangerclass6 = new OpenLayers.Layer.Vector("extreme", {' + '\n')
818 outfile_js_app.write('  visibility: true, ' + '\n')
819 outfile_js_app.write('  projection: map.displayProjection, ' + '\n')
820 outfile_js_app.write('  strategies: [new OpenLayers.Strategy.Fixed()], ' + '\n')
821 outfile_js_app.write('  styleMap: new OpenLayers.StyleMap({"default":{' + '\n')
822 outfile_js_app.write('        strokeColor: "#280523",' + '\n')
823 outfile_js_app.write('        strokeOpacity: 1,' + '\n')
824 outfile_js_app.write('        strokeWidth: 1,' + '\n')
825 outfile_js_app.write('        fillColor: "#280523",' + '\n')
826 outfile_js_app.write('        fillOpacity: 0.5,' + '\n')
827 outfile_js_app.write('        }}),' + '\n')
828 outfile_js_app.write('  protocol: new OpenLayers.Protocol.HTTP({' + '\n')
829 outfile_js_app.write('    url: "' + gmlmercin_path + datestr_forecast[0][0:4] + '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast[0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '_airports_pws_fwi_900913_extreme.gml", ' + '\n')
830 outfile_js_app.write('    format: new OpenLayers.Format.GML({' + '\n')
831 outfile_js_app.write('      extractStyles: true, ' + '\n')
832 outfile_js_app.write('      extractAttributes: true})})});    ' + '\n')
833 outfile_js_app.write('' + '\n')
834 outfile_js_app.write('' + '\n')
835 outfile_js_app.write('' + '\n')
836 outfile_js_app.write('  ' + '\n')
837 outfile_js_app.write('      ' + '\n')
838 outfile_js_app.write('  //Beschriftung      ' + '\n')
839 outfile_js_app.write('  var renderer = OpenLayers.Util.getParameters(window.location.href).renderer;' + '\n')
840 outfile_js_app.write('  renderer = (renderer) ? [renderer] : OpenLayers.Layer.Vector.prototype.renderers;' + '\n')
841 outfile_js_app.write('      ' + '\n')
842 outfile_js_app.write('' + '\n')
843 outfile_js_app.write('  var vectorLayer = new OpenLayers.Layer.Vector("Labels",{' + '\n')
```

```
844 outfile_js_app.write('    visibility: false,' + '\n')
845 outfile_js_app.write('    isBaseLayer:false,' + '\n')
846 outfile_js_app.write('    styleMap: new OpenLayers.StyleMap({"default":{' + '\n↩
        ')
847 outfile_js_app.write('      strokeColor: "#00FF00",' + '\n')
848 outfile_js_app.write('      strokeOpacity: 1,' + '\n')
849 outfile_js_app.write('      strokeWidth: 0,' + '\n')
850 outfile_js_app.write('      fillColor: "#FF5500",' + '\n')
851 outfile_js_app.write('      fillOpacity: 0,' + '\n')
852 outfile_js_app.write('      pointRadius: 6,' + '\n')
853 outfile_js_app.write('      pointerEvents: "visiblePainted",' + '\n')
854 outfile_js_app.write('      label : "${name}",' + '\n')
855 outfile_js_app.write('      fontColor: "${favColor}",' + '\n')
856 outfile_js_app.write('      fontSize: "22px",' + '\n')
857 outfile_js_app.write('      fontFamily: "Courier New, monospace",' + '\n')
858 outfile_js_app.write('      fontWeight: "bold",' + '\n')
859 outfile_js_app.write('      labelAlign: "${align}",' + '\n')
860 outfile_js_app.write('      labelXOffset: "${xOffset}",' + '\n')
861 outfile_js_app.write('      labelYOffset: "${yOffset}"' + '\n')
862 outfile_js_app.write('    }}),' + '\n')
863 outfile_js_app.write('    renderers: renderer' + '\n')
864 outfile_js_app.write('  });' + '\n')
865 outfile_js_app.write('' + '\n')
866 outfile_js_app.write('  var point = new OpenLayers.Geometry.Point↩
        (1126966.56,7234477.80);' + '\n')
867 outfile_js_app.write('  var pointFeature = new OpenLayers.Feature.Vector(point)↩
        ; pointFeature.attributes = {name: "", age: 20, favColor: "blue", align: "↩
        cm"};' + '\n')
868 outfile_js_app.write('  var point1 = new OpenLayers.Geometry.Point↩
        (1126966,7234478);' + '\n')
869 outfile_js_app.write('  var point1Feature = new OpenLayers.Feature.Vector(↩
        point1); point1Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
         align: "cm"};' + '\n')
870 outfile_js_app.write('  var point2 = new OpenLayers.Geometry.Point↩
        (1126874,7234332);' + '\n')
871 outfile_js_app.write('  var point2Feature = new OpenLayers.Feature.Vector(↩
        point2); point2Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
```

```
        align: "cm"};' + '\n')
872 outfile_js_app.write('  var point3 = new OpenLayers.Geometry.Point↩
    (1126837,7234420);' + '\n')
873 outfile_js_app.write('  var point3Feature = new OpenLayers.Feature.Vector(↩
    point3); point3Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
        align: "cm"};' + '\n')
874 outfile_js_app.write('  var point4 = new OpenLayers.Geometry.Point↩
    (1126977,7234278);' + '\n')
875 outfile_js_app.write('  var point4Feature = new OpenLayers.Feature.Vector(↩
    point4); point4Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
        align: "cm"};' + '\n')
876 outfile_js_app.write('  var point5 = new OpenLayers.Geometry.Point↩
    (1126920,7234586);' + '\n')
877 outfile_js_app.write('  var point5Feature = new OpenLayers.Feature.Vector(↩
    point5); point5Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
        align: "cm"};' + '\n')
878 outfile_js_app.write('  var point6 = new OpenLayers.Geometry.Point↩
    (1126827,7234590);' + '\n')
879 outfile_js_app.write('  var point6Feature = new OpenLayers.Feature.Vector(↩
    point6); point6Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
        align: "cm"};' + '\n')
880 outfile_js_app.write('  var point7 = new OpenLayers.Geometry.Point↩
    (1126757,7234517);' + '\n')
881 outfile_js_app.write('  var point7Feature = new OpenLayers.Feature.Vector(↩
    point7); point7Feature.attributes = {name: "", age: 20, favColor: "yellow",↩
        align: "cm"};' + '\n')
882 outfile_js_app.write('' + '\n')
883 outfile_js_app.write('  map.addLayer(vectorLayer);' + '\n')
884 outfile_js_app.write('  vectorLayer.addFeatures([pointFeature, point1Feature, ↩
    point2Feature, point3Feature, point4Feature, point5Feature, point6Feature, ↩
    point7Feature]);' + '\n')
885 outfile_js_app.write('  //Ende Beschriftung' + '\n')
886 outfile_js_app.write('' + '\n')
887 outfile_js_app.write('  var style = {' + '\n')
888 outfile_js_app.write('      strokeColor: "#000",' + '\n')
889 outfile_js_app.write('      strokeOpacity: 0.8,' + '\n')
890 outfile_js_app.write('      strokeWidth: 5,' + '\n')
```

