UNIVERSITÉ DE LAUSANNE

Summary

Faculty of Geosciences and Environment (FGSE)
Institute of Earth Surface Dynamics (IDYST)

Doctor of Science

Software and Numerical Tools for Paleoclimate Analysis

by Philipp S. SOMMER

Data-model comparisons of Holocene (11,700 years ago to present) climate provide an ideal basis for evaluating climate model performance outside the range of modern climate variability. The Holocene is recent enough so that boundary conditions of the underlying physics and forcing are well known, while paleoenvironmental archives are abundant and dated with enough precision to comprehensively reconstruct climate. To date, efforts to reconstruct the spatial patterns of Holocene climate change have been mainly focused on the mid-Holocene (about 6~~,00~~ '000 years ago), but significant discrepancies have already been identified in data-model comparisons.

These data-model discrepancies can be investigated using instrumental datasets covering continental or hemispheric scales which allow us to reconstruct large-scale climatic features, such as atmospheric dynamics or latitudinal temperature gradients. The generation of these datasets for times prior to the 19th century however faces considerable challenge because there are very few direct measurements of climatic variables. We rely on climate proxies as indirect measurements of the paleo climate. The most abundant one is fossil pollen data, i.e. pollen that are produced by vegetation and can be preserved over thousands of years in terrestrial (or coastal) archives (e.g. lake sediments). This proxy is available from all non-glaciated continents over the world in many different climate regimes, and the primary data is becoming increasingly accessible through large publicly available and community-driven relational databases. Our ability to use this proxy for continental-scale climate reconstructions, however, depends on our ability to analyze, explore and find patterns in these rich and heterogeneous databases. In particular, this requires a proper understanding of the uncertainties that are related to the indirect measurement of climate.

In the first part of this thesis, I present three new software tools that tackle the challenge to make this large amount of data accessible, and to build and develop a continental-scale pollen database. These tools cover a wide range of possible applications to leverage our work with site-based proxy data to a continental scale. The first tool I present is a web framework that is built around a map-based interactive database viewer, developed primarily for the Eurasian Modern Pollen Database, EMPD. This new tool makes the database accessible to other researchers and to the general public, and it allows a continuous and stable development of the community-driven database. In addition to the EMPD, I present an extension of this viewer that makes a large northern-hemispheric fossil pollen database accessible and allows its visual exploration.

The second tool tackles the challenge to fill the gaps in certain geographic areas in the pollen database. *straditize* is a digitization software for stratigraphic diagrams,

Chapter 1

Introduction

1.1 Motivation

Our understanding of the climate system is based on computational models that operate on large spatial scales and simulate a complex system of closely related environmental parameters. The evaluation of these models poses a considerable challenge because we need data on continental scale to reconstruct large-scale climatic features, such as atmospheric dynamics or latitudinal temperature gradients. This instrumental data, however, is limited to not even the past two centuries and overlaps highly with the *comfort zone* of the models, i.e. the period where they have both been developed and tested.

A comparison of models with paleo-environmental proxy records, i.e. records from past climates prior to the systematic measurement of meteorology and climatology, provides therefore the only possibility to evaluate the predictive skill of our models for climates that are very different from today. The Holocene, ranging from 11'700 years ago to present, provides an ideal basis for it because (1) it is recent enough so that boundary conditions and forcings are well known, and (2) paleoenvironmental archives are abundant and dated with enough precision to comprehensively reconstruct climate. Each of these archives represent the regional climate condition in the surrounding environment and, when grouped together into large databases; they allow an informed estimate of the climate state over a large period of time and vast geographic areas.

A direct comparison of climate model output and proxies is however still challenging because even the climate proxy record, as an indirect measurement of climate, relies on an inverse modelling approach with associated uncertainties that are not always easy to quantify.

Key challenges for large-scale data-model comparisons on past climates are therefore (1) to gather enough climate proxy information from a spatially large area and a variety of climates, and (2) to provide reliable estimates of the uncertainties associated to the indirect measurements.

These challenges will be addressed in this thesis via the development of new software tools that cover flexible data analysis (chapters 2 and 4), a tool for data gathering (chapter 3), as well as new predictive methods for large-scale paleoenvironmental modelling (chapters 5 and 6).

All these tools are open-source with a strong emphasis on a proper software development that includes documentation and reproducibility.

In the following section 1.2, I will describe the paleo-climate of the current epoch and why this is of interest for future climate predictions. In the subsequent section 1.3, I introduce the influence of software development in this paleo-climatic research, and some of the common open-source software development contents. I conclude this chapter by providing an overview on the contents of this thesis in section 1.3.2.

1.2 Learning from the past — Why we study paleo-climates

Mankind is facing large infrastructural challenges during this century, such as the loss of biodiversity, an exponentially growing world population and an acceleration of growth and globalization of markets (e.g. Ceballos et al., 2015; United Nations, 2019; World Bank, 2002). They all interact with a global climate change that may lead to a new environment none of us ever experienced (Collins et al., 2013). Any future global planning has to account highly diverse responses that range from regional to continental scales (Christensen et al., 2013). As such the complex climate system will enter a state that is significantly different from everything we had since the beginning of the satellite era, i.e. the beginning of global meteorological data acquisition, and even different from what has been experienced within the last 2'000 years (Neukom et al., 2019a,b).

Our knowledge about this new climate is therefore mainly based on computational Earth System Models (ESMs). They face the challenge of simulating a new climate based on our present knowledge of the interactions between the different compartments Ocean, Land and Atmosphere. The validation of it becomes conceptually difficult because of the aforementioned transition into a warmer world during the next century. We are entering a new state and it is questionable how well our models perform (Hargreaves et al., 2013; Mauri et al., 2014).

To evaluate the predictive skill, we rely on our knowledge of paleo-climates, i.e. climates before the systematic measurement of temperature, precipitation, etc.. They provide the only opportunity for a large-scale evaluation of ESMs under conditions very different than today. Paleo-climatic research has therefore been an integral part for climate sciences since the 80s (COHMAP Members, 1988; Joussaume and Taylor, 1995), particularly in the Paleoclimate Modelling Intercomparison Project (PMIP) (Braconnot et al., 2012, 2007a,b; Jungclaus et al., 2017; Kageyama et al., 2016; Otto-Bliesner et al., 2017).

The Holocene interglacial period (11'700 years ago to present) (Walker et al., 2009) is particularly important because it is sufficiently close in time to provide paleo-climate archives and the forcings and boundary conditions are well known (Wanner et al., 2008). With the end of the Younger Dryas around 11'700 years ago, the Earth experienced a climate warming due to changes in orbital precession and obliquity of the Earth, as well as the disappearing residual ice sheets of the Last Glacial Maximum (LGM) (Berger and Loutre, 1991; Peltier, 2004). This results in a multitude of large-scale effects in the atmospheric circulation, such as an increasing amplitude and frequency of the El Niño–Southern Oscillation (ENSO) (Donders et al., 2008), stronger westerly circulation in winter indicating a more positive AO/-NAO over mid-latitudes and the arctic (Funder et al., 2011; Mauri et al., 2014) and changes in the polar amplification and a weakening of the latitudinal temperature gradient (Davis and Brewer, 2009).

Hence, this epoch is of particular interest because the continental setup is comparable to nowadays while still having a climate that is significantly different from present day.

1.2.1 Pollen as a climate proxy

Before 1850, there is almost no instrumental measurement of temperature. Instead we rely on archives such as lake sediments, glaciers, peat bogs, or speleothems that

preserve climate proxies. The latter is a set of variables that are influenced by climate conditions and therefore allow an indirect measurement of climate(-related) parameters at ancient times, e.g. temperature, precipitation or sea-level.

The most abundant climate proxy, that I will also focus on in the next chapters, are pollen assemblages. It is the geographically most spread paleo-climate proxy (Birks and Birks, 1980) and has a long history in quantitative paleo-climatologic reconstructions (e.g. Bradley, 1985; Iversen, 1944; Nichols, 1967, 1969).

The chemically stable polymer sporopollenin allows the pollen grain to be preserved over very long periods of time, in various terrestrial archives such as lakes, wetlands or ocean sediments (Fægri et al., 1989; Havinga, 1967). Pollen are produced by seed-bearing plants (spermatophytes, Wodehouse, 1935) and as such have a high spatial continuity and prevalence (Chevalier et al., in prep). Their compositions are strongly influenced by the surrounding climate, although other factors, such as soil compositions or inter-species competition also play an important role. This dependency allows to reconstruct the driving factor, i.e. climate parameters such as winter and summer temperature, or precipitation from the observed pollen data (Brewer et al., 2007; Chevalier et al., in prep; Juggins, 2013; Juggins and Birks, 2012).

This high abundance of pollen led to multiple regional efforts to combine and homogenize fossil pollen data. This makes pollen particularly useful for large-scale data-model intercomparisons. The earliest examples are the ~~European Pollen Database (EPD)~~ European Pollen Data-base (EPD) and North American Pollen Database (NAPD) that both started around 1990 and developed a similar structure in order to be compatible (Fyfe et al., 2009; Grimm, 2008) . This let to the development of other regional pollen databases, such as the Latin American Pollen Database (LAPD) (Flantua et al., 2015; Marchant et al., 2002) in 1994 or the African Pollen Database (APD) (Vincens et al., 2007) in 1996, and others (see Grimm, 2008). These attempts finally let to the development of the Neotoma database (Williams et al., 2018), a global multiproxy database that incorporates many of the regional pollen databases.

The use of pollen for paleo-climate reconstruction has a long academic tradition in geology (Bradley, 1985) and provides the source of large-scale paleo-climatic reconstructions in number of different studies (Davis et al., 2003; Fischer and Jungclaus, 2011; Marsicek et al., 2018; Mauri et al., 2015, and more). ~~They howeverhave multiple uncertainties ,~~ Such a reconstruction, however, has multiple uncertainties that are often difficult to quantify and to consider (see chapter 5). ~~A key challenge~~ Key challenges for a data-model comparison are dating uncertainties, ~~or the influence~~ influences of seasonality on the proxy (e.g. whether it represents summer, winter or annual temperature) and ~~the quality~~ quality and temporal resolutions of the record. Another challenge is the proper handling of uncertainties ~~of~~ related to the inverse modelling approach (e.g. Guiot and Vernal, 2011; Telford and Birks, 2005, 2009), and the spatial coverage of the proxy (see chapter 3).

1.3 Software for Paleoclimatology

The usage of software is crucial for the quantitative reconstruction of Earth's Climate. Paleoclimate research is facing an information overload problem and requires innovative ~~methodologies~~ methods in the realm of visual analytics, i.e. the interplay between automated analysis techniques and interactive visualization (Keim et al., 2008; Nocke, 2014). As such, a visual representation of the paleoclimate reconstruction has been essential for both, proxies (Bradley, 1985; Grimm, 1988; Nichols, 1967) and models (Böttinger and Röber, 2019; Nocke, 2014; Nocke et al., 2008; Phillips,

1956; Rautenhaus et al., 2018), although the visualization methods significantly differ due to the differences in data size and data heterogeneity.

The second important aspect for software and paleoclimate is the distribution of data to make it accessible to other researchers, the community and policy makers, which is commonly established through online accessible data archives and recently also through map-based web interfaces (Bolliet et al., 2016; Williams et al., 2018).

The following sections provide an overview on the different techniques used by palynologists to visualize and distribute their data and concludes with an introduction into Open-Source Software Development, which forms the basis of all the software solutions that are presented later in this thesis.

1.3.1 Sofware for Proxy Data Analysis, Visualization and Distribution

Due to the nature of stratigraphic data, proxies, especially pollen assemblages, are often treated as a collection of multiple time-series (one-dimensional arrays). The size of one dataset is generally small (in the range of kB) and can be treated as plain text files. Traditionally, numerical and statistical analysis are separated from the visualization.

In palynology, standard analytical tools are Microsoft Excel[1] and the R software for statistical computing (R Core Team, 2019). The latter also involves multiple packages for paleoclimatic reconstruction, such as `rioja` (Juggins, 2017) and `analogue` (Simpson, 2007; Simpson and Oksanen, 2019), and bayesian methods exists in a variety of programming languages ~~(**Tipton2017**; e.g. Haslett et al., 2006; Holmström et al., 2015; Nolan et a~~ (e.g. Haslett et al., 2006; Holmström et al., 2015; Nolan et al., 2019; Parnell et al., 2014; Tipton, 2019) . Alternatively, desktop applications exist, such as Polygon[2] by Nakagawa et al., 2002 or the CREST software presented by Chevalier et al., 2014 and Chevalier, 2019.

It is a long-standing tradition to visualize stratigraphic data, and especially pollen data, in form of a stratigraphic (pollen) diagram (Bradley, 1985; Grimm, 1988). Especially during the 20th century, when it was not yet common to distribute data alongside a peer-reviewed publication, pollen diagrams have been the only possibility to publish the entire dataset (see also chapter 3). The generation of these diagrams is usually based on desktop applications such as C2 (Juggins, 2007) or Tilia[3] (Grimm, 1988, 1991). A more recent implementation into the psyplot framework (Sommer, 2017, chapter 4) is also provided with the psy-strat plugin[4] (Sommer, 2019).

Raw pollen data is at present made available through web archives, such as PANGAEA[5] or the National Climatic Data Center (NCDC) by the National Oceanic and Atmospheric Administration (NOAA)[6]. Collections of data, such as regional pollen databases or project specific collections (e.g. Davis et al., 2013; Whitmore et al., 2005) are usually published in one of the above-mentioned archives or associated with a publication. A different approach has been taken by Bolliet et al., 2016 who developed a small web application as an interface into the data collection, the *ClimateProxiesFinder* (Brockmann, 2016, chapter 2).

Outstanding compared to the previous data interfaces is the new infrastructure for the Neotoma database (Williams et al., 2018). It consists of the map-based web

[1] `https://products.office.com/en/excel`
[2] `http://polsystems.rits-paleo.com`
[3] `https://www.tiliait.com/`
[4] `https://psy-strat.readthedocs.io`
[5] `https://pangaea.de/`
[6] `https://www.ncdc.noaa.gov/data-access/paleoclimatology-data`

a CI service, such as Travis CI[12], Appveyor[13] or CircleCi[14] that are integrated into the Github repository (section 1.3.2). Every commit to the Github repository, or any new pull requests then triggers the tests. This transparently allows to ensure the operability of the software, and the test coverage report ensures that the newly implemented functionality is properly tested. A software development concept that is entirely built on this idea is the test-driven development. Within this framework, new features are implemented by starting with the test that should be fulfilled by the new feature and then improving the software until this test pass (Beck, 2002).

Automated Documentation

Documentation is the key aspect of a sustainable software and much of the geoscientific code has a lack of proper documentation (based on personal experience). For the software in this thesis, four different levels of the documentation play an important role:

The Application programming interface (API) documentation is meant to document the major parts of the software code that is subject to be used by external scripts or packages. It is usually implemented in the code and documents the essential subroutines and methods of the software.

The graphical user interface (GUI) documentation provides help for the most high-level functionality for the software. The GUI is a user interface into the software through graphical elements (such as buttons, checkboxes, etc.). Unlike the API documentation, it should not require knowledge about programming.

The contributing and/or developers guide is targeting other software developers that might want to contribute to the software package. This document states how other software developers should contribute to the software and introduces the central structural aspects and frameworks of the software.

The manual (or also commonly referred to as *the* documentation) is the document that contains all necessary information for the software, such as installation instructions, tutorials, examples, etc.. It often includes some (or multiple) of the above parts.

The documentations for the software in this thesis have been automatically generated with Sphinx, a Python tool to generate documentations in various different formats (such as HTML, PDF, etc.) (Hasecke, 2019; Perez et al., 2011). It is also implemented as a webhook into the Github repository (see section 1.3.2) to automatically generate an up-to-date documentation of the software for each commit to the Github repository. This provides an additional automated test for the software, and especially ~~it's~~ its high-level-interface, in addition to the automated test suite described in the previous section. Most of the manuals for the software packages in this thesis are hosted and build online with the free services offered by readthedocs.org.

[12] `https://travis-ci.org/`
[13] `https://appveyor.com`
[14] `https://circleci.com/`

Distribution through package managers and virtual environments

FOSS software is meant to be extensible and to build upon other FOSS packages. This requires an accurate and transparent handling of ~~it's~~ its dependencies and requirements which is usually provided through the so-called packaging of the software (e.g. Torborg, 2016). There exists a variety of package managers and the choice most often depends on the framework of the software.

The software in this thesis is mainly distributed via two systems. The first one is python's own package manager *pip* which is based on the packages uploaded to pypi.org. The second one, which got increasing importance during the recent past, is the open-source Anaconda Distribution[15]. Both work on multiple operating systems (Windows, Linux and Mac OS), but the Anaconda Distribution contains also non-python packages (e.g. written in C or C++) that multiple Python packages rely on; and it contains a rich suite of R packages.

