universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

„A Comparative Analysis of Region Pairs Matching Current and Future Climate Conditions“

Verfasser

Joachim Ungar

angestrebter akademischer Grad

Magister der Naturwissenschaften (Mag. rer. nat.)

Wien, 2011

Studienkennzahl lt. Studienblatt:	A 455
Studienrichtung lt. Studienblatt:	Kartographie und Geoinformation
Betreuerin / Betreuer:	Univ.-Prof., DI Dr. Wolfgang Kainz

Abstract

This thesis is about mapping climate change in a novel way. Climate models simulate the complex energy and matter fluxes of the climate system within an uncertainty range and produce a huge amount of data with a very high temporal resolution, which are used to derive integrated indicators like temperature means, precipitation totals for a certain time range - a year, a season valid for a certain area. Climate change is indicated through differences between the indicators for time ranges presenting current climate and a future climate scenario. But these integrated indicators do not provide a complete overview of the climate changes and the impacts which will be expected.

To experience the expected changes in a more "tangible" way, the Austrian Institute of Technology (AIT) has developed the Climate Twin application (Loibl et al. 2010). By selecting a certain location ("source region") in an interactive map an algorithm is initialized which compares a set of climate indicators of current climate and future climate and generates a second map showing the matching Climate Twin areas ("target regions") according to their current climate conditions compared to the future climate conditions in the source region.

The main objective of this work is to elaborate a suitable matching method to better identify Climate Twin regions. By using mean values in the description of a whole year's climate conditions, some unfavorable simplifications occur. A mean value by itself does not include other main properties like temperature amplitudes with its peaks and sinks or extreme precipitation events. A suitable similarity measure therefore should also contain basic distribution properties like range, skewness, bipolarity, etc. A comparative analysis of similarity measures for distributions was done by Jan Vegelius et al. (1986) in which two of the reviewed measures, the *Proportional Similarity* (PD, formula 2.1) and the *Hellinger Coefficient* (r_{H}, formula 2.2), were tested to have the most suitable properties.

The approach of comparing the statistical distributions of climate indicators with the methods mentioned above starts with an exploration of the main parameters by analyzing their properties in sample locations (Vienna, Copenhagen, Munich and Rome) in different climate regions. The main questions among others are the selection of suitable climate indicators, their applicability for this approach, the bandwidths of similarity (uncertainty ranges) and the combination and weighting of the similarity (r) indicators to achieve appropriate Climate Twin results.

Zusammenfassung

Diese Arbeit beschäftigt sich mit einem neuen Ansatz, Klimaänderungen interaktiv kartographisch darzustellen. In der Regel werden Klimaänderungen mittels der Darstellung von Veränderungen der Durchschnittstemperaturen zwischen zwei Zeitpunkten kommuniziert, womit mögliche Erwärmungen und Abkühlungen im langjährigen Mittel offenbart werden. Methodisch ist dieser Ansatz zwar korrekt, jedoch erschwert er die Interpretation der tatsächlichen Bedeutung und der Effekte, welche Klimaänderungen mit sich bringen und zwar sowohl für Laien als auch Experten.

Klimamodelle simulieren die komplizierten physikalischen Zusammenhänge innerhalb der Atmosphäre und mit der Erdoberfläche. Sie produzieren somit eine Fülle zeitlich hochaufgelöster Daten, mit welchen integrierte Indikatoren wie Temperaturmittelwerte und Niederschlagssummen für eine bestimtme Zeitspanne (Jahr, Jahreszeit, Monat) für eine bestimmte Fläche berechnet werden. Effekte des Klimawandels werden dann mittels der Differenzen der Indikatoren die Zeitspannen des aktuellen und eines künftigen Klimas beschreiben, quantifiziert. Doch diese integrierten Indikatoren liefern keinen gesamten Überblick über den zu erwartenden Klimawandel und die sich daraus ergebenden regoinalen Effekte. Aus diesem Grund hat das Austrian Institute of Technology (AIT) die „Climate Twins Applikation“ entwickelt (Loibl et al. 2010), in welcher es dem Nutzer ermöglicht wird, Regionen zu identifizieren, welche jetzt bereits ein ähnliches Klima aufweisen, wie ein Ort in einem zukünftigen Zeitraum.