```
891 outfile_js_app.write('     pointRadius: 0,' + '\n')
892 outfile_js_app.write('     strokeLinecap : "square",' + '\n')
893 outfile_js_app.write('     strokeDashstyle: "3 8" ' + '\n')
894 outfile_js_app.write('  };' + '\n')
895 outfile_js_app.write('' + '\n')
896 outfile_js_app.write('  var style2 = {' + '\n')
897 outfile_js_app.write('    fillOpacity: 0.5,' + '\n')
898 outfile_js_app.write('    fillColor: "#ff0000",' + '\n')
899 outfile_js_app.write('    strokeColor: "#f00",' + '\n')
900 outfile_js_app.write('    strokeOpacity: 0.6,' + '\n')
901 outfile_js_app.write('    strokeWidth: 1' + '\n')
902 outfile_js_app.write('  }; ' + '\n')
903 outfile_js_app.write('' + '\n')
904 outfile_js_app.write('    map.addLayers([attrib, dangerclass1, dangerclass2, ↩
        dangerclass3, dangerclass4, dangerclass5, dangerclass6]);' + '\n')
905 outfile_js_app.write('' + '\n')
906 outfile_js_app.write('' + '\n')
907 outfile_js_app.write('  //var style = {' + '\n')
908 outfile_js_app.write('  //  fillOpacity: 0.1,' + '\n')
909 outfile_js_app.write('  //  fillColor: "#000",' + '\n')
910 outfile_js_app.write('  //  strokeColor: "#f00",' + '\n')
911 outfile_js_app.write('  //  strokeOpacity: 0.6' + '\n')
912 outfile_js_app.write('  //};' + '\n')
913 outfile_js_app.write('' + '\n')
914 outfile_js_app.write('' + '\n')
915 outfile_js_app.write('  ' + '\n')
916 outfile_js_app.write('  ' + '\n')
917 outfile_js_app.write('  geolocate.events.register("locationupdated",this,↩
        function(e) {' + '\n')
918 outfile_js_app.write('    vector.removeAllFeatures();' + '\n')
919 outfile_js_app.write('    vector.addFeatures([' + '\n')
920 outfile_js_app.write('      ' + '\n')
921 outfile_js_app.write('       ' + '\n')
922 outfile_js_app.write('      new OpenLayers.Feature.Vector(' + '\n')
923 outfile_js_app.write('        e.point,' + '\n')
924 outfile_js_app.write('        {},' + '\n')
925 outfile_js_app.write('        {' + '\n')
```

```
926 outfile_js_app.write('          graphicName: "cross",' + '\n')
927 outfile_js_app.write('          strokeColor: "#f00",' + '\n')
928 outfile_js_app.write('          strokeWidth: 2,' + '\n')
929 outfile_js_app.write('          fillOpacity: 0,' + '\n')
930 outfile_js_app.write('          pointRadius: 10' + '\n')
931 outfile_js_app.write('        }' + '\n')
932 outfile_js_app.write('      ),' + '\n')
933 outfile_js_app.write('      new OpenLayers.Feature.Vector(' + '\n')
934 outfile_js_app.write('        OpenLayers.Geometry.Polygon.createRegularPolygon(↩
        ' + '\n')
935 outfile_js_app.write('          new OpenLayers.Geometry.Point(e.point.x, e.↩
        point.y),' + '\n')
936 outfile_js_app.write('          e.position.coords.accuracy/2,' + '\n')
937 outfile_js_app.write('          50,' + '\n')
938 outfile_js_app.write('          0' + '\n')
939 outfile_js_app.write('        ),' + '\n')
940 outfile_js_app.write('        {},' + '\n')
941 outfile_js_app.write('        style2' + '\n')
942 outfile_js_app.write('      )' + '\n')
943 outfile_js_app.write('    ]);' + '\n')
944 outfile_js_app.write('    map.zoomToExtent(vector.getDataExtent());' + '\n')
945 outfile_js_app.write('' + '\n')
946 outfile_js_app.write('' + '\n')
947 outfile_js_app.write('  });' + '\n')
948 outfile_js_app.write('' + '\n')
949 outfile_js_app.write('' + '\n')
950 outfile_js_app.write('' + '\n')
951 outfile_js_app.write('' + '\n')
952 outfile_js_app.write('' + '\n')
953 outfile_js_app.write('' + '\n')
954 outfile_js_app.write(' var controls;' + '\n')
955 outfile_js_app.write('' + '\n')
956 outfile_js_app.write('      select = new OpenLayers.Control.SelectFeature([↩
        dangerclass1]);' + '\n')
957 outfile_js_app.write('       ' + '\n')
958 outfile_js_app.write('      dangerclass1.events.on({"featureselected": ↩
        onFeatureSelect, "featureunselected": onFeatureUnselect});' + '\n')
```