One step further, compared to package managers, are the distribution of virtual environments. These systems do not only provide the software, but also a full operating system and the installed dependencies. A popular platform (used also for the Eurasian Modern Pollen Database (EMPD) database) is provided through so-called Docker containers[16]. Compared to package managers, this system has the advantage of simplifying the installation procedure for the user because he only has to download the corresponding docker image. The docker image itself then runs independent of the local file system in a separate isolated mode.

1.4 Challenges tackled in this thesis

In part I of this thesis I present several new software tools that tackle the data analysis, data gathering and data distribution aspects described in the previous section 1.3.

Chapter 2 in this first part describes new tools for the data analysis and distribution of pollen data on a large continental scale. In this chapter I present the new infrastructural tools I developed for the sustainable management of the community-driven Eurasian Modern Pollen Database (EMPD). These tools consist of a flexible and lightweight map-based web interface into the data, the EMPD-viewer, and a webserver for an automated administration of the database. Within this chapter, I also present another use case for the map-viewer that is adapted to a large northern-hemispheric database of fossil and modern pollen records.

The second chapter in this part, chapter 3, describes the new *straditize* software that addresses the problem of gathering proxy data that has been collected during the pre-digital area. This software is a semi-automated digitization package for stratigraphic diagrams, and particularly pollen diagrams that we use to fill gaps in our database in data-poor regions.

I conclude the first part with the presentation of the generic visualization framework psyplot in chapter 4. It is a suite of python packages that are designed for an interactive visual analysis of data, both from a GUI and the command line. This software is the base infrastructure for many of the tools described in the other chapters. It has a very general scope is not limited to paleoclimate analysis.

In the second part I present two new models that leverage site-based observations (or paleo climate reconstruction) onto a continental, or even global scale. The first model in chapter 5 presents the very recent *pyleogrid* package that extends the

[15] `https://www.anaconda.com`

[16] `https://www.docker.com`

the meta data table, together with all the other samples. Another key element of the viewer are the meta data filters, that subset the data using efficient and intuitive filtering tools. This allows to search the database, or to select specific countries, climatic regimes, sample types, samples of a specific data contributor/analyst, and more.

Additional information on the sample is revealed through a bar diagram of the associated pollen data, which is dynamically created when the user clicks on the sample. The viewer also displays monthly, seasonal and annual precipitation and temperature values at the side, based on the WorldClim dataset, version 2 (Fick and Hijmans, 2017).

Finally, the viewer contains elements that allow scientists to contribute to the database, even without dedicated knowledge about the Github framework. The meta data editor allows to edit a sample and then submit it via the data submission form. The request is handled by the EMPD-admin webapp (see section 2.2.3) that pushes the data to the corresponding pull request on Github that is then reviewed by the core database maintainers. Another implemented element is an issue report form that allows the user to highlight erroneous sample information which is then, again through the EMPD-admin, submitted as a Github issue to the data repository.

The web app is fully integrated into the Github framework of the EMPD and loads the displayed data from the online repository. As such, it also provides a further quality control check and allows the data contributors/maintainers to review and edit new contributions before they are merged into the database.

Implementation details

The viewer itself is very light-weight and can be flexibly adapted to other database systems (see for example section 2.3). As the climate proxies finder (Bolliet et al., 2016; Brockmann, 2016), the EMPD-viewers main viewing/filtering functionality it is built upon the *dc* (Zhu and the dc.js Developers, 2019), *crossfilter* (Square, Inc. and crossfilter contributors, 2019) and *leaflet* (Agafonkin and leaflet contributors, 2019) open source JavaScript libraries. We ported the app to the *npm* package manager (npmjs.com) which enables a better and more secure monitoring of the app dependencies. This package manager is also used for an automated testing of the viewer on a CI service, prior to deployment on the official web page. Due to time constrains, the viewer is not yet fully adapted to mobile devices.

2.2.2 The EMPD2 data repository

The raw data of the EMPD2 is accessible as plain text files in the *EMPD-data* Github repository (see table 2.1). The software development framework of Github (see introductory part of section 2.2) is adopted such that issues in the data repository can highlight errors in the database, or provide room for the discussion of potential new efforts that should be considered within the community-database. Pull requests into the repository are new data contributions that can be reviewed by the maintainers before being merged into the official database.

This ~~methodology~~ method allows a fully transparent traceback of changes made to the EMPD through version control. The online access to the raw data files through Github also allows the EMPD viewer to interface with different versions of the database (see previous section).

The EMPD-data repository additionally uses the CI services from Travis CI (travis-ci.org) for automated tests of the meta data in each sample.

2.2.3 The EMPD-admin

In addition to the standard CI services, we developed the EMPD-admin webapp. Inspired by the web management tools of the conda-forge community()[1], this tool provides an automated handling of data contributions from within Github Pull Requests. It behaves like a standard CI service and runs tests on the data contribution, every time changes have been made to the pull request.

But the main purpose of the EMPD-admin is to provide a web tool for an automated administration of the database, which is helpful for a community-project with changing maintainers. Hence, the EMPD-admin web app acts like a bot that reacts on comments from within a pull request (i.e. a data contribution). Maintainers and contributors can use this functionality and directly contact the bot, for instance, to subset the data, run specific tests on subsets of the data, or automatically fix certain meta data issues, such as wrong countries or missing elevation.

The bot is also integrated in the EMPD-viewer (see previous section 2.2.1). Bug reports or edited data are processed by the EMPD-admin and put online as an issue in the github repository, or it updates the corresponding data contribution.

As such, the administration of the database can be done entirely remotely, without having to install dedicated software on a local computer.

Implementation details

The EMPD-admin webapp is hosted for free at Heroku (https://www.heroku.com) at empd-admin.herokuapp.com with a software package documentation hosted at EMPD2.github.io/EMPD-admin. This, again, allows stability independent on the availability of funding. The package can, also be installed locally and used from the command-line, independent of Github and Heroku, which is sometimes helpful for very large data contributions..

The Python library is based on the tornado web framework[2], as well as pandas (McKinney, 2010), a tabular data analysis library for Python, and sqlalchemy (Bayer, 2012), a Python SQL toolkit.

2.2.4 Distribution of the tools

The EMPD is hosted within the EMPD2 Github organization (github.com/EMPD2) at in the EMPD-data repository. The source files of the viewer are accessible at in the EMPD-viewer, and for the EMPD-admin in the EMPD-admin repository (see also table 2.1).

The EMPD-data and the EMPD-admin are additionally both available as so-called Docker container image at `https://hub.docker.com/u/empd2`. These containers are lightweight, standalone, executable packages of software that include everything needed to run an application: code, runtime, system tools, system libraries and settings. As such, they extend standard software packaging systems by providing an entire operating system that contains the target application. This makes it especially useful for web applications (such as the EMPD-admin) that can, as such, operate in a well-defined and portable environment.

The EMPD-admin can, however, also be installed through the standard python package manager `pip`.

[1] conda-forge.org
[2] www.tornadoweb.org

Chapter 3

Straditize

A digitization software for pollen diagrams

Straditize *is published in the Journal of Open Source Software:*

Sommer, Philipp S., Dilan Rech, Manuel Chevalier, and Basil A. S. Davis (Feb. 2019). "straditize: Digitizing stratigraphic diagrams". In: *Journal of Open Source Software* 4.34, p. 1216. DOI: 10.21105/joss.01216. URL: https://doi.org/10.21105/joss.01216.

Abstract. The conversion of printed diagrams or figures into numerical data has become extremely important in ensuring that scientific work, especially from the pre or early digital age, is not lost to science. One of the most common figures used in the paleo-sciences is the stratigraphic diagram, where the results of the analysis of samples are plotted against a common y-axis, usually representing age or depth. Currently this type of diagram is laborious and error prone to digitize using current software designed for simple x/y graphs. Here we present a new open source software written in python that is specifically designed to quickly and accurately digitize stratigraphic diagrams based on a user controlled semi-automatic process. The software is optimized for use with pollen diagrams, but will work well with many other types of similar diagram. The software is fully documented and includes integrated help and tutorials.

3.1 Introduction

As with almost all areas of science, the digital capture, storage, manipulation and sharing of data has almost completely transformed the way that paleo-science is now undertaken compared to just 20-30 years ago. This digital transformation has created entirely new types of datasets, analysis, collaborations and visualizations, but it has also created a profound divide between the science that is available in digitized form, and that which is only available in analogue or paper form. This is a particular problem for science that was undertaken and published before this digital revolution, or where the original digital version of a dataset is unavailable, perhaps through retirement or other personnel changes, accidental damage to data storage devices, incompatible or out of date hardware storage or data file formats.

This data however may still be available as a published or printed diagram, which can be turned into numerical data by digitization, either manually or often using graph digitizing software such as *Graphclick*, *Engage Digitizer*, *Plot Digitizer*, *g3data*, *Digitizelt* and *WebPlotDigitizer* (Rohatgi, 2019). While this approach may be optimal for simple graphs with a single x and y axis, it can rapidly become extremely

time-consuming and error prone for stratigraphic diagrams with a shared y-axis and multiple x-axis. In a stratigraphic diagram the y-axis commonly takes the form of a depth or age scale (or both) reflecting sampling down a sediment core or open sediment section, which is then accompanied by a series of x-axis that plot the results of the analysis on each of these samples. Each sample may have been analyzed for a variety biotic or abiotic paleo-environmental indicators, and plotted on a variety of x-axis scales.

Here we present an open-source software *straditize* (Sommer et al., 2019) that has been specifically developed for digitizing stratigraphic diagrams. The software assumes that the figure follows the standard format associated with stratigraphic figures with a depth or age scale on the y-axis, and then a series of columns that represent the results of the analysis on common samples at specific depths or ages. The design is optimized for pollen diagrams (figure 3.1), but can be used without modification with any similar diagram design irrespective of the type of data being presented. The software first interprets the structure of the stratigraphic diagram and then reconstructs the data associated with each sample. This is done using a semi-automated process whereby many aspects of the software are automated but still editable by the user. The software allows the user to make continual visual checks on the digitization process, and provides the functionality to export the entire project in a data format that is independent of the platform and software[1]. The software is open-source and written in the programming language python (Perez et al., 2011) which makes it very flexible and easy to adapt. It is also equipped with an extensively documented graphical user interface, interactive visualizations and tutorials that allow the user to discover and to use the semi-automated methodologies. Additionally, *straditize* comes with an extensive test suite for a sustainable workflow with automated checks that also ensure the basic functionality of the various features in the software.

3.2 Methods: Treatment of stratigraphic diagram features

In this section we describe the common features of a stratigraphic pollen diagram and their handling within *straditize*. The emphasis is on pollen diagrams A pollen diagram highlighting the features in a stratigraphic diagram is provided in figure 3.1.

3.2.1 Structure of a stratigraphic diagram

3.2.1.1 Stratigraphic columns

A stratigraphic diagram consists of multiple sub diagrams, each visualizing one or more different variables, for example the percentages of different pollen taxa, the concentration of different chemical elements, or the various percentages of different grain sizes. These sub diagrams share one common axis which is usually the age or depth of the core (see fig. 3.1 a). The diagram is then divided into multiple sub diagrams which we refer to as the columns of the stratigraphic diagram (fig. 3.1 b). Each column visualizes the data of one variable, such as a pollen taxon, or multiple variables where these are plotted within the same column (same x-axis), such as winter and summer precipitation shown in the left most column of figure 3.1 . The current version of *straditize* requires that the columns do not overlap and that

[1] *straditize* projects are exported as NetCDF file (Rew et al., 1989) that allows a platform and programming language independent access and sharing of the data

multiple variables plotted in the same column do not overlap either. The software can process multiple variables in a column so long as these are stacked or additive, and therefore they never overlap. An example in a pollen diagram could be where multiple species of, for instance Betula, are plotted as a stacked diagram in a single column so that the sum of the species also shows the total ~~Betula~~*Betula*.

At the top of each column it is usual to find the name of the variable plotted in the column. This may be shown at a variety of angles, but usually either vertically, or at an angle or rotated to make it easier to read and fit within the diagram (fig. 3.1 c). This label can be automatically read by *straditize* and the name assigned to the respective column data, although care should be taken as the label is sometimes offset from the column it represents.

Variables are also often grouped together and the group labelled, for instance in pollen diagrams into *trees & shrubs*, and *herbs* (fig. 3.1 d). For pollen data these groups usually share the same x-axis units/scaling (as in fig. 3.1 b). In *straditize* the units/scaling can be set and applied to a whole group of columns/variables, or set and assigned for each individual column/variable (see fig. 3.1 e).

3.2.1.2 Diagram types

A number of different diagram types are shown in figure 3.1 that are commonly used in pollen diagrams, as well as other stratigraphic diagrams. These can all be identified and read by *straditize*. One of the most commonly found diagrams in pollen diagrams are line diagrams (fig. 3.1 f), or line diagrams where the area underneath of the line is filled to make an area diagram (fig. 3.1 h). Data is also often commonly presented as bar diagrams that make it clearer where the individual samples are located (fig. 3.1 g). Both bar and line/area diagrams may also be stacked, where (as we have already mentioned) columns may contain multiple variables stacked one upon the other (fig. 3.1 i). These various diagram types require different digitization strategies, which we discuss in the digitization section below (see section 3.2.2).

3.2.1.3 Informative features

Other more specialized features can also be found in pollen diagrams that provide additional information for the reader, but are more difficult to interpret for the software. For instance the taxa or variables in a pollen diagram are usually all plotted on the same scale even if they are in different columns, so that direct visual comparison can be made between them. However, whilst this works well for large percentage values, it can often be difficult to see changes in low percentage values, which may still be ecologically important. To help the reader see these changes in low percentages, pollen diagrams often include a vertical exaggeration. This means that the percentages for a pollen taxa in a column will be plotted with two lines, one showing the percentage value shown on the scale, and the other showing the percentage value multiplied or exaggerated by a certain factor (usually 5 or 10) (fig. 3.1 j). A different approach to the same problem is to mark the low percentages with a symbol or marker. For instance, a common method is to mark all samples with less than 0.5% or 1.0% with a "+" symbol (fig. 3.1 k).

Other features that are often added to pollen diagrams and other stratigraphic diagrams are vertical and horizontal lines. Vertical lines often denote the start of a column, representing the baseline of the y-axes. These are often a continuous or discontinuous dashed line, and when it is quite a thick line it can be difficult to define

FIGURE 3.2: Cleaned binary image of the data part of figure 3.1. Informative features (Y-axes and horizontal lines) have been removed. Exaggerations and occurrences are still in the binary image and are considered separately in section ??3.2.2.4.

Straditize therefore has an automatic algorithm to detect vertical axes that tries to minimize the removing of real data pixels. This detection is described by the following algorithm:

Let $C(i)$ be the most frequent color in the pixels of a pixel column i, and $D(i)$ the amount of data pixels in this column. A pixel column that is covered by data with more than a user-defined threshold (by default 30 percent of the data part height, see section 3.2.2.1) is considered as part of a y-axis if

- it is either the first pixel column in the subdiagram with data (i.e. $D(j) = 0$ for $j < i$ and j being larger or equal than the column start of the diagram),
- or the dominant color of the pixel column is the same as for the previous pixel column (i.e. $C(i) = C(i-1)$) and the number of data points is approximately the same (i.e. $D(i) \approx D(i-1)$).

This procedure results in one vertical line per column and works independent of whether it is a dotted, dashed or solid line. However, the line width is critical and may vary a lot. If a diagram column contains a filled line (fig. 3.1 h) that merges with the vertical line defining the y-axis, then the algorithm could potentially overestimate the width of the y-axis line. Therefore, we use the median of all of the estimated lines from the various columns as the width of the y-axis line, and reduce the width of each of the lines to this amount. The algorithm then looks for informative features or other features that appear to be part of the data columns but are not part of the data behind the image. These include small features such as axis tick marks for example, and lighter pixels (i.e. close to white) that are usually a result from the rasterization of the diagram image. *straditize* then moves the start of the column because it assumes that the starting point for the x-axis (for pollen taxa it would be the 0% line) is in the middle of the vertical line marking the y-axis.

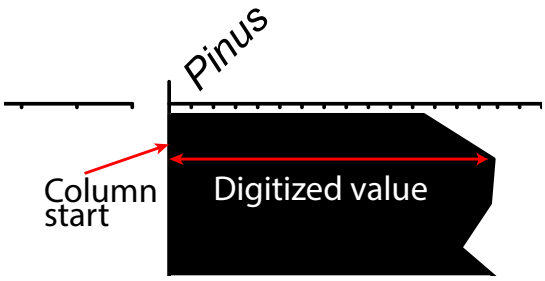

FIGURE 3.3: Illustration of the basic digitization strategy of *straditize*. For each diagram column in the binary image (fig. 3.2), we use the pixel that is located furthest to the right on the curve and take the distance to the column start as the digitized value. This is then transformed from the pixel scale into the data units based on the user input.

(4) Handling low taxon values

One particular feature often associated with pollen diagrams is the use of vertical exaggeration to help visualize changes in low percentages. Ordinarily, low percentages could be viewed better by changing the x-axis scale for the taxa with low percentages, but with pollen data it is also important to be able to make a visual comparison across all of the taxa listed, including those with high pollen percentages. Therefore, a common scale for all taxa is important. The exaggeration (fig. 3.1 j) usually takes the form of a second line outside of the first, usually representing x5 or x10 exaggeration of the scale presented on the x-axis. This line could be expected to have greater precision than the first for any given pixel resolution, but problems emerge when the value of the exaggerated value exceeds the width of the scale on the x-axis, so that the line marking the exaggerated values is truncated at high values above a certain threshold. Another common way to help the reader identify low percentages in pollen diagrams is to use a marker (often a "+" symbol, fig. 3.1 k) for values below a certain threshold. This can be particularly useful for very low counts where the author wants the reader to be aware that pollen of a certain taxa was found, even if the pollen counted was very low. This is often used for instance with "important" taxa such as ~~Cereals~~ *Cereals* that can indicate human agricultural activity, and taxa like ~~Larix~~ *Larix* (Larch) that have notoriously low pollen productivity and where <1.0% in a pollen diagram may actually represent 20% of ~~Larix~~ *Larix* trees in the surrounding landscape. For the purpose of digitization, the *straditize* user can either remove these exaggerations, or use some of the functions available in *straditize* to consider both the non-exaggerated and the exaggerated information in the diagram. In the case of the use of symbols to represent values below a threshold, it needs to be decided what value the symbol will represent once it is digitized and turned into numerical data. In any case, exaggerations and occurences are automatically removed from the image before the diagram is digitized (see next step 3.2.2.5).

(5) Digitizing the diagram

After removing the informative features (see section 3.2.2.3) and exaggerations (see section 3.2.2.4), *straditize* can automatically digitize the various columns on a pixel basis. In general *straditize* treats every column of the diagram separately and uses different algorithms for the various plotting types:

Area and line diagrams such as those shown in the fig. 3.1 f and fig. 3.1 h are digitized based on the pixel located furthest to the right on the curve in any particular column (illustrated in figure 3.3).

Bar diagrams as in figure 3.1 g are also digitized based on the the pixel located furthest to the right on the bar in any particular column. Additionally, *straditize* distinguishes between two adjacent bars by using a user defined threshold (by default two pixels). Additionally, it identifies bars that are significantly wider than the others (which would indicate two or more overlapping bars) and then the user can split them manually.

Stacked diagrams as in figure 3.1 i have to be digitized manually. The user has to manually distinguish the different areas using the selection tools in *straditize*.

These procedures each result in one value per pixel row in each variable column in the data part. The next step is then to extract only the rows that are necessary to regenerate the diagram, that is, the pixel rows that correspond to the samples.

(6) Finding the samples

A key function of *straditize* is its ability to identify sample levels in the data, so that measurements of x-axis values in each column for each variable are assigned to the appropriate y-axis sample depth or age across all columns and variables. The search and assignment of sample levels can be done either automatically or manually, and if done automatically, this can also be later edited following manual checking.

The algorithm is thereby split into two steps:

1. For each column: Identify the intervals that contain exactly one sample (the rough locations, i.e. certain consecutive pixel rows in every column where we know that there is a sample, but we do not know where exactly)

2. Align the overlapping intervals between the columns to estimate the exact location

The implementation of step 1 necessarily differs between bar and area/stacked ~~/~~or line diagrams. With bar diagrams *straditize* uses the bars identified in the previous digitization step (section 3.2.2.5) to define these rough sample locations, while for the other diagram types the algorithm looks for local extrema in the graph line, i.e. intervals that are lower or higher than the surrounding areas. This implies that each sample is associated with a local minimum or maximum in at least one of the diagram columns. This holds well for pollen diagrams that usually sum up to 100% across all of the columns or variables in the diagram, but it is however not generally true for all stratigraphic diagrams.

Step 2 then aligns these rough intervals and uses the overlapping information from the different columns to estimate the exact location of the sample. This is described by the following procedure, where we focus on a simple case of only two diagram columns. Assume that the two columns i and j have a sample in the corresponding overlapping intervals I_i and I_j (i.e. $I_i = \{r_{i,1}, r_{i,2}, \ldots\}$, and $I_j = \{r_{j,1}, r_{j,2}, \ldots\}$ with $I_i \cap I_j \neq \emptyset$). To find the exact location, *straditize* distinguishes the following cases:

If one of the intervals contains only one pixel row (i.e. $|I_i| = 1$ or $|I_j| = 1$), *straditize* sets the sample at exactly this location

2015), VAPOR (Clyne et al., 2007) or Avizo[1] (e.g. Böttinger and Röber, 2019; Nocke et al., 2015; Rautenhaus et al., 2018; Wong et al., 2014) whereas statistical or climate literature commonly uses R (R Core Team, 2019), Python (Oliphant, 2006; Perez et al., 2011), Climate Data Operators (CDOs) (Schulzweida, 2019) or other command-line tools.

This separation, however, devalues the interplay between the new knowledge from the visualization step, that commonly raises the need for more statistical and mathematical processing of the initial data. This calls for integrated and flexible tools that tackle both steps: the data processing and the visualization, a requirement that is currently not fulfilled by the visualization tools described above. An example software that integrates data processing and data visualization is provided with the Earth System Model Evaluation Tool (ESMValTool) (Eyring et al., 2016). This framework provides common diagnostics for ESMs to enable model intercomparisons. The tool, however, has limited interactivity and a slow learning curve for the implementation of new diagnostics.

This lack leads to large efforts of climate scientists to develop scripts for the data processing and visualization. They usually do not follow a systematic framework and as such need to be adapted every time a new project starts which also make them difficult to share with other researchers. The new *psyplot* framework wants to generalize this data processing and visualization by providing a framework that is highly flexible, interoperates with standard computational data processing tools and enables flexible visualizations and adaptations. The software is written in the programming language Python (Perez et al., 2011) and builds upon the visualization package *matplotlib* (Hunter, 2007) and the N-dimensional array processing package *xarray* (Hoyer and Hamman, 2017), that closely interoperates with the numeric packages numpy and scipy (Jones et al., 2001; Oliphant, 2006) and the parallel computing library dask (Dask Development Team, 2016). Due to the flexibility of Python, it can be used from the command-line, a GUI (section 4.3.3) or jupyter notebooks[2] (Kluyver et al., 2016). As such, it supports out-of-core computation (i.e. the processing of data too large to fit into memory), a rich set of visualization methods from matplotlib, and can be extended to other visualization packages, such as the 3D-visualization framework VTK (Sommer, 2019b).

The next section 4.3 provide an overview of the framework with ~~it's~~ its data model, plugins and GUI. Sections 4.4 and 4.5 finally discuss further usage and extensions to the software. For more information, usage and implementation examples I also refer to the online documentation `https://psyplot.readthedocs.io`.

4.3 The psyplot framework

The psyplot framework consists of three parts: The core structure that is built upon *xarray* and provides the general infrastructure (section 4.3.1), the plugins that use the plotting functionalities of *matplotlib* (section 4.3.2), and the GUI (section 4.3.3).

4.3.1 Data model

Psyplot and xarray

psyplot acts as a high-level interface into the packages *xarray* and *matplotlib*. The first one is a recent package for N-dimensional labeled arrays that adopts Unidata's

[1] `https://www.fei.com/software/avizo3d/`
[2] `https://jupyter.org/`

self-describing Common Data Model on which the network Common Data Form (netCDF) is built (Brown et al., 1993; Hoyer and Hamman, 2017; Rew and Davis, 1990). The package integrates with standard python from the python environment, such as the computing and analysis packages numpy (Oliphant, 2006), scipy (Jones et al., 2001; Oliphant, 2007), pandas (McKinney, 2010) and statsmodels (Seabold and Perktold, 2010), but also offers intuitive interfaces for other packages, such as a package for empirical orthogonal functions (EOFs, Dawson, 2016), CDOs (Müller, 2019), fourier transforms (Uchida et al., 2019) ~~any~~ and many more[3]. This large ~~range of extensibility~~ potential for extension distinguishes psyplot from other high-level visualization software, such as ParaView or Vapor, ~~and they can all~~ as such python packages can be implemented as a ~~formatoption (~~so-called *formatoption* (without hyphen, see below) or used in a pre-processing step.

Psyplot core structure

The core structure of psyplot consists of five base classes that interact with each other, the visualization objects *Plotter* and its *Formatoptions*, the data objects *DataArray*, an *InteractiveList* of them, and a collection of all of them, the psyplot *Project*. It is schematically visualized in figure 4.1.

The most high-level API object is the psyplot project that consists of multiple data objects that are (or are not) visualized. The main purpose is a parallel handling of multiple plots/arrays that may also interact with each other (e.g. through the sharing of ~~formatoptions~~*formatoptions*). It mainly spreads update commands to ~~it's~~ its contained objects, but also serves as a filter for the data objects. Furthermore, one project may be split up into sub projects which then only control a specific part of the main project, e.g. for a specific formatting of only a small part of the data.

The next level is the *DataArray* from the xarray package (or more explicitly, ~~it's~~ its accessor, the *InteractiveArray*[3]), that holds the data of one (or more) variables (e.g. temperature) and its corresponding coordinates (e.g. time, latitude, longitude, etc.). It may be one or multidimensional depending on the chosen visualization method. psyplot offers several methods to provide the coordinates for the plotting of different grids to make the visualization easier. The software can ~~interprete~~ interpret CF Conventions[4] and UGRID conventions for unstructured grids (Jagers et al., 2018).

Multiple of these arrays can also be grouped together into an *InteractiveList* that shall be visualized by the same plot method (e.g. multiple lines or a scalar field with overlying vector field).

The visualization part in the framework is managed by the *Plotter* class, a collection of multiple *Formatoptions*. Each plotter subclass is designed to visualize the data in a specific manner (e.g. via line plots, violin plots, or map plots) and is completely defined through it's ~~formatoptions~~*formatoptions*.

Formatoptions are the core of the psyplot structure. The standard functionality of a formatoption is to control the visual appearance of one aspect of the plot (e.g. through the colormap, figure title, etc.). It is, however, completely unlimited and can also do data manipulations or calculations. The psy-reg plugin for example (see section 4.3.2) implements a formatoption that performs a regression through the data that is then visualized. As mentioned earlier, each plotter is set up through ~~it's formatoptions~~ its *formatoptions* where each formatoption has a unique formatoption

[3] several packages related to xarray are listed in the docs at `http://xarray.pydata.org/en/stable/related-projects.html` and psyplots integration (accessors) in particular is shown at `https://psyplot.readthedocs.io/en/latest/accessors.html`.

[4] `http://cfconventions.org`

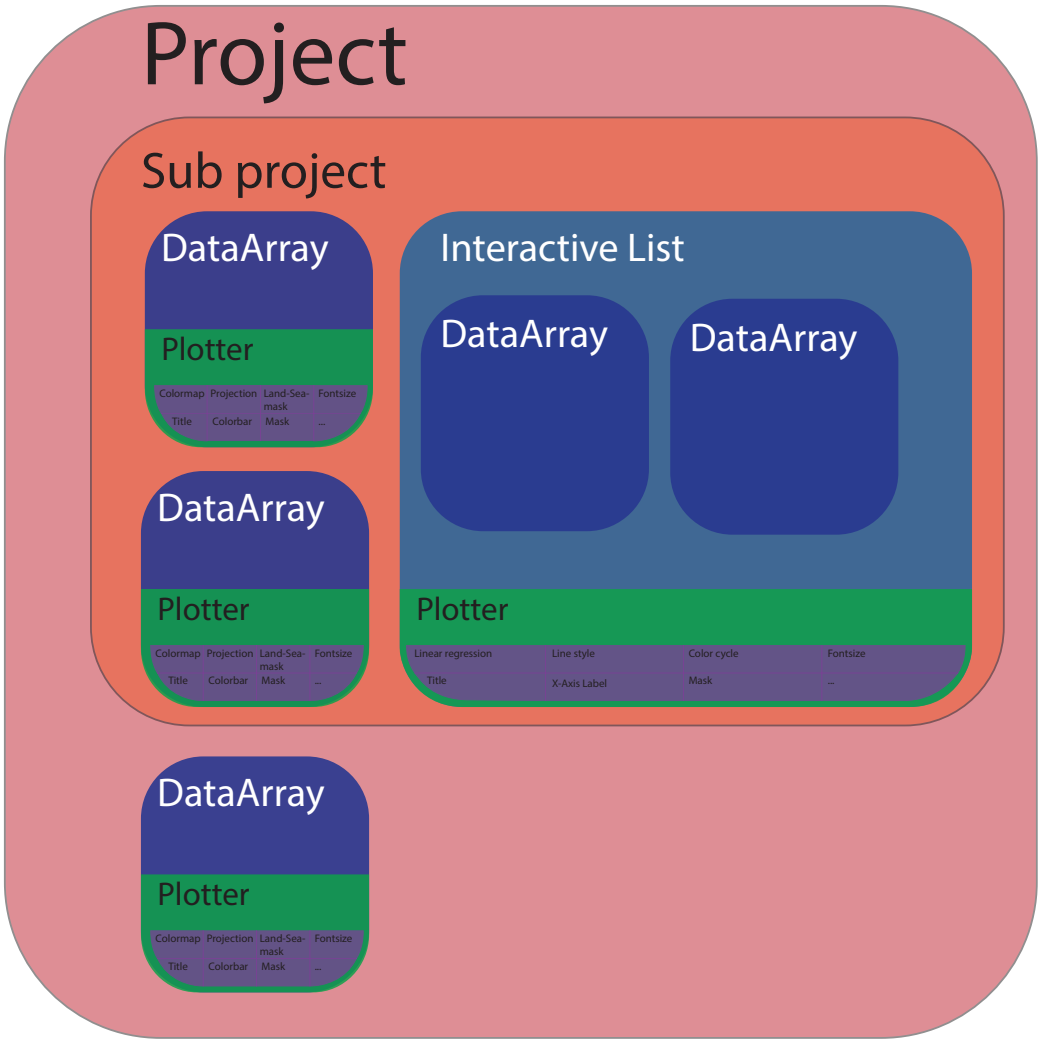

FIGURE 4.1: The psyplot core framework. A (sub) project consists of n-dimensional data arrays or a list of these that are each visualized by a plotter. Each plotter consists of a set of ~~formatoptions~~ *formatoptions* that control the appearance of the plot or performs data manipulation.

key inside the plotter. This formatoption key (e.g. *title* or *cmap*) is what is used for updating the plot, manipulating the data, etc.. ~~Formatoptions~~ *Formatoptions* might also interact with other ~~formatoptions~~ *formatoptions* inside the plotter or from other plotters. This concept of ~~formatoptions~~ *formatoptions* allows to use the same formatoption with all different kinds of plotters and the interaction of multiple plots with each other. Common plot features, such as the figure title, colormap, etc., therefore ~~don't~~ do not have to be implemented explicitly for every plotter but can be used from existing implementations. This framework also allows a very easy integration and development of own ~~formatoptions~~ *formatoptions* with a low or high level of complexity.

4.3.2 Psyplot plugins

The psyplot package provides the core of the data management described in the previous section 4.3.1. The real visualization is implemented in external plugins. The advantage of this approach is an increased flexibility of the entire framework (collaborations can evolve through dedicated plugins) and of managing the various dependencies of the packages. As such, the dependencies of psyplot are rather week (only xarray is needed), but the dependencies of the plugins can be more extensive (e.g. for geo-referencing or advanced statistics).

Each plugin defines new *Plotters* and *Formatoptions* that are specific to the purpose of the visualization/analysis task. The plotters can also be implemented as a plot method (see supplements 4.B to 4.E) and accessed through the psyplot core API (see supplements 4.A for an example).

The current lists of plugins include *psy-simple* for rather simple and standard visualization tasks, *psy-maps* for geo-referenced plots, *psy-reg* for statistical analysis visualization, and *psy-strat* for stratigraphic diagrams.

psy-simple: The psyplot plugin for simple visualizations

Much of the functionality that is used by other plugins is developed in the psy-simple plugin. This package targets simple visualizations and currently includes plot methods for one-dimensional data: line plots, bar plots and violin plots; for two-dimensional data: scalar plots, vector plots and combined scalar and vector plots; and plots that do not require complex data manipulation: a density plot and a plot of the weighted geographic mean. A full list of examples is provided in the supplementary material, section 4.B.