```
959 outfile_js_app.write(' ' + '\n')
960 outfile_js_app.write('      map.addControl(select);' + '\n')
961 outfile_js_app.write('      select.activate();    ' + '\n')
962 outfile_js_app.write('    ' + '\n')
963 outfile_js_app.write('    ' + '\n')
964 outfile_js_app.write('    function onPopupClose(evt) {' + '\n')
965 outfile_js_app.write('      select.unselectAll();' + '\n')
966 outfile_js_app.write('    }' + '\n')
967 outfile_js_app.write('    function onFeatureSelect(event) {' + '\n')
968 outfile_js_app.write('      var feature = event.feature;' + '\n')
969 outfile_js_app.write('      var selectedFeature = feature;' + '\n')
970 outfile_js_app.write('      var content = "<h2>"+feature.attributes.name + "</↩
        h2>" + feature.attributes.description;' + '\n')
971 outfile_js_app.write('      var popup = new OpenLayers.Popup.FramedCloud("↩
        chicken",' + '\n')
972 outfile_js_app.write('        feature.geometry.getBounds().getCenterLonLat(),' ↩
        + '\n')
973 outfile_js_app.write('        new OpenLayers.Size(100,100),' + '\n')
974 outfile_js_app.write('        content,' + '\n')
975 outfile_js_app.write('        null, true, onPopupClose' + '\n')
976 outfile_js_app.write('      );' + '\n')
977 outfile_js_app.write('      ' + '\n')
978 outfile_js_app.write('' + '\n')
979 outfile_js_app.write('      feature.popup = popup;' + '\n')
980 outfile_js_app.write('      map.addPopup(popup);' + '\n')
981 outfile_js_app.write('    }' + '\n')
982 outfile_js_app.write('    function onFeatureUnselect(event) {' + '\n')
983 outfile_js_app.write('      var feature = event.feature;' + '\n')
984 outfile_js_app.write('      if(feature.popup) {' + '\n')
985 outfile_js_app.write('        map.removePopup(feature.popup);' + '\n')
986 outfile_js_app.write('        feature.popup.destroy();' + '\n')
987 outfile_js_app.write('        delete feature.popup;' + '\n')
988 outfile_js_app.write('      }' + '\n')
989 outfile_js_app.write('    }' + '\n')
990 outfile_js_app.write('' + '\n')
991 outfile_js_app.write('' + '\n')
992 outfile_js_app.write('};' + '\n')
```

```
993
994 outfile_js_app.close()
995
996 return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + out_path + '↩
        fwi_extreme_900913.shp' + ' ' + wmsgmlin_path + datestr_forecast[0][0:4] + ↩
        '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_extreme.gml', shell=True)
997 return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + out_path + '↩
        fwi_veryhigh_900913.shp' + ' ' + wmsgmlin_path + datestr_forecast[0][0:4] +↩
         '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_veryhigh.gml', shell=True)
998 return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + out_path + '↩
        fwi_high_900913.shp' + ' ' + wmsgmlin_path + datestr_forecast[0][0:4] + '/'↩
         + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_high.gml', shell=True)
999 return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + out_path + '↩
        fwi_moderate_900913.shp' + ' ' + wmsgmlin_path + datestr_forecast[0][0:4] +↩
         '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_moderate.gml', shell=True)
1000 return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + out_path + '↩
        fwi_low_900913.shp' + ' ' + wmsgmlin_path + datestr_forecast[0][0:4] + '/' ↩
        + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_low.gml', shell=True)
1001 return_code = subprocess.call('ogr2ogr -f "ESRI Shapefile" ' + out_path + '↩
        fwi_verylow_900913.shp' + ' ' + wmsgmlin_path + datestr_forecast[0][0:4] + ↩
        '/' + datestr_forecast[0][4:6] + '/' + 'sard_fwi_idw_' + datestr_forecast↩
        [0][0:4] + datestr_forecast[0][4:6] + datestr_forecast[0][6:8] + '↩
        _airports_pws_fwi_900913_verylow.gml', shell=True)
1002
1003 return_code = subprocess.call('ogr2ogr -t_srs EPSG:4326 -s_srs EPSG:900913 ' + ↩
        out_path + 'fwi_extreme.shp ' + out_path + 'fwi_extreme_900913.shp', shell=↩
        True)
```

```
1004 return_code = subprocess.call('ogr2ogr -t_srs EPSG:4326 -s_srs EPSG:900913 ' + ↩
         out_path + 'fwi_veryhigh.shp ' + out_path + 'fwi_veryhigh_900913.shp', ↩
         shell=True)
1005 return_code = subprocess.call('ogr2ogr -t_srs EPSG:4326 -s_srs EPSG:900913 ' + ↩
         out_path + 'fwi_high.shp ' + out_path + 'fwi_high_900913.shp', shell=True)
1006 return_code = subprocess.call('ogr2ogr -t_srs EPSG:4326 -s_srs EPSG:900913 ' + ↩
         out_path + 'fwi_moderate.shp ' + out_path + 'fwi_moderate_900913.shp', ↩
         shell=True)
1007 return_code = subprocess.call('ogr2ogr -t_srs EPSG:4326 -s_srs EPSG:900913 ' + ↩
         out_path + 'fwi_low.shp ' + out_path + 'fwi_low_900913.shp', shell=True)
1008 return_code = subprocess.call('ogr2ogr -t_srs EPSG:4326 -s_srs EPSG:900913 ' + ↩
         out_path + 'fwi_verylow.shp ' + out_path + 'fwi_verylow_900913.shp', shell=↩
         True)
```

A.2 R scripts

A.2.1 idw.R

```
1
2  #  pkgs <- c('akima', 'spgrass6', 'RODBC', 'VR', 'gstat', 'maptools')
3  #  install.packages(pkgs, dependencies=TRUE, type='source')
4
5
6  rm(list = ls())
7
8  args<-commandArgs(TRUE)
9  demin = toString(args[1])
10 csvin = toString(args[2])
11
12 csvout_avgtemp_nn = paste(toString(args[3]),"avgtemp_nn.csv", sep="")
13 demout_avgtemp_nn = paste(toString(args[3]),"avgtemp_nn.asc", sep="")
14 csvout_precip24all = paste(toString(args[3]),"precip24all.csv", sep="")
15 demout_precip24all = paste(toString(args[3]),"precip24all.asc", sep="")
16 csvout_tempnoon_nn = paste(toString(args[3]),"tempnoon_nn.csv", sep="")
17 demout_tempnoon_nn = paste(toString(args[3]),"tempnoon_nn.asc", sep="")
18 csvout_humidnoon = paste(toString(args[3]),"humidnoon.csv", sep="")
19 demout_humidnoon = paste(toString(args[3]),"humidnoon.asc", sep="")
20 csvout_windkmhnoon = paste(toString(args[3]),"windkmhnoon.csv", sep="")
21 demout_windkmhnoon = paste(toString(args[3]),"windkmhnoon.asc", sep="")
22
23
24 #AVGTEMP_NN,PRECIP24ALL,TEMPNOON_NN,HUMIDNOON,WINDKMHNOON
25
26 nodata = toString(args[4])
27 projstring = toString(args[5])
28
29 #+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
30
31 #library(grid)
32 library(gstat)
33
34
35 #setwd("/home/mic/Desktop/idw")
```