This package also implements most of the functionality to handle unstructured grids in 2D visualizations and defines most of the commonly used ~~formatoptions~~*formatoptions*. The latter include text manipulation (such as plot title, figure title, x- and y-axis labels, etc.), data masking, x- and y-axis tick labeling and positioning, as well as color coding for 2D plots (colormap, colormap sections, etc.).

psy-maps: The psyplot plugin for visualizations on a map

psy-maps builds on top of the psy-simple plugin and extends ~~it's~~ its functionality for visualizations on a map using the functionalities of the cartopy package (Met Office, 2010 - 2015) (see supplements 4.C for examples). It simplifies as such the automated generation of maps for climate model data through the flexibility of the psyplot framework.

psy-maps currently implements additional ~~formatoptions~~ *formatoptions* for choosing the projection of the map, selecting the geographic region, drawing the contintens or shaded reliefs of land and ocean, and more. One feature that distinguishes psy-maps from other visualization software, even from pure cartopy, is the ability to visualize unstructured geo-referenced grids on the map. For this purpose, triangles are projected in a pre-processing step to the target projection, prior to the visualization with matplotlib. This drastically increases the performance and makes it possible to visualize even very large data sets. As such, psy-maps visualizes a global scalar field on a hexagonal grid of roughly 4.4 million grid cells ($\approx$ 13 km resolution) in roughly 3.5 minutes. The interactive usage of such a large dataset is however limited by the functionalities of matplotlib to handle such an immense amount of data.

psy-reg: The psyplot plugin for visualizing and calculating regression plots

psy-reg performs regression analysis on 1D variables using the methods of the ~~statsmodels~~ stats-models (Seabold and Perktold, 2010) and scipy (Jones et al., 2001; Oliphant, 2007) packages, and visualizes the results with the functionalities of the psy-simple plugin. As such, it implements ~~formatoptions~~ *formatoptions* for univariate regressions, confidence intervals via bootstrapping, and combined plots of the data density and the fitted model (see also supplements 4.D). The necessity for this package arose from the need to visualize a regression model, compare it (visually) with the original data and to use it afterwards. Other python packages either focus only on the generation of the regressions (such as statsmodels or scipy), or on their visualization (such as seaborn (Waskom et al., 2018)). The psyplot plugin makes it possible to generate the visualization and to access the underlying regression model parameters and uncertainties.

psy-reg has been heavily used for the parameterization of the weather generator in chapter 6 which also gave the initial motivation for the package.

psy-strat: A psyplot plugin for stratigraphic plots

psy-strat (Sommer, 2019a) is the latest plugin for psyplot that has been developed for stratigraphic diagram visualization. It is particularly designed for the straditize software (Sommer et al., 2019, chapter 3) and was motivated by the need for an automated creation of pollen diagrams. One example of such a diagram is provided in the supplementary material, section 4.E.

As the psy-reg and psy-maps plugins, psy-strat uses the functionalities of the psy-simple plugin for a visualization of multiple variables in separate diagrams that share one common vertical axis (usually age or depth)[5]. Additionally, besides the integration that is common for every psyplot plugin (see next section 4.3.3), psy-strat contains additional functionalities for the psyplot GUI. This implementation allows the user to select and reorder the variables (pollen taxa) that are shown in the stratigraphic diagram.

4.3.3 The psyplot Graphical User Interface

Psyplots objective of providing a platform for flexible and convenient data analysis is further approached with the *psyplot-gui* package. This extension to the framwork provides a GUI for simplified access to the plotting features in psyplot.

[5]See psy-strat.readthedocs.io for an example of psy-strat.

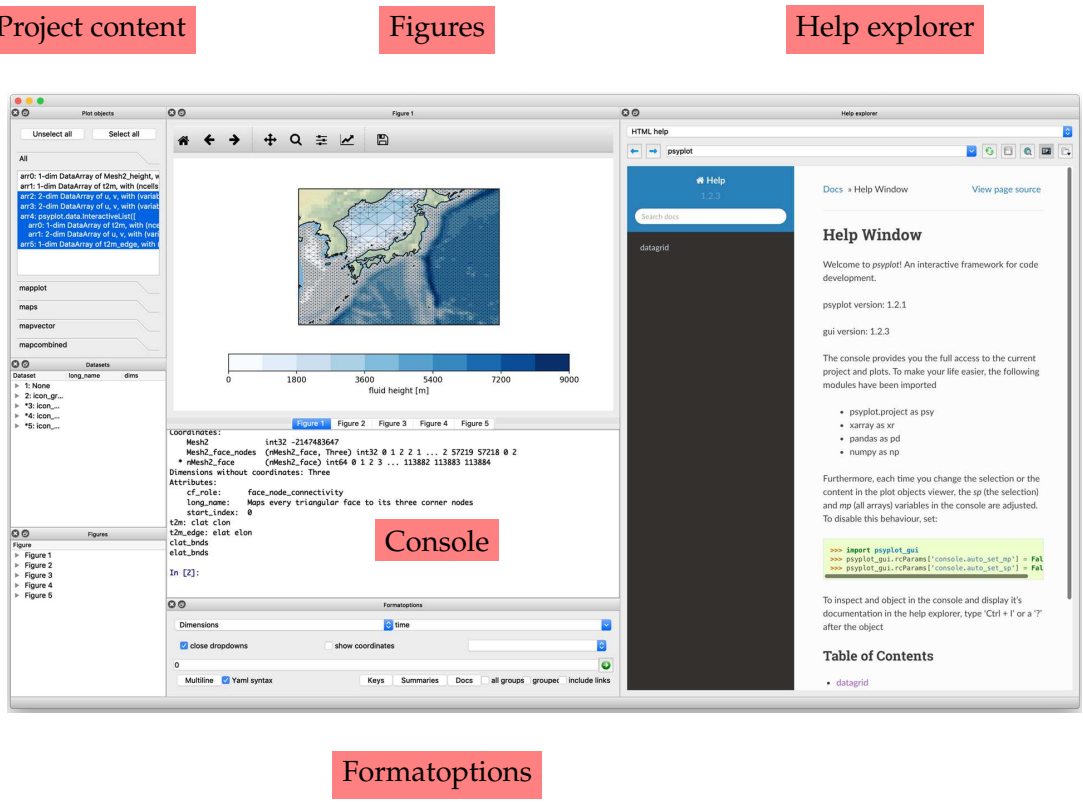

FIGURE 4.2: Screenshot of the psyplot GUI. The left part shows the content of the psyplot project, the upper center the plots, and the right part contains the help explorer. Below the plots, there is also the IPython console for the usage from the command line and a widget to update the ~~formatoptions~~ *formatoptions* of the current project.

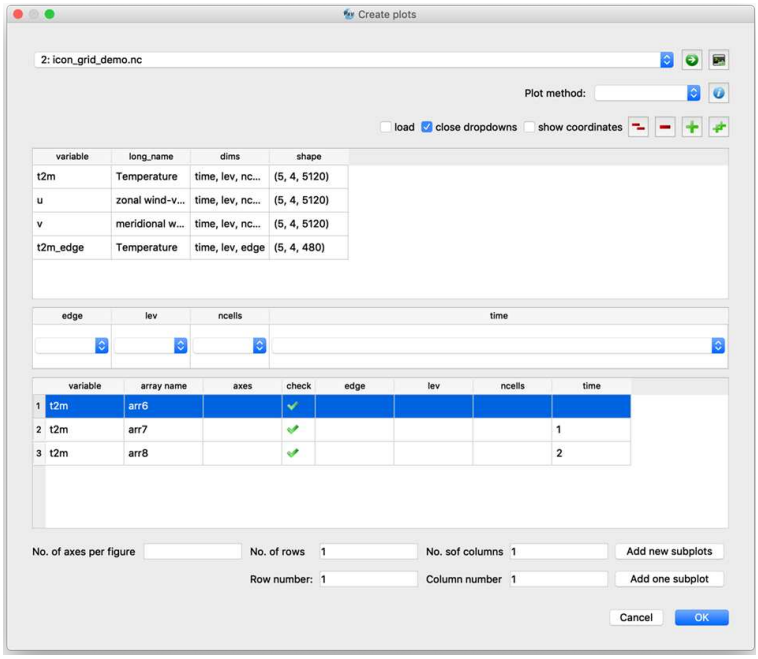

FIGURE 4.3: Plot creation dialog to generate new figures from an xarray dataset.

A strong focus of this interface is, again, the flexibility. psyplot-gui is based on the cross-platform PyQt5 library[6], a very flexible and frequently used package for graphical user interfaces. This enables other software to develop additional features for the package (see psy-strat in the previous section 4.3.2, for instance, or straditize in chapter 3) and to flexibly change the layout of the application. The GUI is complemented with an interactive console to provide a fully integrated python environment for data analysis.

The next paragraphs provide an overview on the various widgets, that are also displayed in figure 4.2 and 4.3.

Console

The central aspects to guarantee flexibility of the application is an in-process IPython console, based on the qtconsole package[7] that provides the possibility to communicate with the psyplot package via the command line and to load any other module or to run any other script or notebook, or even to run commands in different programming languages, such as R (R Core Team, 2019) or Julia (Bezanson et al., 2017). The console is fully integrated both ways into the GUI. The documentation of every python object in the terminal, for instance, can be viewed in the help explorer of the GUI. And vice versa: a change of the current project through the project content widgets, also changes the corresponding python variable in the shell.

Help explorer

As a complement to the console, the GUI contains a help explorer to provide immediate and dynamic access to the documentation of python objects in the console, rendered as an HTML webpage[8]. Furthermore, the help explorer is connected to multiple other widgets of the GUI in order to provide a dynamically generated documentation. The documentation of available ~~formatoptions~~ *formatoptions* in the psyplot project, for instance, are rendered as HTML upon request, in order to make the various plot methods more accessible. The same principle works for the plot methods that are accessible in the plot creator.

Plot creator

The plot creator (figure 4.3) is the starting point of the GUI into the psyplot framework (at least, if one does not use the console or a script to generate the plots). It loads data from the disk or the in-process console, and essentially provides a wrapper around the psyplot plotting call (see suppl. section 4.A). It additionally displays the documentation of the method and its associated ~~formatoptions~~*formatoptions*. This widget creates new plots, that are appended to the psyplot project and are accessible through the console and the project content widgets.

Project content

The psyplot project is the most high-level API element in the psyplot framework (see section 4.3.1) and is displayed in the project content widgets of the GUI. All

[6] PyQt5 can be accessed via `https://riverbankcomputing.com/software/pyqt/intro`.

[7] `https://github.com/jupyter/qtconsole`

[8] The help explorer widget has been originally motivated by the *Help* widget of the Scientific PYthon Development EnviRonment, Spyder (`https://www.spyder-ide.org/`) and uses the sphinx package (Hasecke, 2019) to convert restructured Text into HTML.

other elements, such as the ~~formatoptions~~ *formatoptions* widget or the plot creator, are interfering with the project, and it is accessible as a variable in the console. The project content widget can be used to see the various items in the project, but it is also used to select the specific items for the so-called *current* sub-project. The latter is dynamically set in the console through the `sp` variable and it is used by the ~~formatoptions~~ *formatoptions* widget to update the plotting parameters of the selected items.

Formatoptions

As mentioned in section 4.3.1, ~~formatoptions~~ *formatoptions* are the core elements in psyplot that control the figure aesthetics of the plots and/or perform data manipulations. The generic ~~formatoptions~~ *formatoptions* widget provides access to these parameters, in order to update them for the selected items in the current project. The formatoption itself (i.e. the python object) can in turn generate a widget that is implemented in the ~~formatoptions~~ *formatoptions* widget, to make the available options more accessible. The *title* formatoption, for instance, generates a drop-down menu to select variable attributes (e.g. variable name, variable units, etc.) which is then embedded in the ~~formatoptions~~ *formatoptions* widget. The modifications of the ~~formatoptions~~ *formatoptions* via this widgets, updates the figures of the selected items.

Figures and plots

The plots generated by the plotting methods are displayed in dedicated widgets inside the GUI and can be dynamically adjusted using the ~~formatoptions~~ *formatoptions* widget or the console. The underlying library of the current implemented psyplot plugins, matplotlib, implements multiple backends to display the data interactively, or to export them as PDF, PNG, etc. The psyplot GUI has implemented a backend on top of the PyQt5 backend of matplotlib, which embeds the figures in the GUI. psyplot can, however, work with any backend of matplotlib and does not depend on the specific implementation.

4.4 Conclusions

psyplot (Sommer, 2017e) is a new data visualization framework that integrates rich computational and mathematical software into a flexible framework for visualization. It differs from most of the visual analytic software such that it focuses on extensibility in order to flexibly tackle the different types of analysis questions that arise in pioneering research. The design of the high-level API of the framework enables a simple and standardized usage from the command-line, python scripts or jupyter notebooks. A modular plugin framework enables a flexible development of the framework that can potentially go into many different directions. The additional enhancement with a flexible GUI makes it the only visualization framework that can be handled from the conveniently command-line, and via point-click handling. It also allows to build further desktop applications on top of the existing framework.

The plugins of psyplot currently provide visualization methods that range from simple line plots, to density plots, regression analysis and geo-referenced visualization in two dimensions. The software is currently entirely based on the visualization methods of matplotlib (Hunter, 2007), the most established visualization package in the scientific python community. However, the framework itself is agnostic to

the underlying visualization method and can, as such, leverage a variety of existing analytical software.

4.5 Outlook

The possibilities for further development of the psyplot framework are numerous, due to ~~it's~~ its intrinsic generality. The core of the psyplot framework will, in the future, be extended with a standardized algorithm for the generation of animations. Psyplot projects already have the functionality of being saved to a file and reloaded, but they will also be exportable as python scripts for a more flexible reusability and adaptability. The update process within a psyplot project (currently every item in the project is updated in parallel) also has potential for improvement by using a single-threaded scheduler approach that better reflects if one formatoption depends on the ~~formatoptions~~ *formatoptions* of another plotter.

The GUI has especially high potential for further development, as it still lacks widgets to quickly and intuitively modify the visual appearance of the plots. The only possibility inside the GUI (besides the console) is to use the ~~formatoptions~~ *formatoptions* widget whose main focus however is on flexibility, rather than usability and has, as such, limited possibilities for adaptation to specific use cases.

Another focus will be the development of new plot methods inside the psyplot framework. The major aspect will be on the development of 3D visualization methods of geo-referenced data, using recently published software that builds on top of the visualization toolkit VTK (Sullivan and Kaszynski, 2019; Sullivan and Trainor-Guitton, 2019), see Sommer, 2019b. psyplot has the unique potential to generate 3D visualizations conveniently from the command line, a distinguishing feature, compared to other visualization software packages, such as ParaView or Vapor. Further potential enhancements for visualizations can involve standard interactive visual analytic tools, e.g. such that the interactive selection of features in one plot affects the visualization in another plot (so-called brushing and linking).

Chapter 5

pyleogrid

A Probabilistic Approach for Gridding Paleo Climate Data

5.1 Introduction

Paleo-climate reconstructions are most often undertaken on a site by site basis to provide a record of climate change at a specific place through time. The integration of data obtained from multiple sites however provides the basis for investigating spatially explicit reconstructions of climate through time. This spatio-temporal perspective can provide powerful insights into the climate system that are not easily discernible from the typical 1-dimensional approach associated with single site records. Spatially explicit data allows us to see how spatial patterns in climate variables change through time, providing a way of identifying the underlying causes of climate change. It also allows us to match the spatial scale of Earth-system models, which are based on grid-boxes that often reflect climatic changes at a very different spatial resolution than that experienced at the scale of a single site.

Here, we describe a computationally efficient methodology for integrating multiple paleo-climate records from different sites into a single spatio-temporal record that simultaneously takes into account the associated uncertainties. This method also involves projecting the data onto a uniform spatial grid and regular time-step. This approach is different from the conventional approach to gridding, often called *pseudo-gridding*, in which records that fall within a grid box are simply combined in some way to represent the grid box value (e.g. Bartlein et al., 2010; Marcott et al., 2013; Marsicek et al., 2018; Waelbroeck et al., 2009). Similarly, samples from the records within a grid box are also combined or binned into time-windows to create a regular time-step. ~~Instead, our methodology leaves the data where it is in time and space and interpolates it through onto~~ Our method instead does not aggregate the reconstruction spatially or temporally but rather interpolates the data to a user-defined 3-dimensional spatial grid and regular time-step. This approach has been used in previous studies (Davis et al., 2003; Mauri et al., 2014, 2015), but here we integrate chronological and reconstruction uncertainties into the gridding process, allowing us to propagate these uncertainties through time and space onto our grid network.

Gridding has many advantages over other simple mapping approaches, including *pseudo-gridding*. It allows us to calculate more accurately area-averages and energy balances, as well as to make direct comparisons with Earth system models at a comparable grid box size and regular time-step. Gridding also allows a changing paleo-climate site/sample network through time to be stabilized, making it easier to compare one time period with another. We are also able to create a more complete record of climate in time and space, using the entire sampling network to infer

climate in places and at times where we may not have a site/sample.