```
36
37 d = read.csv(csvin, na.strings = nodata)
38 coordinates(d) = ~ LON + LAT
39 proj4string(d) = CRS(projstring)
40
41 dem1 = read.asciigrid(demin)
42 proj4string(dem1) = CRS(projstring)
43
44
45 idw_avgtemp_nn.out = idw(AVGTEMP_NN ~ 1, d[!is.na(d$AVGTEMP_NN),], newdata = ↩
       dem1, idp = 2.0)
46 write.csv(row.names=TRUE, file = csvout_avgtemp_nn, idw_avgtemp_nn.out$var1.↩
       pred)
47 #write.asciigrid(idw_avgtemp_nn.out, attr = 1, na.value = -9999, demout_avgtemp↩
       _nn)
48
49 idw_precip24all.out = idw(PRECIP24ALL ~ 1, d[!is.na(d$PRECIP24ALL),], newdata =↩
        dem1, idp = 2.0)
50 write.csv(row.names=TRUE, file = csvout_precip24all, idw_precip24all.out$var1.↩
       pred)
51 #write.asciigrid(idw_precip24all.out, attr = 1, na.value = -9999, demout_↩
       precip24all)
52
53 idw_tempnoon_nn.out = idw(TEMPNOON_NN ~ 1, d[!is.na(d$TEMPNOON_NN),], newdata =↩
        dem1, idp = 2.0)
54 write.csv(row.names=TRUE, file = csvout_tempnoon_nn, idw_tempnoon_nn.out$var1.↩
       pred)
55 #write.asciigrid(idw_tempnoon_nn.out, attr = 1, na.value = -9999, demout_↩
       tempnoon_nn)
56
57 idw_humidnoon.out = idw(HUMIDNOON ~ 1, d[!is.na(d$HUMIDNOON),], newdata = dem1,↩
        idp = 2.0)
58 write.csv(row.names=TRUE, file = csvout_humidnoon, idw_humidnoon.out$var1.pred)
59 #write.asciigrid(idw_humidnoon.out, attr = 1, na.value = -9999, demout_↩
       humidnoon)
60
61 idw_windkmhnoon.out = idw(WINDKMHNOON ~ 1, d[!is.na(d$WINDKMHNOON),], newdata =↩
```

```
     dem1, idp = 2.0)
62 write.csv(row.names=TRUE, file = csvout_windkmhnoon, idw_windkmhnoon.out$var1.↩
     pred)
63 #write.asciigrid(idw_windkmhnoon.out, attr = 1, na.value = -9999, demout_↩
     windkmhnoon)
```

A.2.2 fwiidw.R

```
1
2  #  pkgs <- c('akima', 'spgrass6', 'RODBC', 'VR', 'gstat', 'maptools')
3  #  install.packages(pkgs, dependencies=TRUE, type='source')
4
5
6  rm(list = ls())
7
8  args<-commandArgs(TRUE)
9  demin = toString(args[1])
10 csvin = toString(args[2])
11
12
13 csvout_ffmc = paste(toString(args[3]),"_ffmc.csv", sep="")
14 demout_ffmc = paste(toString(args[3]),"_ffmc.asc", sep="")
15 csvout_dmc = paste(toString(args[3]),"_dmc.csv", sep="")
16 demout_dmc = paste(toString(args[3]),"_dmc.asc", sep="")
17 csvout_dc = paste(toString(args[3]),"_dc.csv", sep="")
18 demout_dc = paste(toString(args[3]),"_dc.asc", sep="")
19 csvout_isi = paste(toString(args[3]),"_isi.csv", sep="")
20 demout_isi = paste(toString(args[3]),"_isi.asc", sep="")
21 csvout_bui = paste(toString(args[3]),"_bui.csv", sep="")
22 demout_bui = paste(toString(args[3]),"_bui.asc", sep="")
23 csvout_fwi = paste(toString(args[3]),"_fwi.csv", sep="")
24 demout_fwi = paste(toString(args[3]),"_fwi.asc", sep="")
25
26 #AVGTEMP_NN,PRECIP24ALL,TEMPNOON_NN,HUMIDNOON,WINDKMHNOON
27
28 nodata = toString(args[4])
29 projstring = toString(args[5])
```

```
30
31 #+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
32
33 #library(grid)
34 library(gstat)
35
36
37 #setwd("/home/mic/Desktop/idw")
38
39 d = read.csv(csvin, na.strings = nodata)
40 coordinates(d) = ~ LON + LAT
41 proj4string(d) = CRS(projstring)
42
43 dem1 = read.asciigrid(demin)
44 proj4string(dem1) = CRS(projstring)
45
46
47 idw_ffmc.out = idw(FFMC_CLASS ~ 1, d[!is.na(d$FFMC_CLASS),], newdata = dem1, ↩
       idp = 2.0)
48 write.csv(row.names=TRUE, file = csvout_ffmc, idw_ffmc.out$var1.pred)
49 write.asciigrid(idw_ffmc.out, attr = 1, na.value = -9999, demout_ffmc)
50
51 idw_dmc.out = idw(DMC_CLASS ~ 1, d[!is.na(d$DMC_CLASS),], newdata = dem1, idp =↩
       2.0)
52 write.csv(row.names=TRUE, file = csvout_dmc, idw_dmc.out$var1.pred)
53 write.asciigrid(idw_dmc.out, attr = 1, na.value = -9999, demout_dmc)
54
55 idw_dc.out = idw(DC_CLASS ~ 1, d[!is.na(d$DC_CLASS),], newdata = dem1, idp = ↩
       2.0)
56 write.csv(row.names=TRUE, file = csvout_dc, idw_dc.out$var1.pred)
57 write.asciigrid(idw_dc.out, attr = 1, na.value = -9999, demout_dc)
58
59 idw_isi.out = idw(ISI_CLASS ~ 1, d[!is.na(d$ISI_CLASS),], newdata = dem1, idp =↩
       2.0)
60 write.csv(row.names=TRUE, file = csvout_isi, idw_isi.out$var1.pred)
61 write.asciigrid(idw_isi.out, attr = 1, na.value = -9999, demout_isi)
62
```