Our new method applies a probability approach to data integration, whereby the full uncertainty of each sample is considered in the gridding process, rather than just the sample mean used in previous ~~methodologies~~ methods (Mauri et al., 2015). From this, we can calculate the uncertainties associated with the temporal and spatial distances between our reconstruction samples/sites and the points on the grid network. We do this through an ensemble bootstrapping approach in which we repeatedly grid the data, each time using a different set of samples that are randomly ~~generated~~ selected to reflect the reconstruction and chronological uncertainties of the site network. In this study, we use pollen-data, which provides the most accessible and spatially distributed paleo-climate data available for the late-Quaternary period.

The strength of this new ~~methodology~~ method is that it treats all of the paleo-climate samples and sites as a single integrated paleo-climate record from which it is possible to extract a single regionally coherent climatic reconstruction, complete with uncertainties. Pollen-based reconstructions often have high sample to sample uncertainties, and the vegetation at individual sites can be influenced by non-climatic factors such as soils, disease, fire, migration lag and human impact. Our ~~methodology~~ method allows us to fully utilize the large quantity of pollen-data that is available, to extract the regional background climate signal from what may be locally quite noisy data. It also has a significant advantage over other methods that integrate records using Bayesian approaches (e.g. Holmström et al., 2015), in that it is much more computationally efficient, making it possible to undertake analysis at continental scales with hundreds and thousands of sites and samples.

5.2 Data

The ensemble based gridding method is adapted to paleo-climates. In this study, we describe the method using a large set of western Eurasian fossil pollen assemblages that have been transformed to summer (June, July and August) (JJA) temperatures. We focus on pollen data because it is the spatially most widely available proxy during the Holocene, but it is important to mention that the reconstruction method is agnostic to the climate proxy, because it does not explicitly use the pollen assemblages but rather alters the standard climate reconstruction method under the aspect of ~~it's~~ its methodological uncertainties. As such, the following sections describe the fossil and modern pollen database for this use case (section 5.2.1) and the associated uncertainties of the temperature reconstruction method (section 5.2.3) and the dating of the fossil pollen samples (section 5.2.4).

5.2.1 Pollen database

The source data for this study is a subset of the latest development version of the POLNET database, a northern hemispheric, extra-tropical collection of pollen assemblages (Davis and Kaplan, 2017; Sommer et al., 2019). The purpose of this database is to generate the source for large-scale climate reconstruction during the Holocene (past 12'000 years) that can be used for model-data comparisons. The subset that we use in this study to describe and develop the gridding method contains fossil and modern pollen assemblages of western Eurasia, a region that has already been under investigation in the previous study by Mauri et al., 2015.

The fossil database contains raw pollen counts with in total about 1350 datasets that consists of 80500 fossil samples. The majority of the fossil pollen data (left part

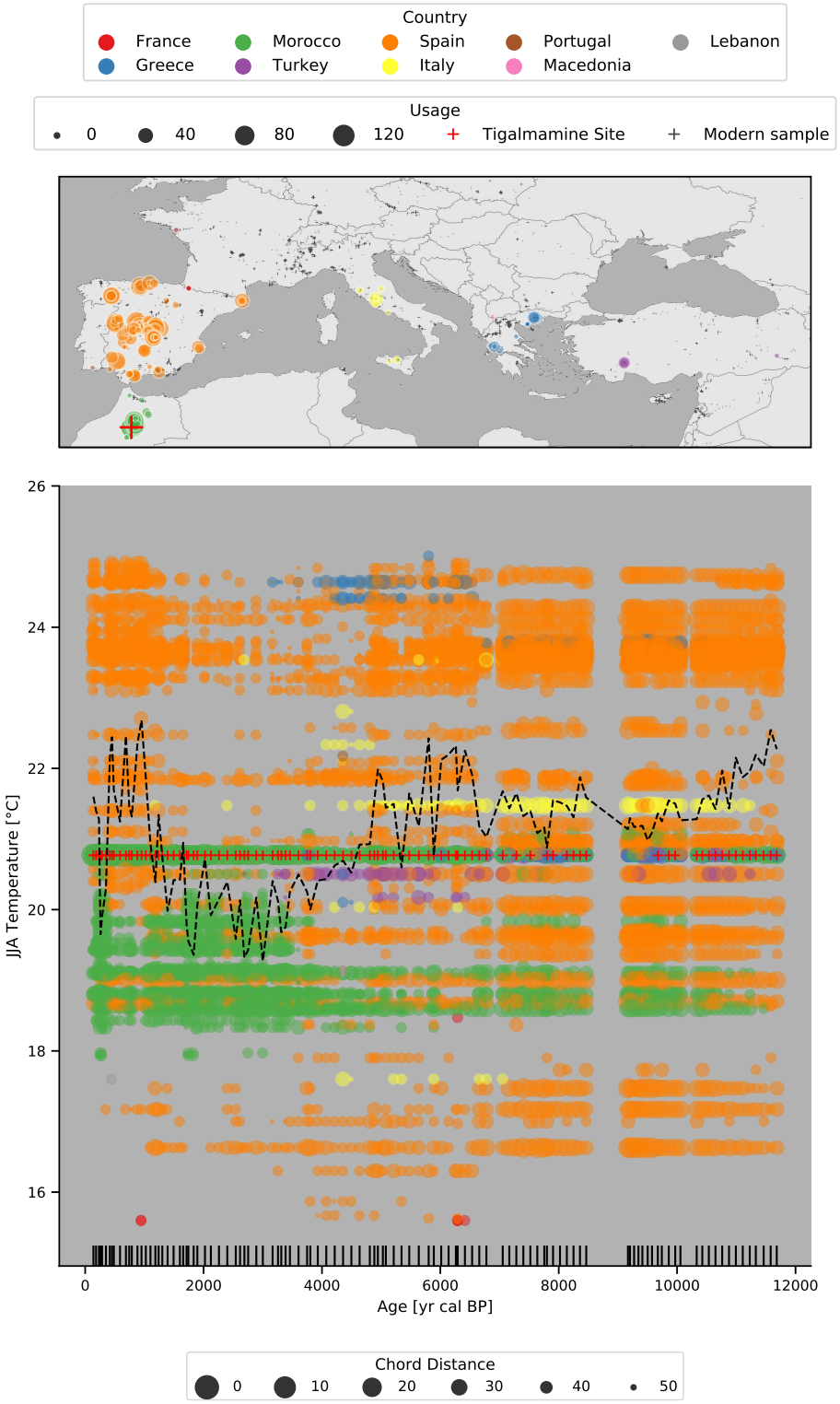

FIGURE 5.2: Climate analogues of the Tigalmamine site (red cross). Every circle corresponds to one modern analogue that was one of the fifties closest analogues in at least one sample within the Tigalmamine dataset. The color-coding of each circle is based ~~it's~~ its corresponding country (see legend at the top). The marker size in the top plot depends on the usage of the sample as modern analogue. The larger the marker, the more samples in the Tigalmamine dataset use it as modern analogue. Tiny crosses in the map show the locations of the rest of the modern calibration data. The lower plot shows the summer temperature for the analogue (y-axis) at the age of the Tigalmamine sample (x-axis). The marker size in this plot corresponds to the chord distance between modern and fossil pollen assemblage. I.e., the larger the dot, the closer (and more important) the analogue. The dashed line shows the weighted average of all the climate analogues per sample. The age of each Tigalmamine sample is shown with the vertical lines at the bottom of the plot. Red crosses in the lower plot show the Tigalmamine core top sample that has been used as an analogue in 58 out of the 110 samples.

matches. The early Holocene (12k to 8k BP) is dominated by modern samples from Spain, with a wider and more uniform temperature regime, when compared to the later periods. During the transition in the mid-Holocene (8k to 4k BP), analogues from across the Mediterranean Sea play a more important role, in particular from Greece, Italy and Turkey. The lower temperature regime is then dominated by Moroccan samples (green) that are of particular importance during the late Holocene (4k BP to present) due to the above mentioned presence of Cedrus.

The weighted average of the analogues (black dashed line in figure 5.2) is in general about one to two degrees lower than the one in Cheddadi et al., 1998 (very likely due to the above-mentioned erroneous calibration data they used). The trends are however similar: Higher temperatures in early Holocene (the *spanish analogues* dominate) with a drop around 6k BP (Moroccan climates). Our weighted average however also shows a clear increase during the past 2000 years, again driven by spanish analogues.

The climatic and geographic space that is covered by the analogues is further discussed in section 5.3.2.

5.2.3 Site-based holocene temperature estimates

A standard approach for site-based climate reconstruction from fossil pollen assemblages is the modern analogue technique (MAT) (also called *k*-nearest neighbors). This technique estimates the climate of the fossil sample as the (weighted) climate average of the most similar modern samples (i.e. the closest modern analogues). It has the major advantage that it requires little parameterization efforts and can be applied over a large spatial area that covers many different climate regimes (Mauri et al., 2015).

For this purpose, we follow the standard approach and assign ~~a~~ JJA temperature values ~~to~~ for each modern calibration sample (figure ??5.1), taken from the corresponding grid cell in the WorldClim dataset, version 2 at 30 seconds (Fick and Hijmans, 2017).

Every pollen assemblage is then transformed from raw counts to percentages, based on the total sum of terrestrial pollen counts per sample that we deem useful for the reconstruction. This excludes low samples with low counts and taxa with low occurrences. We use squared-chord distance from the R package *rioja* (Juggins, 2017) as a measure of similarity. For a given transformed fossil pollen assemblage $\{f_t\}$ and a modern pollen assemblage $\{m_t\}$, where $t = \{1, \ldots, N\}$ denotes one of the N individual taxa in the assemblages, this distance measure is defined as

$$d = \sum_{t=1,\ldots,N} \left(\sqrt{f_t} - \sqrt{m_t} \right)^2$$

This distance is calculated between every modern and every fossil sample in the entire database (section 5.2.1). The standard, non-probabilistic setup would now compute the climate of the fossil sample as the mean climate of the k closest analogues (e.g. $k = 6$), eventually weighted by their corresponding distance d. There are many variations of this technique (see for example Birks et al., 2010, including various measures of similarity, choices about k, the maximum allowed distance d between modern and fossil assemblage, subsampling of the calibration dataset to avoid spatial autocorrelation (Guiot and Vernal, 2011; Telford and Birks, 2005, 2009), and by grouping pollen taxa into so-called plant-functional types (PFTs) (Davis et

al., 2003; Mauri et al., 2015, e.g.). They all, however, have in common that the categorical, multi-modal distribution of the climate of the modern analogues is simplified into a unimodal distribution, represented by the mean of the analogue climates. Therefore, in our ensemble approach, we do not take the mean but sample the climate of the analogues directly. This is further discussed in the methods section 5.3.2 and 5.4.1.

5.2.4 Age uncertainties

In addition to the methodological uncertainties of the climate reconstruction method (previous section 5.2.3), we provide a framework to handle dating uncertainties. During the gridding step (see next section 5.3.3), every sample is weighted by the age difference to the target reconstruction age. The previous studies by Davis et al., 2003 and Mauri et al., 2015 do not take this uncertainty, that can be as high as multiple centuries, into account although they influence the gridded temperature reconstruction.

The reason is a systematic problem of pollen samples that we overcome here with the recent developments in the pollen community. In palynology, each sample in a sediment core is is dated using a so-called age-depth model, a function that maps each depth of the sediment core to an age. This function is based on a few chronological control points where the age has been determined instrumentally (for lake sediments in the Northern Hemisphere, these are commonly radiocarbon (^{14}C dates) and interpolates/extrapolates to the depths of the sample locations. Various methodologies exist to define these age-depth models, ranging from simple linear interpolation methods (Bennett, 1994) to the more recently developed bayesian techniques of the Bchron (Haslett and Parnell, 2008) and BACON (Blaauw and Christen, 2011) models.

The early approaches have been proven to provide unreliable uncertainty estimates (Telford et al., 2004) and there has been no standardized way to report the uncertainties, if they are reported at all. For this reason we (and previous studies) cannot rely on the age uncertainties reported in the pollen database. An alternative approach is to recalculate the chronology for every dataset in the database (see Goring, 2019, for instance), but this also requires parameterization for reliable uncertainties and goes beyond the scope of this study.

Instead, we follow an approach that is based on two aspects: age uncertainties are higher for older samples, and samples that are farther away from the radiocarbon dates (i.e. chronological control points). Additionally, samples behave differently if the sample is surrounded by two chronological points (i.e. the sampe age is interpolated) or not (sample age is extrapolated). To quantify these relationship, we perform a study based on all datasets (ca. 30'000 samples) from the Neotoma paleoecology database (Williams et al., 2018) that have age-depth models estimated with BACON, a model that has been proven to provide more reliable age uncertainty estimates (Trachsel and Telford, 2016). For the sake of implementation (the age sampling in section 5.3.1 assumes a normal distribution), we apply several assumptions and ~~proximations~~ approximations to the Neotoma samples, in particular:

1. We assume that every dataset with a BACON chronology in Neotoma keeps the defaults and reports the limits of the 95% confidence interval (CI)

2. We keep only the maximal distance of the CI limits from the reported age (i.e. we assume a symmetric distribution)

3. We assume that the distribution is normal (i.e. the 95% CI corresponds to the 2σ interval, where σ^2 denotes the scale parameter) and a division in half of the maximal distance (see previous assumption) gives the standard deviation σ (which is what we call the age uncertainty)

The resulting data is illustrated in figure 5.3. The grayscale density plots in the background shows the high dispersal of the data and the number of samples decreases strongly with higher distance to the control point or older samples (red lines). Nonetheless, the mean of the data (blue lines) reveals the increasing nature of both relationships, as mentioned before.

Figure 5.3 also shows two models that have been fitted to the data. The first one is a standard simple univariate linear model $y = a + b \cdot x$ (orange line). This model simulates the increasing trend of both variables although it does not capture the non-linear relationship between age and age-uncertainty. A reason for this non-linearity arises from the time-dependency of the radiocarbon calibration curve and ~~it's~~ its associated errors. This non-linear behavior gives the motivation to use a constrained linear Generalized Additive Model (GAM), a smooth semi-parametric model of the form

$$\mathbb{E}[y|X] = \beta_0 + f_1(X_1)$$

in the univariate case, or

$$\mathbb{E}[y|X] = \beta_0 + f_1(X_1) + f_2(X_2)$$

in the bivariate case. The feature functions f_1 and f_2 are based on penalized B splines with a constraint for monotonic increasing, the expected value $\mathbb{E}[y|X]$ is based on a normal distribution. The GAM model has been fitted with the *pyGAM* software package (Servén et al., 2018). This model enables to better simulate the non-linear features as can be seen with the green lines in figure 5.3.

These results approve the initial hypotheses and justify the choice of a bivariate GAM for predicting age uncertainties based on the distance to the chronological control point, and the age of the sample. The two models, together with a bivariate simple linear regression model, and again for interpolated and extrapolated samples, are shown in the central column of figure 5.4. Both model classes (simple linear and GAM) are able to reproduce the general shape of the observed data, although the GAM better resolves the non-linear relationship between the three variables.

The final uncertainties, predicted for the data set presented in the previous section 5.2.1, are shown in the supplementary figure 5.17.

5.3 Method

With the intrinsic methodological uncertainties of climate and dating in mind, we present a new ensemble-based approach on gridding the reconstructions from the individual sites. Each ensemble member is generated with randomized sample ages and climate, derived from the corresponding uncertainty measures (see previous sections 5.2.3 and 5.2.4), with additional constraints arising from the integrity of the individual dataset (sediment core). We explain these in more details in sections 5.3.1 and 5.3.2. The final gridding step for each ensemble member is based on a modified setup of Mauri et al., 2015, but can also be extended with other interpolation algorithms, as described in section 5.3.3). We implemented the method as the python package *pyleogrid* that efficiently scales to large datasets and ensemble sizes, and shortly describe it in section 5.3.4.

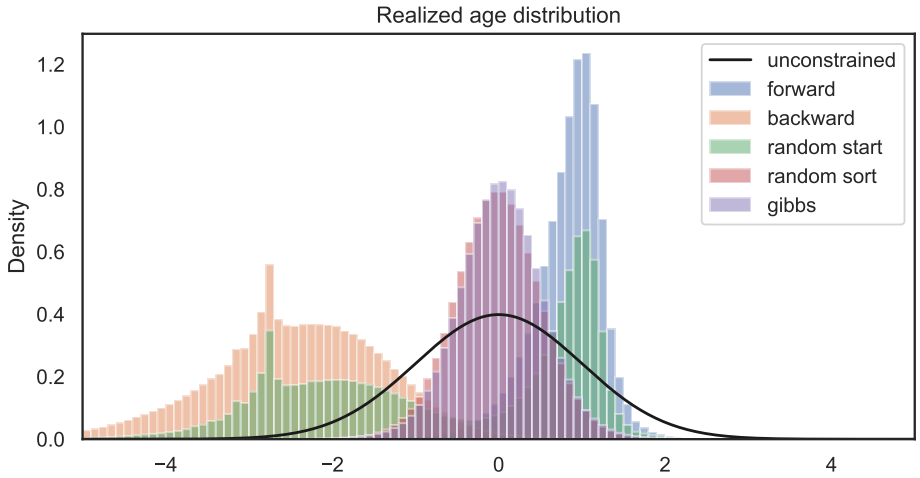

FIGURE 5.5: Histograms of age sampling methods for the site in section 5.2.2 with an ensemble size of 10'000. Every sampled age has been centered at the reported age of the corresponding sample and scaled by its age uncertainty. The black line shows the unconstrained distribution (a standard normal with a standard deviation of 1), the other histograms show the realized distributions for each of the age sampling methods (section 5.3.1). Note that *random sort* and *Gibbs* histograms highly overlap.

5.3.1 Constrained age sampling

Every dataset has an intrinsic monotonicity constraint that the sample deeper down the core has an older age. An inversion of this constraint is very rare and is usually visible in the stratigraphy of the core, such that affected samples are ruled-out before. As such, a classic unconstrained sampling of ages[1] using a normal distribution centered at reported sample age and a scale corresponding to the estimated age uncertainty (section 5.2.4) violates this constraint. Samples are inverted in such a case when their uncertainty intervals overlap and as such the individual ensemble member would not maintain the integrity of the individual core. We illustrate an example for such a core in section 5.4.1.

pyleogrid therefore implements different variants of this constraint with the Gibbs sampling being the one that is finally used.

The intuitive approach

The most intuitive approach is to randomly draw a sample age and ~~constraint~~ constrain the age of the neighboring sample with it. This can be done in a *forward* manner, such that every older sample has to be older than the previous younger sample, or in a backward manner, i.e. the younger sample has to be younger than the neighboring older sample. We will show in the paragraphs below that this method is biased, nevertheless we mention it here because of the intuitivity of the approach and because the reason for the failure is non-trivial.

As such, we demonstrate three different algorithms:

[1] We call it the unconstrained distribution for convenience, but keeping in mind that every sampled age has to be older than -70 yr cal BP.

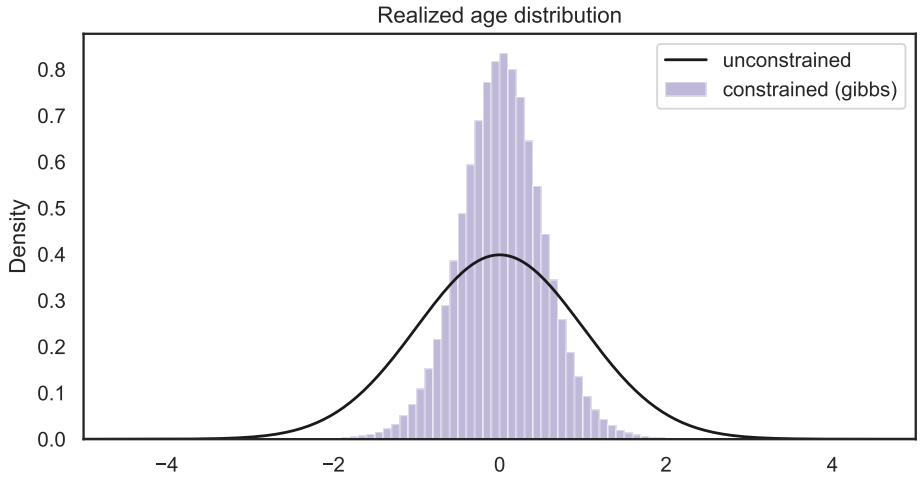

FIGURE 5.6: Realized age distribution for the entire dataset (section 5.2.1) with the *random* method (section 5.3.1). The individual sample distributions have been centered and scaled as in figure 5.5.

sample based on its unconstrained distribution[1]. In the second step, we order these random ages while maintaining the order of samples in each dataset. As such, we assign an age to each sample that is not necessarily drawn from its own distribution, but rather from the one of a neighboring sample. When samples overlap, this then truncates the tails of realized distribution and effectively decreases the reported age uncertainty, as can be seen in the figures 5.5, 5.18. This approach is mathematically difficult to justify because it violates the common methodology that each sample has a unique confidence interval that it needs to explore. Therefore the method might introduce some hidden biases in the sampled distributions that are difficult to quantify. Nevertheless, the algorithm is very fast and much closer to the desired joint distribution, than the previous *intuitive* approach. But in order to avoid any biases and to guarantee a mathematically correct result, we chose to implement a Gibbs sampling algorithm to sample from the desired constrained distribution.

The Gibbs sampling approach

Algorithm 1 Accept/Reject algorithm. $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution with location parameter μ and shape parameter σ.

Set $i = 0$
Set $\boldsymbol{\mu}$ as vector of the reported ages in *dataset*
Set $\boldsymbol{\sigma}$ as vector of estimated age uncertainties
Set $\mathbf{a}$ (the target age vector) to be of length $\boldsymbol{\mu}$
while $i < 1$ **or not** *is_monotonic*$(\mathbf{a})$ **do**
    $\mathbf{a} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
    Set $i = i + 1$
end while

The biases of the above-mentioned algorithms led to the development of a Markov chain Monte Carlo (MCMC) sampling algorithm. An accept/reject algorithm, which draws a set of random ages for all unconstrained sample distributions in a core at

once and accepts the draw if the monotonicity condition is satisfied and rejects the sample if the monotonicity condition was initially explored. For one realization of the ages **a** in a given dataset, this is described with the pseudo-code in algorithm 1. This standard approach however did not find a monotonic solution within ten million iterations for a high-resolution site such as it has been used in the previous section.

Therefore we decided to implement a Gibbs sampler, an algorithm that is commonly used in Bayesian inference to obtain a sequence of samples from conditional probability distributions, which generate samples from a multivariate joint distribution when this distribution is unknown and/or cannot be sampled directly. In our case, this distribution if the distribution of all sample ages in one dataset, where each sample age is conditioned by ~~it's~~ its younger and older neighbor. Let $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots \mu_N)$ be the reported ages of the N pollen samples in one individual dataset with estimated age uncertainties $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots \sigma_N)$. The reported ages fulfill the monotonicity constrain, i.e. $\mu_j \leq \mu_k$ for all j, k with $1 \leq j \leq k \leq N$. The objective of our sampling approach is to generate M random realizations of $\boldsymbol{\mu}$, denoted by $\mathbf{X}^{(m)} = \left(X_1^{(m)}, X_2^{(m)}, \dots X_N^{(m)}\right)$ with $m = 1, \dots, M$, that all fulfill the monotonicity constrain. In other words, the realizations $\mathbf{X}^{(m)}$ are constrained to fulfill

$$X_j^{(m)} \leq X_k^{(m)}, \text{ for all } j, k \text{ with } 1 \leq j \leq k \leq N \text{ and } 1 \leq m \leq M. \tag{5.1}$$

We set the intial value to the reported ages ($\mathbf{X}^{(1)} = \boldsymbol{\mu}$) where we know that the constrain is fullfilled. For the following realizations $\mathbf{X}^{(m+1)}$ with $1 < m \leq M$ we sample each component $X_j^{(m)}$ with $1 \leq j \leq N$ conditioned by its previous sample $X_{j-1}^{(m)}$ and, most importantly, conditioned by the next sample, but from the previous realization, i.e. $X_{j+1}^{(m-1)}$. As such, we define the sampled age of $X_j^{(m)}$ with

$$X_j^m = \mathcal{N}(X_{j-1}^{(m)}; X_{j+1}^{(m-1)}; \mu_j, \sigma_j^2) \tag{5.2}$$

where $\mathcal{N}(a; b; \cdot, \cdot)$ denotes a random variate of the truncated normal distribution with lower limit a and upper limit b. Although this algorithm always starts with the youngest sample in the dataset for every realization, such as the *forward* method, it does not push every sample to the lower tail of the distribution because every sampled age is conditioned by the age of the next pollen sample from the previous realization. It is mathematically proven that the combined realizations $\left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots \mathbf{X}^{(M)}\right\}$ of this algorithm approximates the joint distribution of the sample ages in the dataset under the given constraint, and that each marginal distribution of a the age of a particular pollen sample $1 \leq j \leq N$ is approximated by $\left\{X_j^{(1)}, X_j^{(2)}, \dots, X_j^{(M)}\right\}$.

As it is common for a MCMC algorithm, each realization is correlated with nearby realizations. The first samples are particularly correlated with the initial value $\boldsymbol{\mu}$ and it is therefore common practice to discard the first 1000 realizations, the so-called *burn-in* period.

To avoid an autocorrelation between the successive realizations, we *thin* our set of sampled ages and keep only every tenth realization until we have the desired amount of M realizations. The value of 10 has been shown to be sufficient using an autocorrelation analysis of the different samples in the Tigalmamine record.

As can be seen in figure 5.5 and 5.18, the outcomes are very close to the above mentioned *random sort* approach. However, a look into the realized distribution of

the last sample in figure 5.18 reveals a negative bias of the distribution sampled with the *random sort* approach.

The realized (and standardized) distribution of the entire database presented in section 5.2.1 is finally shown in figure 5.6. The comparison with the unconstrained distribution in this figure highlights the need for a constrained sampling because the latter significantly reduces the width of the distribution.

5.3.2 Temperature sampling

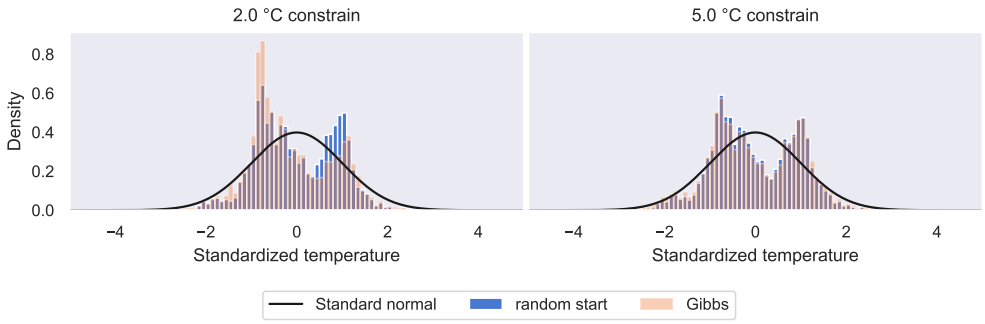

FIGURE 5.7: Histograms of temperature sampling methods for the site in section 5.2.2 with an ensemble size of 10'000 for a climatic constraint of (left) 2 °C, and (right) 5 °C. As in figure 5.5, every sampled temperature has been centered at the weighted average of the corresponding modern analogues and scaled by the corresponding weighted standard deviation. The black line shows the unconstrained distribution (a standard normal with a standard deviation of 1), the other histograms show the realized distributions for *random start* and *Gibbs* temperature sampling method (section 5.3.1).

As already mentioned in 5.2.3, our sampling approach does not use the temperature and uncertainty reported for every single variable. Instead, it samples the underlying distribution. As such, our method can be adapted to multiple site-specific reconstruction methods, such as weighted averaging (WA), weighted-averaging ~~partial least~~ partial least squares (WAPLS) (Birks et al., 1990; Braak and Juggins, 1993) or other approaches (e.g. Birks et al., 2010; Brewer et al., 2007; Juggins, 2013). In this study, we use a modern analogue technique (MAT) approach (see section 5.2.3) and sample the discrete set of climate analogues for each sample. The probability to select an analogue (i.e. ~~it's~~ its weight) is thereby determined by the chord distance between the fossil and modern pollen assemblages. The closer the assemblages (relative to the other potential analogues), the higher the weight.

This methodology is substantially different from the standard approach, such that ~~is~~ it takes the multimodality of the analogues into account, whereas the standard ~~appraoch~~ approach (weighted average of the k closest analogues) estimates a unimodal distribution. It additionally better represents the discrete nature of the analogue approach whereas the standard method intrinsically assumes a continuity in the distribution. In fact, only a small part of the available climate space is actually represented by the modern analogues. The 5'500 modern analogues for Tigalmamine, for instance (110 samples with 50 analogues each), are represented by only 240 distinct modern pollen samples with only 131 distinct JJA temperatures and they eventually span a large climate space (see figure 5.2).

The latter gives the motivation for a climatic constraint that ensures the integrity of the individual dataset. It is, for instance, impossible that two samples from the

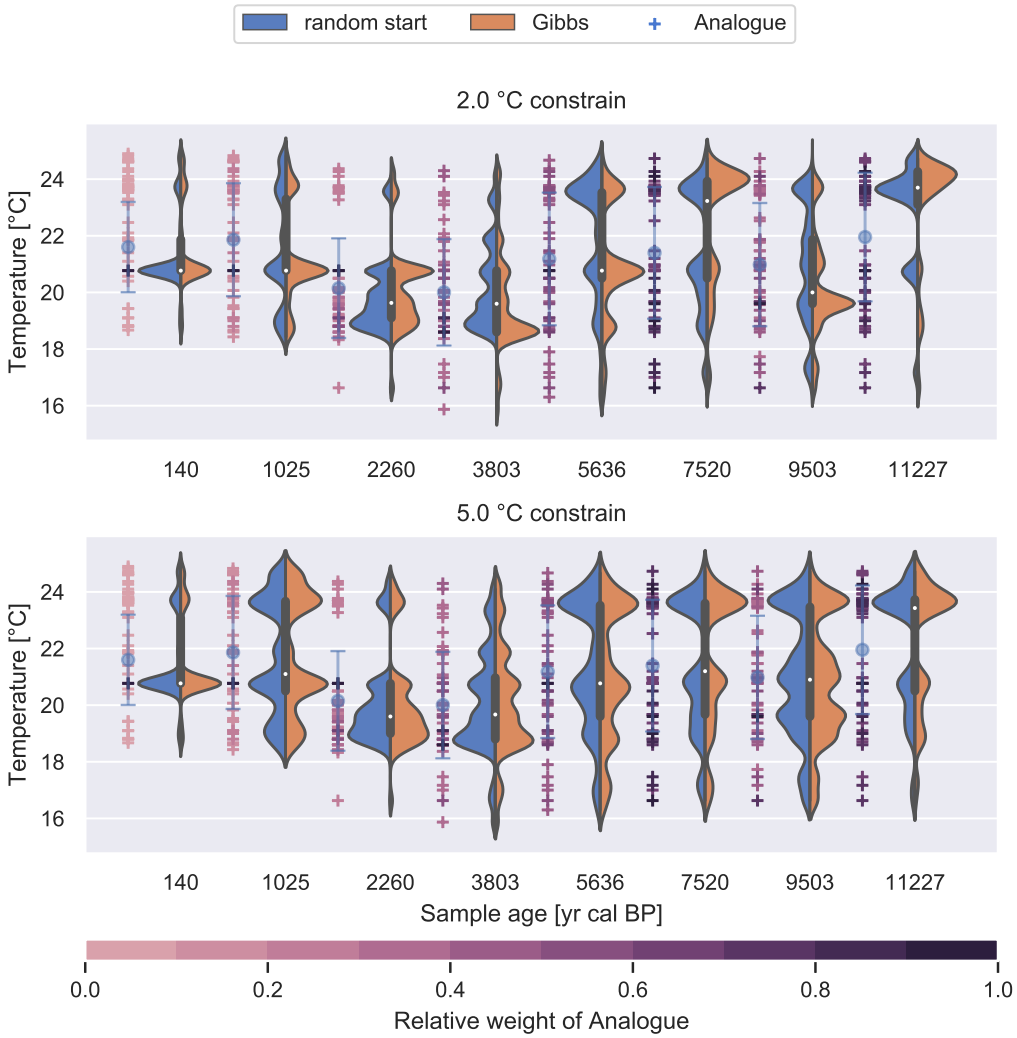

FIGURE 5.8: Violin plots for sampled temperature distributions of some samples in the Tigalmamine record with a climatic threshold of (top) 2 and (bottom) 5 °C. Blue (left) distributions are realized with the *random start* method, ocher (right) areas with Gibbs sampling. The crosses to the left of each violin shows the locations of the climate analogues. Each cross is color coded by ~~it's~~ its chord distance relative to the chord distance of the closest analogue (i.e. the closest analogue has always a weight of 1).

same dataset but 200 years apart experience a temperature difference of five degrees or more between each other. This is, however, a possible combination, considering the underlying set of analogues (see figure 5.2 for instance). We therefore perform a constrained sampling, as in section 5.3.1, and implement a fixed temperature threshold T. Every sampled analogue in each dataset (i.e. every choice of the discrete distribution for each pollen sample) is constraint to not differ by more than T ~~degrees celsius~~ degree Celsius from its temporally neighboring samples. The exact choice of T is a critical assumption and has a major impact on the realized temperature distribution for each sample. We decided for a very conservative estimate of five ~~degrees~~degree Celsius, which is only applied if the samples do not differ by more than 1000 years. These choices are further discussed in section 5.4.