```
63 idw_bui.out = idw(BUI_CLASS ~ 1, d[!is.na(d$BUI_CLASS),], newdata = dem1, idp =↩
       2.0)
64 write.csv(row.names=TRUE, file = csvout_bui, idw_bui.out$var1.pred)
65 write.asciigrid(idw_bui.out, attr = 1, na.value = -9999, demout_bui)
66
67 idw_fwi.out = idw(FWI_CLASS ~ 1, d[!is.na(d$FWI_CLASS),], newdata = dem1, idp =↩
       2.0)
68 write.csv(row.names=TRUE, file = csvout_fwi, idw_fwi.out$var1.pred)
69 write.asciigrid(idw_fwi.out, attr = 1, na.value = -9999, demout_fwi)
```

A.3 Shell scripts

A.3.1 0200b_live_yesterday.sh

```
1  #!/bin/bash
2
3  #GETDATE
4  python getdate.py out=dates.tmp out_path=./
5
6  #READOUT YESTERDAY'S DATE FROM 'getdate.py' OUTPUT
7  datestr=0;
8  datestr_yesterday=0;
9  value=-1;
10
11 while read datestr;
12 do
13 if [ $value = '-1' ]
14 then
15 datestr_yesterday=$datestr
16 fi
17
18 value=`expr $value + 1`;
19 done < dates.tmp
20
21
22 datestr_yesterday="20120906"
```

```
23
24
25 #DOWNLOAD
26 #python wunderground_api_download.py \
27 # stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET,IORISTAN2,↩
       IORISTAN4,IORISTAN5,ISARDEGN5,ISARDEGN6,ISARDEGN8,ISARDEGN9,ISARDEGN10,↩
       ISARDEGN15,ISARDEGN17,ISARDEGN22,ISARDEGN27,ISSPORTO1 \
28 # reg=sard \
29 # s_date=${datestr_yesterday} \
30 # e_date=${datestr_yesterday} \
31 # out_path=./../data/weather/wunderground_2011/download/json/sard/ \
32 # feature=history \
33 # sleeptime=240
34
35
36 #EXTRACT
37 python wunderground_api_extract_history.py \
38   stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET,IORISTAN2,↩
        IORISTAN4,IORISTAN5,ISARDEGN5,ISARDEGN6,ISARDEGN8,ISARDEGN9,ISARDEGN10,↩
        ISARDEGN15,ISARDEGN17,ISARDEGN22,ISARDEGN27,ISSPORTO1 \
39   date=${datestr_yesterday} \
40   in_path=./../data/weather/wunderground_2011/download/json/sard/ \
41   out_path=./../data/weather/wunderground_2011/combined/sard/ \
42   out_praefix="sard" \
43   out_suffix="_airports_pws"
44
45
46 #CORRECT
47 python wunderground_corr.py \
48   s_date=${datestr_yesterday} \
49   e_date=${datestr_yesterday} \
50   in_path=./../data/weather/wunderground_2011/combined/sard/ \
51   out_path=./../data/weather/wunderground_2011/corrected/sard/ \
52   in_praefix="sard_" \
53   in_suffix="_airports_pws"
54
55
```

```
#IDW-INTERPOLATION
python wunderground_idw.py \
  reg=sard res=3000 \
  indem=dem_srtm41_sard_clipreg_32632_res3000_buf7500_nodata.asc \
  indem_path=../data/elevation/srtm_90m/sard/ \
  incsv_praefix=sard_ \
  incsv_suffix=_airports_pws_corr \
  s_date=${datestr_yesterday} \
  e_date=${datestr_yesterday} \
  incsv_path=./../data/weather/wunderground_2011/corrected/sard/ \
  out_path=./../data/weather/wunderground_2011/idw/sard/ \
  out_praefix=sard_idw_ \
  nodata=-999 \
  projstring=+proj=utm+zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs


#PYFWI-FORMAT
python wunderground_splt.py \
  incsv_praefix=sard_idw_ \
  s_date=${datestr_yesterday} \
  e_date=${datestr_yesterday} \
  incsv_path=./../data/weather/wunderground_2011/idw/sard/ \
  instart=sard_idw_corr_startdate.csv \
  instart_path=../data/weather/wunderground_2011/startdate/ \
  out_path=./../data/weather/wunderground_2011/fwi/sard/singledays/ \
  out_praefix=sard_idw_ \
  outcsv_suffix=

python wunderground_splt_append.py \
  stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET,IORISTAN2,↩
      IORISTAN4,IORISTAN5,ISARDEGN5,ISARDEGN6,ISARDEGN8,ISARDEGN9,ISARDEGN10,↩
      ISARDEGN15,ISARDEGN17,ISARDEGN22,ISARDEGN27,ISSPORTO1 \
  s_date=${datestr_yesterday} \
  e_date=${datestr_yesterday} \
  in_path=./../data/weather/wunderground_2011/fwi/sard/singledays/ \
  out_path=./../data/weather/wunderground_2011/fwi/sard/ \
  in_praefix=sard_idw_ \
```

```
91    out_praefix=sard_idw_ out_suffix="_all"
92
93  #PYFWI
94  FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_IORISTAN2_all.csv
95  FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_IORISTAN4_all.csv
96  FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_IORISTAN5_all.csv
97  FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN5_all.csv
98  FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN6_all.csv
99  FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN8_all.csv
100 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN9_all.csv
101 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN10_all.csv
102 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN15_all.csv
103 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN17_all.csv
104 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN22_all.csv
105 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISARDEGN27_all.csv
106 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_ISSPORTO1_all.csv
107 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEA_all.csv
108 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEB_all.csv
109 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEC_all.csv
110 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIED_all.csv
111 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEE_all.csv
112 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEF_all.csv
113 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEH_all.csv
114 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEO_all.csv
115 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIER_all.csv
116 FWI.py ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIET_all.csv
117
118 #COMBINE
119 python wunderground_fwicomb_cut.py \
120   stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET,IORISTAN2,↩
        IORISTAN4,IORISTAN5,ISARDEGN5,ISARDEGN6,ISARDEGN8,ISARDEGN9,ISARDEGN10,↩
        ISARDEGN15,ISARDEGN17,ISARDEGN22,ISARDEGN27,ISSPORTO1 \
121   s_date=${datestr_yesterday} \
122   e_date=${datestr_yesterday} \
123   in_path=./../data/weather/wunderground_2011/fwi/sard/ \
124   out_path=./../data/weather/wunderground_2011/fwi/sard/singledays/ \
125   in_praefix=sard_idw_  \
```