In the remainder of this section, we focus on the implementation of this conditional sampling, because of its substantial impact on the realized distribution, as

have a strong influence on the outcome. The effects of a reduced number of analogues (figures 5.12 and 5.15), as well as a stronger climatic constraint (figures 5.11 and 5.14) resulted in more extreme temperature anomalies over the same geographic regions. This result is coherent with the sampled temperatures presented in the methods section (figure 5.8) where we show that a higher climatic constraint particularly strengthens the closer climate analogues and as such effectively reduces the number of analogues that is used in the reconstruction. There is no clear and definitive answer to the question how many analogues one should use and how strong the climatic constraint should be. Based on the results however, we favor a less strong climatic constraint of 5 °C and a high number of analogues (50). The philosophy behind is to let the method choose which analogue it uses. If there are inconsistencies within the set of modern analogues for a particular dataset, then this should be reflected by the ensemble standard deviation, and can eventually be compensated by the spatially neighboring samples in the gridding process.

These are all problems that arise from the underlying discrete distribution of modern analogues. An alternativity would be to use a PDF based method (Chevalier et al., 2014; Chevalier, 2019, for instance) that can potentially overcome these weeknesses by providing a more continuous distribution for sampling the climate.

The third aspect of the uncertainty is related to the gridding algorithm itself which can be added on top of the uncertainty of our ensemble method. For the *Tps* method, this uncertainty can be estimated conveniently using the *predictSE* function of the *fields* R-package that approximates the covariances of the prediction based on a linear combination of the observed data under the assumption of fixed covariance parameters (see Nychka et al., 2017 for details). A calculation of this standard error for 20 ensemble members revealed that it is rather independent of the individual realization. As such, we present the averaged standard error of these 20 members in the supplementary figure 5.20. We can see that this uncertainty estimate is high towards boundaries of the interpolation domain, but smaller than the ensemble standard deviations in between.

5.6 Conclusions

With *pyleogrid* we ~~are presenting~~ present a new methodological framework that transforms multiple site-based proxy-climate reconstructions into a joint spatio-temporal probabilistic climate reconstruction. Our method exploits the climatic and temporal space that is spanned by the intrinsic uncertainties related to the proxy-climate reconstruction method and the dating of the samples, in order to approximate the distribution of potential climate states in the geographic area of interest. Our approach requires little parameterization, is computationally efficient and can be scaled to large hemispheric or even global areas. The generic ensemble approach we present is in principle agnostic to the underlying proxy-climate reconstruction method and to the gridding method and can therefore be extended to a wide range of potential applications.

Compared to previous approaches of a large-scale gridded reconstruction, our method therefore provides a more reliable uncertainty estimate based on our constrained sampling approaches, which is essential for the comparison with climate model output.

The methodology comes with two side-products that are essential for the spatio-temporal ensemble approach. The first one is a methodology to estimate dating

uncertainties based on a bivariate model of the age of the sample and its temporal distance to the closest chronological control point in the dataset. This model is based on all samples with BACON-based chronologies from the Neotoma database under the assumption that they all report the age uncertainties as the 95th percent confidence interval. This strong assumption can be relaxed for future improvements of this method by using the very recent peer-reviewed dataset by Wang et al., 2019 which contains standardized chronologies for more than 500 datasets.

The other side-product, is a probabilistic variant of the site-based modern analogue technique (MAT) that uses a Gibbs sampling algorithm for the ages and analogue climates in order to approximate the joint distribution within a single dataset. This sampling algorithm is constrained for the individual dataset through first, the monotonicity of the sample ages, and second, a climatic threshold that must not be overcome between too temporally neighboring samples. We compare this sampling algorithm to computationally faster algorithms that involve a forward sampling (climate/age of a sample is constrained by ~~it's~~ its younger predecessor), its inversion, the backward sampling, as well as a combined approach that uses forward and backward sampling. For age sampling we also test an algorithm that starts with an unconstrained sampling of the ages and then applies an a posteriori sorting in order to maintain the correct distribution of samples in the core (sections 5.3.1 and 5.3.2). All of these approaches, however reveal biases when compared to the computationally more demanding, but ~~stationary~~ statistically correct distribution of the Gibbs sampler. The software package *pyleogrid* therefore implements a computationally efficient version of this Gibbs sampling algorithm that efficiently scales from one single dataset to tenth of thousands of realizations of more than a thousand individual datasets, as it has been used in this study.

This sampling successfully reconstructs a probabilistic version of the individual dataset and provides reliable uncertainty estimates. An evaluation of this realization with respect to the number of modern analogues, and the climatic constraint of the sampling algorithm reveals a close linkage of the two parameters. Further developments might therefore explore the usage of other proxy-climate reconstruction methods that provide a more continuous distribution to sample from.

station-based monthly climate dataset was available at this time (Walter and Lieth, 1967), limitations in computers and storage allowed only the simplest treatment of these data. The first global simulations of the net primary productivity of the terrestrial biosphere (Lieth, 1975), thus used rasterized polygons of annual meteorological variables that had been crudely interpolated from the station-based climatology. A decade later saw the development of better computers and more sophisticated global vegetation models (Prentice et al., 1992; Prentice, 1989) that recognized the need for forcing at a sub-annual timestep and development of these models was done in parallel with the first global, gridded high resolution (0.5°) monthly climatology (Leemans and Cramer, 1991). At the time, monthly meteorological data was the only feasible global data that could be produced, in terms of the raw station data available to feed the interpolation process, the processing time required to produce gridded maps, and the data storage and transfer capabilities of contemporary computer systems and networks. Global gridded monthly climate data thus became the standard for not only large-extent vegetation modeling (Haxeltine and Prentice, 1996; Haxeltine et al., 1996; Kaplan et al., 2003; Kucharik et al., 2000; Woodward et al., 1995), but also for a wide range of studies on biodiversity and species distribution (e.g., Elith et al., 2006), vegetation trace gas emissions (e.g., Guenther et al., 1995), and even the geographic distribution of human diseases (e.g., Bhatt et al., 2013)

Over subsequent years, the global gridded monthly climate datasets were improved (New et al., 1999, 2002), developed with very high spatial resolution (Hijmans et al., 2005), and expanded beyond ~~climatological~~ mean climate to cover continuous timeseries over decades (Harris et al., 2014; Mitchell and Jones, 2005; New et al., 2000). The latter was an essential requirement for forcing dynamic global vegetation models (DGVMs) (e.g., Sitch et al., 2003). However, despite increasing quality, spatial resolution, and temporal extent in these datasets, the basic time step remained monthly, partly for legacy reasons — models had been developed in an earlier era subject to computational limitations and therefore used a monthly timestep for efficiency even if this was no longer strictly a constraint — and partly because of the challenge in developing a global, high-resolution climate dataset with a daily or shorter timestep still presented a major data management challenge.

On the other hand, there was increasing awareness that accurate simulation of many earth surface processes required representation of processes at a shorter-than-monthly timestep. Global simulation of surface hydrology (Gerten et al., 2004), crop growth (Bondeau et al., 2007), or biogeophysical processes (Krinner et al., 2005) needed sub-monthly forcing to produce reliable results. To address this need for better forcing data, two main approaches were taken: either monthly climate data were downscaled online using a stochastic weather generator (e.g., Pfeiffer et al., 2013), or a sub-daily, high-resolution, gridded climate timeseries was generated directly by merging high-temporal-resolution reanalysis data (e.g., NCEP, 6h, 2.5°) with high-spatial-resolution monthly climate data (e.g., CRU, 0.5°). The latter process resulted in the CRUNCEP dataset (Viovy and Ciais, 2016; Wei et al., 2014), which, while global, is large even by modern standards (ca. 350 Gb), is not available at spatial resolution greater than 0.5°, and covers only the period 1901-2014.

Forcing data for global vegetation and other models with shorter than monthly resolution at higher spatial resolutions than 0.5°, or for any other period than the last ca. 120 years, e.g., for the future or the more distant past, may therefore only be available through downscaling techniques. One approach to overcome the limitations of currently available datasets could be to use GCM output directly, however, most GCM output currently available does not have greater than 0.5° spatial resolution,

wet days), 3-4) monthly mean daily minimum and maximum temperature, 5) mean cloud fraction, and 6) wind speed. The model outputs are the same variables at daily resolution. This section summarizes the basic workflow in the model which is also shown schematically in Figure 6.1 and Algorithm 2.

The first approximation of the daily variables comes from smoothing the monthly time series using a mean-preserving algorithm (Rymes and Myers, 2001).

For precipitation we then first use the Markov Chain approach (section 6.3.2) to decide the wet/dry state of the day. If it is a wet day, we calculate the gamma parameters using the equations (6.7) and (6.8). The resulting distribution allows us to draw a random number, the precipitation amount of the currently simulated day. If we are above the threshold μ, we draw a second random number from the ~~GP~~ Generalized Pareto (GP) distribution parameterized via equation (6.9) and the chosen GP shape.

The next step modifies the means of temperature, wind speed and cloud fraction depending on the wet/dry state of the day (lines 11 and 15 in algorithm 2). After that, we use the cross-correlation approach described in Richardson, 1981 (lines 18 - 20 and Equation 6.3.2) and calculate the daily values of these variables. Finally we use the quantile-based bias correction described in section 6.3.4 to correct the simulated wind speed.

We restrict the weather generator to reproduce the exact number of wet days (± 1) as the input and to be within a 5% range of the total monthly precipitation (with a maximum allowed deviation of 0.5 mm). If the program cannot produce these results, the procedure described above is repeated (see line 4).

6.3 Model development

GWGEN is based on the WGEN weather generator (Richardson, 1981), using the method of defining the model parameters based on monthly summaries described by Geng et al., 1986 and Geng and Auburn, 1987. GWGEN diverges from the original WGEN by using a hybrid-order Markov chain to simulate precipitation occurrence (Wilks, 1999a), and a hybrid Gamma-GP distribution (Furrer and Katz, 2008; Neykov et al., 2014) to estimate precipitation amount. Temperature, cloud cover, and wind speed are calculated following (Richardson, 1981), using cross correlation and depending on the wet/dry-state of the day. We further add a quantile-based bias correction for wind speed and minimum temperature, which improves the simulation results significantly.

In the following subsections, we first describe the global weather station database used to develop and evaluate the model, then describe the underlying relationships that we use to define GWGEN's parameters.

6.3.1 Development of a global weather station database

To parameterize GWGEN, we assembled a global dataset of daily meteorological observations. Precipitation and minimum and maximum daily temperature come from the daily Global Historical Climatology Network (GHCN-Daily) database (Menne et al., 2012a,b). The GHCN-Daily consists of observations collected at ca. 100'000 weather stations on all continents and many oceanic islands. As the GHCN-Daily stations are highly concentrated in some parts of the world, particularly in the conterminous United States, we selected stations for our study using a geographic anti-aliasing filter to avoid an especially strong geographic bias in the generation of the

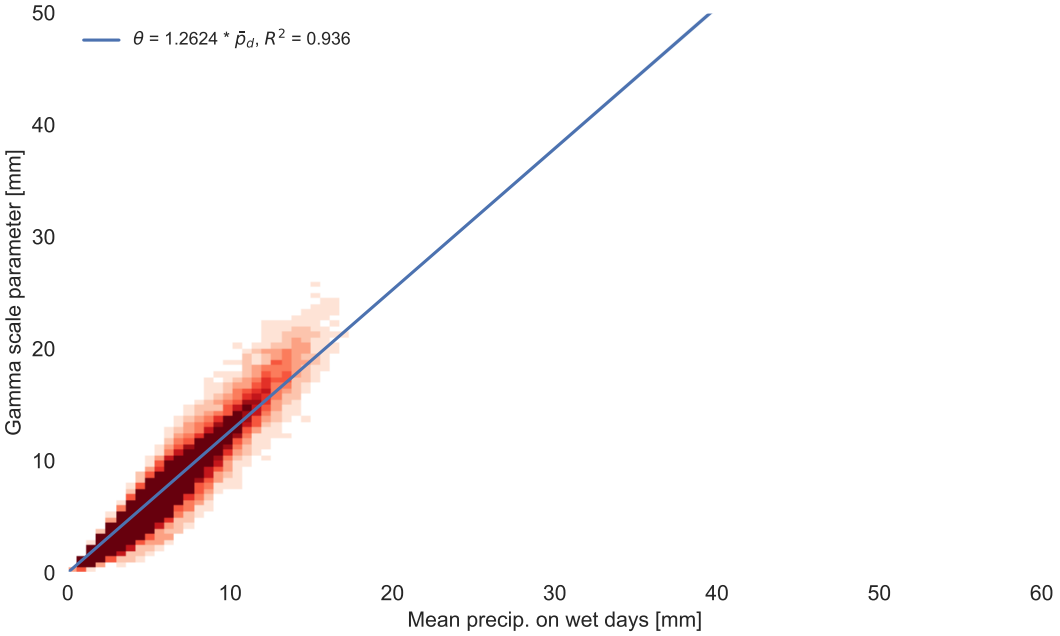

FIGURE 6.4: Mean precipitation - Gamma scale relationship. The blue line represents the best fit line of the mean precipitation on wet days to the estimated gamma scale parameter of the corresponding distribution. Each data point corresponds to one multi-year series of one month for one station.

in the month with precipitation, which show the characteristic linear relationship described by Geng et al., 1986 (Figure 6.3).

Because the transition probabilities (p_{001}) and (p_{101}) must be zero by definition when the fraction of wet days (f_{wet}) is zero, i.e., a completely dry month, we force the linear regression between these quantities to pass through the origin. Likewise, we require the regression line for (p_{11}) to equal 1 when f_{wet} is 1. One has to note, however, that this ~~methodology~~ method artificially increases the R^2 coefficient for the fit because we fix the intercept (see for example Gordon, 1981).

The analysis results in the the following relationships:

$$p_{11} = 0.2549 + 0.7451 \cdot f_{\text{wet}} \tag{6.1}$$

$$p_{101} = 0.8463 \cdot f_{\text{wet}} \tag{6.2}$$

$$p_{001} = 0.7240 \cdot f_{\text{wet}}. \tag{6.3}$$

In the weather generator (see line 6 in algorithm 2) we determine if any given day will have precipitation by calculating the appropriate probability density function selected from equations (6.1)-(6.3) on the basis of the precipitation state of the previous day (or two). Comparing the calculated probability from the selected equation with a random number $u \in [0, 1]$, a precipitation day is simulated if u is greater than its corresponding probability.

Precipitation amount

Following the original WGEN (Richardson, 1981), GWGEN disaggregates precipitation amount using a statistical distribution. A number of different probability

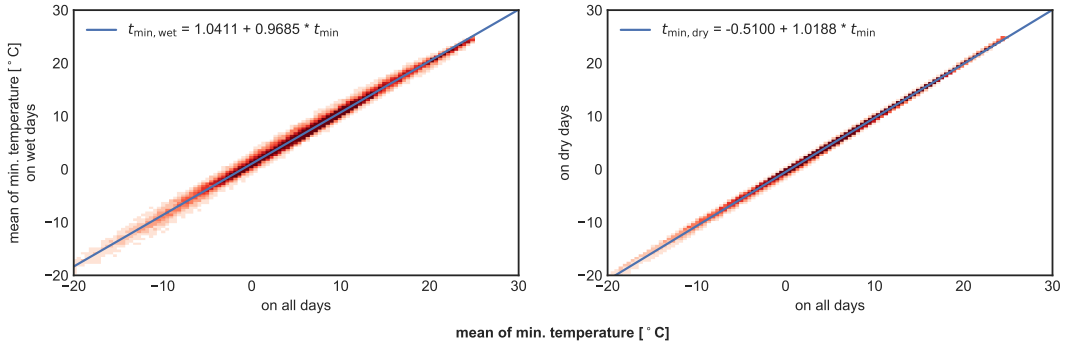

FIGURE 6.5: Correlation of minimum temperature on wet and dry days to the monthly mean. The y-axes show the mean minimum temperature on wet or dry days respectively, the blue line corresponds to the best fit line. Parameters of the fits are also shown in table 6.1.