```
126   in_suffix="_all" \
127   out_praefix=sard_idw_singleday_
128
129 python wunderground_fwicomb.py \
130   stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET,IORISTAN2,↩
          IORISTAN4,IORISTAN5,ISARDEGN5,ISARDEGN6,ISARDEGN8,ISARDEGN9,ISARDEGN10,↩
          ISARDEGN15,ISARDEGN17,ISARDEGN22,ISARDEGN27,ISSPORTO1 \
131   s_date=${datestr_yesterday} \
132   e_date=${datestr_yesterday} \
133   in_praefix=sard_idw_singleday_ \
134   in_path=./../data/weather/wunderground_2011/fwi/sard/singledays/ \
135   out_path=./../data/weather/wunderground_2011/fwicomb/sard/ \
136   out_praefix=sard_fwi_ \
137   out_suffix=_airports_pws
138
139
140 #FWI-IDW
141 python wunderground_fwiidw.py \
142   indem=dem_srtm41_sard_clipreg_32632_res3000_buf7500_nodata.asc \
143   indem_path=../data/elevation/srtm_90m/sard/ \
144   incsv_path=./../data/weather/wunderground_2011/fwicomb/sard/ \
145   incsv_praefix=sard_fwi_ \
146   incsv_suffix=_airports_pws \
147   s_date=${datestr_yesterday} \
148   e_date=${datestr_yesterday} \
149   out_path=./../data/weather/wunderground_2011/fwiidw/sard/ \
150   out_path_merc=./../data/weather/wunderground_2011/fwiidwmerc/sard/ \
151   out_praefix=sard_fwi_idw_ \
152   out_suffix=_airports_pws \
153   nodata=-999 \
154   projstring=+proj=utm+zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
155
156 rm -R ./../data/weather/wunderground_2011/fwi/sard/singledays/
157 mkdir ./../data/weather/wunderground_2011/fwi/sard/singledays
```

A.3.2 0200c_live_forecast.sh

```
#!/bin/bash

#GETDATE
python getdate.py out=dates.tmp out_path=./

#READOUT DATES OF YESTERDAY, TODAY AND NEXT 9 DAYS FROM 'getdate.py' OUTPUT
datestr=0;
datestr_yesterday=0;
datestr_day0=0;
datestr_day1=0;
datestr_day2=0;
datestr_day3=0;
datestr_day4=0;
datestr_day5=0;
datestr_day6=0;
datestr_day7=0;
datestr_day8=0;
datestr_day9=0;
value=-1;


while read datestr;
do


if [ $value = '-1' ]
then
datestr_yesterday=$datestr
fi

if [ $value = '0' ]
then
datestr_day0=$datestr
fi

```

```
36 if [ $value = '1' ]
37 then
38 datestr_day1=$datestr
39 fi
40
41 if [ $value = '2' ]
42 then
43 datestr_day2=$datestr
44 fi
45
46 if [ $value = '3' ]
47 then
48 datestr_day3=$datestr
49 fi
50
51 if [ $value = '4' ]
52 then
53 datestr_day4=$datestr
54 fi
55
56 if [ $value = '5' ]
57 then
58 datestr_day5=$datestr
59 fi
60
61 if [ $value = '6' ]
62 then
63 datestr_day6=$datestr
64 fi
65
66 if [ $value = '7' ]
67 then
68 datestr_day7=$datestr
69 fi
70
71 if [ $value = '8' ]
72 then
```

```
73  datestr_day8=$datestr
74  fi
75
76  if [ $value = '9' ]
77  then
78  datestr_day9=$datestr
79  fi
80
81  value=`expr $value + 1`;
82
83  done < dates.tmp
84
85
86  #datestr_yesterday="20120929"
87  #datestr_day0="20120930"
88  #datestr_day1="20121001"
89  #datestr_day2="20121002"
90  #datestr_day3="20121003"
91  #datestr_day4="20121004"
92  #datestr_day5="20121005"
93  #datestr_day6="20121006"
94  #datestr_day7="20121007"
95  #datestr_day8="20121008"
96  #datestr_day9="20121009"
97
98
99
100 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEA_all.csv ./../data↩
        /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.csv
101 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEB_all.csv ./../data↩
        /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.csv
102 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEC_all.csv ./../data↩
        /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.csv
103 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIED_all.csv ./../data↩
        /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.csv
104 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEE_all.csv ./../data↩
        /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.csv
```

```
105 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEF_all.csv ./../data↩
    /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.csv
106 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEH_all.csv ./../data↩
    /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.csv
107 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIEO_all.csv ./../data↩
    /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.csv
108 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIER_all.csv ./../data↩
    /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.csv
109 cp ./../data/weather/wunderground_2011/fwi/sard/sard_idw_LIET_all.csv ./../data↩
    /weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.csv
110
111 #DOWNLOAD
112
113 python wunderground_api_download.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH↩
    ,LIEO,LIER,LIET reg=sard s_date=${datestr_day0} e_date=${datestr_day0} out_↩
    path=./../data/weather/wunderground_2011_forecast/download/json/sard/ ↩
    feature=forecast10day sleeptime=240
114
115
116
117 #EXTRACT
118 python wunderground_api_extract_forecast10day.py stations=LIEA,LIEB,LIEC,LIED,↩
    LIEE,LIEF,LIEH,LIEO,LIER,LIET date=${datestr_day0} in_path=./../data/↩
    weather/wunderground_2011_forecast/download/json/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/combined/sard/ out_praefix="sard" out_↩
    suffix="_airports_pws"
119
120
121 #CORRECT
122 python wunderground_corr.py s_date=${datestr_day0} e_date=${datestr_day0} in_↩
    path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=.↩
    /../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="↩
    sard_" in_suffix="_airports_pws"
123 python wunderground_corr.py s_date=${datestr_day1} e_date=${datestr_day1} in_↩
    path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=.↩
    /../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="↩
    sard_" in_suffix="_airports_pws"
```

```
124 python wunderground_corr.py s_date=${datestr_day2} e_date=${datestr_day2} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
125 python wunderground_corr.py s_date=${datestr_day3} e_date=${datestr_day3} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
126 python wunderground_corr.py s_date=${datestr_day4} e_date=${datestr_day4} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
127 python wunderground_corr.py s_date=${datestr_day5} e_date=${datestr_day5} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
128 python wunderground_corr.py s_date=${datestr_day6} e_date=${datestr_day6} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
129 python wunderground_corr.py s_date=${datestr_day7} e_date=${datestr_day7} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
130 python wunderground_corr.py s_date=${datestr_day8} e_date=${datestr_day8} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
131 python wunderground_corr.py s_date=${datestr_day9} e_date=${datestr_day9} in_path=./../data/weather/wunderground_2011_forecast/combined/sard/ out_path=./../data/weather/wunderground_2011_forecast/corrected/sard/ in_praefix="sard_" in_suffix="_airports_pws"
132
133
134 #IDW-INTERPOLATION
135 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard
```

```
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day0↩
    } e_date=${datestr_day0} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
136 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day1↩
    } e_date=${datestr_day1} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
137 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day2↩
    } e_date=${datestr_day2} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
138 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day3↩
    } e_date=${datestr_day3} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
139 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day4↩
    } e_date=${datestr_day4} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
140 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day5↩
    } e_date=${datestr_day5} incsv_path=./../data/weather/wunderground_2011_↩
```

```
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
141 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day6↩
    } e_date=${datestr_day6} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
142 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day7↩
    } e_date=${datestr_day7} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
143 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day8↩
    } e_date=${datestr_day8} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
144 python wunderground_idw.py reg=sard res=3000 indem=dem_srtm41_sard_clipreg_↩
    32632_res3000_buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard↩
    / incsv_praefix=sard_ incsv_suffix=_airports_pws_corr s_date=${datestr_day9↩
    } e_date=${datestr_day9} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/corrected/sard/ out_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ out_praefix=sard_idw_ nodata=-999 projstring=+proj=utm+↩
    zone=32+ellps=WGS84+datum=WGS84+units=m+no_defs
145
146
147 #PYFWI-FORMAT
148 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day0} e_↩
    date=${datestr_day0} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
```

```
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
149 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day0} e_date=${datestr_day0} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
150 #PYFWI
151 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
152 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
    csv
153 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
154 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
155 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
156 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
    csv
157 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
158 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
159 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
160 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
    csv
161 #COMBINE
162 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day0} e_date=${datestr_day0} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
    _  in_suffix="_all" out_praefix=sard_idw_singleday_
163 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
    ,LIER,LIET s_date=${datestr_day0} e_date=${datestr_day0} in_praefix=sard_↩
```

```
    idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
    sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
    fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
164
165
166
167 #PYFWI-FORMAT
168 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day1} e_↩
    date=${datestr_day1} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
169 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day1} e_date=${datestr_day1} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
170 #PYFWI
171 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
172 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
    csv
173 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
174 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
175 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
176 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
    csv
177 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
178 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
179 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
```

```
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.csv
#COMBINE
python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET s_date=${datestr_day1} e_date=${datestr_day1} in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw_  in_suffix="_all" out_praefix=sard_idw_singleday_
python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET s_date=${datestr_day1} e_date=${datestr_day1} in_praefix=sard_idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws


#PYFWI-FORMAT
python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day2} e_date=${datestr_day2} incsv_path=./../data/weather/wunderground_2011_forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data/weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ outcsv_suffix=
python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET s_date=${datestr_day2} e_date=${datestr_day2} in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_praefix=sard_idw_ out_suffix="_all"
#PYFWI
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.csv
```

```
195 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.csv
196 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.csv
197 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.csv
198 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.csv
199 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.csv
200 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.csv
201 #COMBINE
202 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET s_date=${datestr_day2} e_date=${datestr_day2} in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw_  in_suffix="_all" out_praefix=sard_idw_singleday_
203 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET s_date=${datestr_day2} e_date=${datestr_day2} in_praefix=sard_idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
204
205
206 #PYFWI-FORMAT
207 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day3} e_date=${datestr_day3} incsv_path=./../data/weather/wunderground_2011_forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data/weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ outcsv_suffix=
208 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO,LIER,LIET s_date=${datestr_day3} e_date=${datestr_day3} in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_praefix=sard_idw_ out_suffix="_all"
```

```
209 #PYFWI
210 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
        csv
211 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
        csv
212 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
        csv
213 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
        csv
214 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
        csv
215 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
        csv
216 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
        csv
217 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
        csv
218 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
        csv
219 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
        csv
220 #COMBINE
221 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
        LIEO,LIER,LIET s_date=${datestr_day3} e_date=${datestr_day3} in_path=./..↩/
        data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
        weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
        _  in_suffix="_all" out_praefix=sard_idw_singleday_
222 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
        ,LIER,LIET s_date=${datestr_day3} e_date=${datestr_day3} in_praefix=sard_↩
        idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
        sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
        fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
223
224
225 #PYFWI-FORMAT
226 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day4} e_↩
        date=${datestr_day4} incsv_path=./../data/weather/wunderground_2011_↩
```

```
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
227 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day4} e_date=${datestr_day4} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
228 #PYFWI
229 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
230 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
    csv
231 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
232 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
233 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
234 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
    csv
235 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
236 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
237 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
238 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
    csv
239 #COMBINE
240 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day4} e_date=${datestr_day4} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
    _  in_suffix="_all" out_praefix=sard_idw_singleday_
241 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
```

```
    ,LIER,LIET s_date=${datestr_day4} e_date=${datestr_day4} in_praefix=sard_↩
    idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
    sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
    fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
242
243 #PYFWI-FORMAT
244 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day5} e_↩
    date=${datestr_day5} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
245 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day5} e_date=${datestr_day5} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
246 #PYFWI
247 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
248 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
    csv
249 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
250 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
251 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
252 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
    csv
253 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
254 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
255 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
256 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
```

```
    csv
257 #COMBINE
258 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day5} e_date=${datestr_day5} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
    _  in_suffix="_all" out_praefix=sard_idw_singleday_
259 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
    ,LIER,LIET s_date=${datestr_day5} e_date=${datestr_day5} in_praefix=sard_↩
    idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
    sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
    fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
260
261 #PYFWI-FORMAT
262 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day6} e_↩
    date=${datestr_day6} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
263 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day6} e_date=${datestr_day6} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
264 #PYFWI
265 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
266 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
    csv
267 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
268 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
269 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
270 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
```