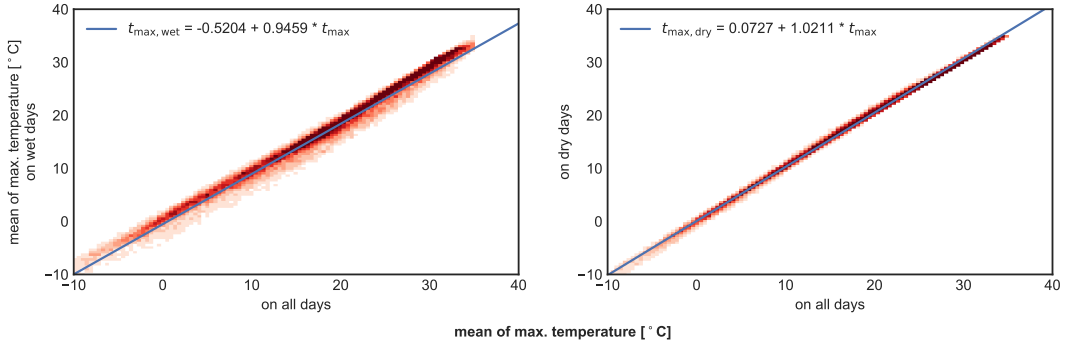

FIGURE 6.6: Correlation of maximum temperature on wet and dry days to the monthly mean. The y-axes show the mean maximum temperature on wet or dry days respectively, the blue line corresponds to the best fit line. Parameters of the fits are also shown in table 6.1.

As proposed by Geng et al., 1986, we use this relationship in our model to estimate the scale parameter of the distribution. Using this approach, the gamma shape parameter α is a constant, given via

$$\alpha = \frac{\bar{p}_d}{\theta} = \frac{1}{1.262}. \tag{6.8}$$

The GP scale parameter σ on the other hand is calculated during the simulation following Neykov et al., 2014 via

$$\sigma = \frac{1 - F(\mu)}{f(\mu)}. \tag{6.9}$$

The other parameters of the GP distribution are obtained through a sensitivity analysis described in section 6.3.5.

Temperature

Following the standard WGEN methodology (Richardson, 1981) and Geng et al., 1986, daily temperature is determined through 2 processes: First, the wet/dry state of the day, and second the cross correlation (section 6.3.2).

This better resolves the complex behavior close to $0\,\mathrm{m\,s^{-1}}$ compared to a linear fit. The plots are shown in the figures 6.10 and 6.11 and the parameters for the fits are shown in table 6.1.

Cross correlation

Following Richardson, 1981 we use cross correlation to add additional residual noise to the simulated meteorological variables, which provides more realism in the daily weather result. This ~~methodology~~method, based on Matalas, 1967 preserves the serial and the cross correlation between the simulated variables. It implies that the serial correlation of each variable may be described by a first-order linear autoregressive model

Given the cross correlation matrix $M_0 \in \mathbb{R}^4 \times \mathbb{R}^4$ and the lag-1 correlation matrix $M_1 \in \mathbb{R}^4 \times \mathbb{R}^4$, we calculate

$$A = M_1 M_0^{-1} \qquad BB^T = M_0 - M_1 M_0^{-1} M_1^T. \tag{6.15}$$

The matrices A, B, M_0 and M_1 are calculated using the stations from the EECRA database in figure 6.2. The results are

$$M_0 = \begin{pmatrix} 1. & 0.565 & 0.041 & 0.035 \\ 0.565 & 1. & -0.089 & -0.043 \\ 0.041 & -0.089 & 1. & 0.114 \\ 0.035 & -0.043 & 0.114 & 1. \end{pmatrix} \qquad M_1 = \begin{pmatrix} 0.933 & 0.55 & 0.016 & 0.03 \\ 0.557 & 0.417 & -0.066 & -0.043 \\ 0.004 & -0.095 & 0.599 & 0.093 \\ 0.011 & -0.063 & 0.061 & 0.672 \end{pmatrix}.$$

$$M_0 = \begin{pmatrix} 1. & 0.565 & 0.041 & 0.035 \\ 0.565 & 1. & -0.089 & -0.043 \\ 0.041 & -0.089 & 1. & 0.114 \\ 0.035 & -0.043 & 0.114 & 1. \end{pmatrix}$$

$$M_1 = \begin{pmatrix} 0.933 & 0.55 & 0.016 & 0.03 \\ 0.557 & 0.417 & -0.066 & -0.043 \\ 0.004 & -0.095 & 0.599 & 0.093 \\ 0.011 & -0.063 & 0.061 & 0.672 \end{pmatrix}. \tag{6.16}$$

leading to

$$A = \begin{pmatrix} 0.916 & 0.031 & -0.018 & 0.001 \\ 0.485 & 0.135 & -0.069 & -0.047 \\ 0.004 & -0.043 & 0.592 & 0.023 \\ 0.012 & -0.043 & -0.02 & 0.672 \end{pmatrix} \qquad B = \begin{pmatrix} 0.358 & 0. & 0. & 0. \\ 0.112 & 0.809 & 0. & 0. \\ 0.142 & -0.06 & 0.785 & 0. \\ 0.077 & -0.016 & 0.061 & 0.733 \end{pmatrix}. \tag{6.17}$$

The columns and rows in the two matrices correspond to min. and max. temperature, cloud fraction and square root of wind speed, respectively.

In the weather generator, the variables $T_{\min}, T_{\max}, c$ and w are then calculated using a combination of residual noise χ_i (where i denotes the current simulated day) and the mean of the variables. χ_i is determined by the other variables and the previous day using A and B from above (Matalas, 1967; Richardson, 1981). Hence, χ_i is

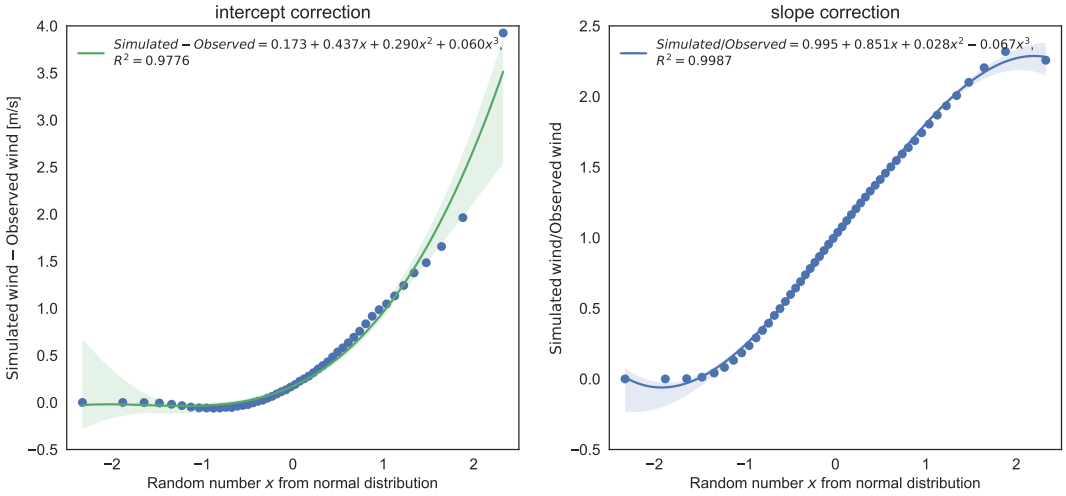

FIGURE 6.14: Basis for the wind bias correction. For the left plot, each data point corresponds to the difference of a simulated percentile to the observed percentile. For the right plot (wind speed), each data point corresponds to the fraction of simulated to the observed wind speed for a given percentile. The random number on the x-axis represents the residual value from a normal distribution centered at 0 with standard deviation of unity, as it is used in the cross correlation approach (Richardson, 1981).

6.3.4 Bias correction

After evaluating the results of GWGEN for wind speed for the different quantiles (see previous subsection 6.3.3) we found a strong, systematic bias between the simulated and the observed values. This observation led us to adopt a further measure to improve the quality of the model output by implementing a quantile-based bias correction.

We use an empirical distribution correction approach (quantile-mapping) (Lafon et al., 2012) to a posteriori correct the simulated data. In the quantile evaluation (previous subsection 6.3.3) we saw that the simulated wind speed is a linear function of the observed wind speed, i.e. $w_{sim} = \text{intercept} + \text{slope} \cdot w_{obs}$ (best fit line in figure 6.12). Therefore, we use two steps here, one is for the difference between simulation and observation (ideally 0), the other one is the fraction of observation and simulation (ideally 1). The first one corresponds to the intercept with the y-axis in figure 6.12, the second one to the slope of the best fit line. The analysis is based on every second percentile between 1 and 100 (i.e. $1, 3, 5, \ldots$) and mapped to ~~it's~~ its corresponding random number $u \in \mathbb{R}$ from a normal distribution as it is used for the cross correlation in the weather generator (section 6.3.2, x-axis in figure 6.14 and Richardson, 1981).

Regarding the intercept (fig. 6.14, left) we see that it strongly follows an exponential function given through

$$f_{exp}(u) = \mathrm{e}^{a\,u+b}, \qquad a, b, u \in \mathbb{R}. \tag{6.21}$$

The slope (fig. 6.14, right) on the other hand can be described by a simple third-order polynomial given by

$$p3(u) = c_0 + c_1\,u + c_2\,u^2 + c_3\,u^3, \qquad c_0, c_1, c_2, c3, u \in \mathbb{R} \tag{6.22}$$

Hence, given the best fit lines in figure 6.14, the simulated wind speed is corrected via

$$w'_{sim} = \frac{w_{sim} - f_{exp}(u)}{p3(u)} \tag{6.23}$$

with $a = 1.1582, b = -1.3359, c_0 = 0.9954, c_1 = 0.8508, c_2 = 0.0278, c_3 = -0.0671$.

6.3.5 Sensitivity analysis

The Generalized-Pareto part of the hybrid Gamma-GP distribution, which we used to simulate precipitation amount, has two parameters: the GP shape, and the threshold parameter. Unlike the gamma parameters, we were unable to relate these GP parameters to any of the monthly summary data we use as input to GWGEN. Hence, we decided to set fixed values for these parameters, and determine them through a sensitivity analysis.

To select the "best" values of the GP parameters, we compared simulated with observed precipitation amounts, running GWGEN with a wide range of realistic parameter values. To quantitatively assess the model performance, we used two metrics: 1) direct comparison of the quantiles (see previous section), and 2) a ~~Kolmogorov-Smirnov~~ Kolmogo-rov-Smirnov (KS) test that evaluates whether two data samples come from significantly different distributions. Our criteria were

1. The R^2 correlation coefficient between simulated and observed quantiles
2. The fraction $\frac{\text{simulated precipitation}}{\text{observed precipitation}}$ from the slopes in figure 6.13 and ~~it's~~ its deviation from unity
3. the fraction of simulated (station specific) years that are significantly different (KS test) from the observation
4. The mean of the above values

We tried two different approaches to select the gamma-GP crossover threshold: first we tried a fixed crossover point, second we used a quantile-based crossover point. For the latter, the model chooses to use the GP distribution if the quantile of the random number drawn from the gamma distribution is above a certain quantile threshold. This introduces a flexible crossover point in our hybrid distribution which, however, did not improve the results significantly. We therefore show here only the results using the fixed crossover point.

The values of the crossover point for our sensitivity analysis were 2, 2.5, 3, 4 and from 5 to 20 in steps of 2.5 and 20 to 100 in steps of 5. Furthermore we varied the GP shape parameter from 0.1 to 3 in steps of 0.1 (810 experiments in total). The results of this sensitivity analysis are shown in the supplementary material, figure 6.15.

In general we found that the three criteria 1, 2 and 3 could not be optimized all together at the same time. The R^2 is best for high thresholds and low GP shape parameters, the slope is best for low to intermediate thresholds and a low GP shape and the KS statistic is best for low threshold and intermediate GP shape parameters.

However, R^2 did not vary that much (from 0.68 to 0.74) and from a visual evaluation of the corresponding quantile plots we saw that the higher quantiles (>90) were much better represented for a better KS result. Hence, we chose to follow the KS test criteria, which is also the strictest of our evaluation methods but again compared the different quantile plots to get good results for the higher quantiles. Finally, we chose a threshold of 5 mm and a GP shape parameter of 1.5. For this setting, 81.7% of the

Chapter 7

Conclusions

Paleoclimatological large-scale reconstructions allow an independent evaluation of global climate models ~~and their skill~~ skill to simulate climates outside the range of modern climate variability. In the previous chapters of this thesis I described several new open-source tools that can be used to leverage the single site-based proxy-climate reconstruction onto ~~a~~ continental, or even global scale, by using a combination of thousands of different records.

For the EMPD (Davis et al., in prep) I developed a web-framework to communicate and manage the community-driven database in a transparent and sustainable way (chapter 2). The framework consists of an interactive web-based interface into the data and an automated administration webapp. The entire framework is based on the free webservices ~~that are~~ provided by the version control platform Github and as such allow to trace back every change to the database and provides a variety of tools to manage new contributions and/or changes to the database. This methodology assures stable and intuitive access to the database, independent of the available funding, and contributors or maintainers. One can think of many further potential applications for this framework that can be applied to any regional pollen (or in general, proxy) database. The EMPD is only one example, other potential use cases are the LAPD (Flantua et al., 2015), or the APD (Vincens et al., 2007). The method can also be applied to communicate a study-specific collection of proxy sites and use already implemented analysis and visualization tools for the data, or add new methods specific to the scope of the study. The future plans with this project therefore include a further generalization of the methods, particularly the visualization methods of the EMPD- and POLNET-viewer (section 2.3), to make it widely applicable. The integration with Github allows an easy way to share the source, and to host the interactive interface on the same platform without any costs.

The next tool I presented is the stratigraphic digitization software straditize in chapter 3 (Sommer et al., 2019). This package ~~transform~~ transforms stratigraphic diagrams, i.e. diagrams where the analysis of samples are plotted against a common y-axis, usually representing age or depth. The potential applications for this software are numerous because of the existence of hundreds of pollen datasets (and more) that are only available as pollen diagrams in the publications. This software provides the unique possibility to make this data from the pre-digital era accessible in a reasonable amount of time. Further extensions to this package will involve the support of new diagram types (e.g. multiple lines in a single diagram column). A strong focus will lie on the documentation of the software in order to make it easier and accessible. This will involve video tutorials, more tutorials for the various diagrams directly embedded into the software, and there are still some parts of the software that are not yet sufficiently document.

In chapter 4, I further described the interactive visualization framework psyplot (Sommer, 2017), a cross-platform open source python project that combines plotting

and data management into a single framework that can be used conveniently via command-line and a GUI. The software differs from most of the visual analytic software such that it focuses on extensibility and flexibility ~~that can~~ and can therefore be used in a variety of research questions. It particularly provides the basis for the straditize software (chapter 3) and for multiple of the analysis in the chapters 5 and 6. psyplot currently provide visualization methods that range from simple line plots, to density plots, regression analysis and geo-referenced visualization in two dimensions. In the future, this will be enhanced with 3D visualization methods to provide the first visualization tool in climate research that can be used conveniently from the command line (Sommer, 2019).

In the second part of my thesis I described two new computational models that leverage site-based information into models for large-scale prediction of paleo environments. The first one is the pyleogrid package, a new method for a spatio-temporal gridded climate estimate from a database of proxy-climate reconstruction. It is a probabilistic extension of the method by Mauri et al., 2015 and Davis et al., 2003 that provides reliable uncertainties by incorporating the intrinsic dating and proxy-climate reconstruction uncertainties of the individual sites, in order to generate a product that can ~~be conveniently~~ conveniently be used for data-model intercomparison. In addition, this framework contains two novel methods, (1) a model to predict dating uncertainties based on the age of the pollen sample and the time-difference to the closest chronological control point, and (2) a new probabilistic version of the MAT, based on constrained Gibbs sampling algorithms for the age of the samples and the climate reconstructions. The ensemble method is very scalable, both in terms of the size of the spatio-temporal domain, and the computational resources that are used for the prediction. This method will be used in the near future to generate a climate reconstruction for the entire northern hemisphere that is based on the POLNET database as described in Davis and Kaplan, 2017. Further developments will also concentrate on a revision of the age uncertainty estimate using the recent database by Wang et al., 2019 which contains standardized chronologies for more than 500 datasets.

Finally, the second model in chapter 6 describes the new global weather generature GWGEN (Sommer and Kaplan, 2017), a statistical model for a temporal downscaling of monthly to daily climatology. This model can be applied on the entire globe whilst being parameterized based on a large dataset of weather stations with more than 50 million records. The aim is to provide a tool that can be implemented in a global vegetational model for paleo environments. These models require daily meteorology as input which poses a considerable challenge considering the long simulation period they have.

I additionally developed another model in collaboration, that is not related to paleo and therefore not included in the main part of this thesis. This model, the Integrated Urban Complexity Model (IUCM) (Cremades and Sommer, 2019), presents a new method that simulates the transformation of urban areas while focusing on a low energy consumption from urban mobility. I mention this model here because it also incorporates the infrastructural methodologies that I developed for psyplot and GWGEN. This additional use-case highlights one of the important aspects of open-source software development, that is a flexible, extensible and sustainable modular framework, where the packages related to a specific product can be used in multiple other products. Other examples for it are the *docrep* and *sphinx-nbexamples* packages (Sommer, 2018a,b) that I primarily developed for psyplot but are used in a variety of recently developed packages now (e.g. Abernathey et al., 2017; Banihirwe et al., 2019; Uchida, 2018).

List of Figures