```
    csv
271 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
272 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
273 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
274 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
    csv
275 #COMBINE
276 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day6} e_date=${datestr_day6} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
    _  in_suffix="_all" out_praefix=sard_idw_singleday_
277 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
    ,LIER,LIET s_date=${datestr_day6} e_date=${datestr_day6} in_praefix=sard_↩
    idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
    sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
    fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
278
279 #PYFWI-FORMAT
280 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day7} e_↩
    date=${datestr_day7} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
281 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day7} e_date=${datestr_day7} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
282 #PYFWI
283 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
284 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
```

```
    csv
285 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
286 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
287 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
288 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
    csv
289 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
290 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
291 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
292 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
    csv
293 #COMBINE
294 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day7} e_date=${datestr_day7} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
    _  in_suffix="_all" out_praefix=sard_idw_singleday_
295 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
    ,LIER,LIET s_date=${datestr_day7} e_date=${datestr_day7} in_praefix=sard_↩
    idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
    sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
    fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
296
297 #PYFWI-FORMAT
298 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day8} e_↩
    date=${datestr_day8} incsv_path=./../data/weather/wunderground_2011_↩
    forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
    /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
    wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
    outcsv_suffix=
299 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
```

```
    LIEO,LIER,LIET s_date=${datestr_day8} e_date=${datestr_day8} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
    praefix=sard_idw_ out_suffix="_all"
#PYFWI
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
    csv
FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
    csv
#COMBINE
python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
    LIEO,LIER,LIET s_date=${datestr_day8} e_date=${datestr_day8} in_path=./../↩
    data/weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/↩
    weather/wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw↩
    _  in_suffix="_all" out_praefix=sard_idw_singleday_
python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,LIEO↩
    ,LIER,LIET s_date=${datestr_day8} e_date=${datestr_day8} in_praefix=sard_↩
    idw_singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/↩
    sard/singledays/ out_path=./../data/weather/wunderground_2011_forecast/↩
    fwicomb/sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws

```

```
315 #PYFWI-FORMAT
316 python wunderground_splt.py incsv_praefix=sard_idw_ s_date=${datestr_day9} e_↩
        date=${datestr_day9} incsv_path=./../data/weather/wunderground_2011_↩
        forecast/idw/sard/ instart=sard_idw_corr_startdate.csv instart_path=../data↩
        /weather/wunderground_2011_forecast/startdate/ out_path=./../data/weather/↩
        wunderground_2011_forecast/fwi/sard/singledays/ out_praefix=sard_idw_ ↩
        outcsv_suffix=
317 python wunderground_splt_append.py stations=LIEA,LIEB,LIEC,LIED,LIEE,LIEF,LIEH,↩
        LIEO,LIER,LIET s_date=${datestr_day9} e_date=${datestr_day9} in_path=./../↩
        data/weather/wunderground_2011_forecast/fwi/sard/singledays/ out_path=./../↩
        data/weather/wunderground_2011_forecast/fwi/sard/ in_praefix=sard_idw_ out_↩
        praefix=sard_idw_ out_suffix="_all"
318 #PYFWI
319 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEA_all.↩
        csv
320 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEB_all.↩
        csv
321 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEC_all.↩
        csv
322 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIED_all.↩
        csv
323 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEE_all.↩
        csv
324 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEF_all.↩
        csv
325 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEH_all.↩
        csv
326 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIEO_all.↩
        csv
327 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIER_all.↩
        csv
328 FWI.py ./../data/weather/wunderground_2011_forecast/fwi/sard/sard_idw_LIET_all.↩
        csv
329 #COMBINE
330 python wunderground_fwicomb_cut.py stations=LIEA,LIEB,LIEC,LIEE,LIEF,LIEH,LIEO,↩
        LIER,LIET s_date=${datestr_day9} e_date=${datestr_day9} in_path=./../data/↩
        weather/wunderground_2011_forecast/fwi/sard/ out_path=./../data/weather/↩
```

```
    wunderground_2011_forecast/fwi/sard/singledays/ in_praefix=sard_idw_  in_↩
    suffix="_all" out_praefix=sard_idw_singleday_
331 python wunderground_fwicomb.py stations=LIEA,LIEB,LIEC,LIEE,LIEF,LIEH,LIEO,LIER↩
    ,LIET s_date=${datestr_day9} e_date=${datestr_day9} in_praefix=sard_idw_↩
    singleday_ in_path=./../data/weather/wunderground_2011_forecast/fwi/sard/↩
    singledays/ out_path=./../data/weather/wunderground_2011_forecast/fwicomb/↩
    sard/ out_praefix=sard_fwi_ out_suffix=_airports_pws
332
333
334
335 #FWI-IDW
336 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day0} e_date=${↩
    datestr_day0} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
337 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day1} e_date=${↩
    datestr_day1} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
338 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day2} e_date=${↩
    datestr_day2} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
339 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day3} e_date=${↩
    datestr_day3} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
```

```
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
340 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day4} e_date=${↩
    datestr_day4} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
341 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day5} e_date=${↩
    datestr_day5} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
342 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day6} e_date=${↩
    datestr_day6} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
343 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day7} e_date=${↩
    datestr_day7} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
344 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
    buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
    /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
    sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day8} e_date=${↩
    datestr_day8} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
    praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
    =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
```

```
345 python wunderground_fwiidw.py indem=dem_srtm41_sard_clipreg_32632_res3000_↩
        buf7500_nodata.asc indem_path=../data/elevation/srtm_90m/sard/ incsv_path=.↩
        /../data/weather/wunderground_2011_forecast/fwicomb/sard/ incsv_praefix=↩
        sard_fwi_ incsv_suffix=_airports_pws s_date=${datestr_day9} e_date=${↩
        datestr_day9} out_path=/var/www/gml/ out_path_merc=/var/www/gml/ out_↩
        praefix=sard_fwi_idw_ out_suffix=_airports_pws nodata=-999 projstring=+proj↩
        =utm+zone=35+ellps=WGS84+datum=WGS84+units=m+no_defs
346
347
348 python platform.py \
349   date=${datestr_day0} \
350   in_praefix="sard_fwi_idw_" \
351   out_suffix="_airports_pws" \
352   gmlin_path=./gml/ \
353   gmlmercin_path=./gml/ \
354   out_path=/var/www/ \
355   wmsgmlin_path=/var/www/gml/
356
357 cp /var/www/forecastday0_fwi.html /var/www/index.html
358 chmod -R 755 /var/www/gml/
359
360 rm -R ./../data/weather/wunderground_2011_forecast/fwi/sard/singledays/
361 mkdir ./../data/weather/wunderground_2011_forecast/fwi/sard/singledays
```