Das vorrangige Ziel dieser Arbeit ist es also, eine geeignete Methode zu entwickeln, um aus vorhandenen Datensätzen Regionen mit ähnlichen klimatischen Eigenschaften auszumachen. Normalerweise werden Klimata anhand von Mittelwerten (Temperatur) oder Summenwerten (Niederschlag) über größere Zeiträume gebildet. Ein Mittelwert beinhaltet jedoch keine Informationen über die Amplituden von Temperaturkurven oder über Extremwerten von Niederschlagsereignissen. In ein geeignetes Ähnlichkeitsmaß sollten also die wesentlichen Eigenschaften einer statistischen Verteilung wie deren Varianz, Schiefe, Krümmung oder Bipolarität einfließen. Eine Evaluierung gegebener Ähnlichkeitsmaße für Verteilungen wurde von Jan Vegelius et al. (1986) durchgeführt, bei der sich zwei Maße, die *Proportional Similarity* (PD, Formel 2.1) und der *Hellinger Koeffizient* (r_{H}, Formel 2.2) herauskristallisierten, welche die besten Möglichkeiten für derartige Anwendungen bieten.

Die Verwendung derartiger Ähnlichkeitsmaße für Klimadaten erfordert deren Prüfung anhand von Testdatensätzen. Dafür wurden die Städte Wien, München, Kopenhagen und Rom herangezogen. Wichtige weitere Schritte sind die Auswahl geeigneter Klimaindikatoren, deren Anwendbarkeit auf die Ähnlichkeitsmaße, die (Unsicherheits-)Bandbreiten der Ähnlichkeit, sowie die Kombination und Gewichtung der einzelnen Ähnlichkeitswerte um aussagekräftige Ergebnisse, nicht nur anhand weniger integrierter Indikatoren sondern anhand des Vergleichs der gesamten Verteilung der Temperatur- und Niederschlagsdaten über eine aktuelle und einer künftige Zeitspanne, zu produzieren.

Acknowledgements

First of all, I would like to thank my mentors, especially Wolfgang Loibl for providing me with ideas, profound knowledge and permanent, sometimes time-consuming support during the last year. I am grateful for having dived with his and my colleagues'—namely Jan Peters-Anders and Johann Züger—help into the fascinating but rather back-breaking world of data preparation. Furthermore I would like to show my gratitude to all the other people sharing time with me in the AIT, especially Christoph Aubrecht and Klaus Steinnocher, guiding me through my first year in the science and research area.

For reviewing the master thesis, criticizing and scrutinizing I owe my deepest gratitudes to Norbert, Frederik and Hannes as well as Bob and Bert for their courageous performance.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project, mainly my parents who made my studies possible, my brothers and sisters including their families, and my dearest friends.

Contents

List of Figures

List of Tables

Abbreviations

r value	value for similarity
PD	Proportional Similarity
r_{H}	Hellinger Coefficient
CLM	Climate Local Model
COSMO-CLM	COnsortium for Small-scale MOdelling - Climate Local Model
OLAP	OnLine Analytical Processing
pdf	probability density function
SOLAP	Spatial OnLine Analytical Processing

1 Introduction

Although terms like "global warming" or "anthropogenic forced warming" lead to heated debates between "the scientific consensus" and global warming "deniers", a change in the main climatological variables during time is being accepted by a vast majority of both researchers and the public. The climate system is highly dynamic, complex and has changed steadily over the known history. Therefore it is feasible to assume that change whatsoever is to be expected again in the future. Anticipating the forms of changes is important to estimate the possible future state of the environment and to evaluate which kinds of new challenges they bear.

The evaluation of climate change and developing adaption strategies is related to climate model results, which simulate the climate-relevant physical processes and produce a vast amount of quantitative data. To gain thorough understanding of the implications of expected upcoming climate changes, ways have to be found to communicate the model results in a more "tangible" way. The Climate Twins application, developed by the AIT, is an attempt to translate the climate model results into easily understandable information. A user who is interested in the future climate conditions of his hometown selects a certain location in an intuitively usable web application and a map shows regions in Europe which now have similar climate conditions as the future climate conditions of the selected "source" location. In this way the scientific output is directly related to real world conditions and is therefore more easily understandable.

Climate can be seen as the amount of statistical distributions of various climate indicators like temperature, precipitation, air humidity and many others through space and time. Talking of climate within the context of quantitative data analysis requires the careful selection of the right climate indicators and their statistical parameters representing "climate".

In the current Climate Twins version, climate is represented by monthly mean temperatures and precipitation sums. The matching algorithm compares the future climate of the source location with the current climates of every region in Europe month by month. A region—or raster cell given by the climate model—is identified as Climate Twin if the deviations of the monthly indicator values lie within a given threshold for 6 to 8 (low similarity), 9 to 10 (high similarity) or 11 to 12 (very high similarity) months of the year for each climate indicator respectively.

This matching method is a first approximation of evaluating climate similarity and has a few drawbacks. In fact, the accuracy and applicability of the similarity identification strongly depends on the selection of the climate indicators and the similarity thresholds or uncertainty ranges. Too few climate indicators and too wide uncertainty ranges will identify too many and too large Climate Twin regions, whereas too many indicators and too narrow uncertainty ranges will identify little or no Climate Twin regions. Using monthly mean values also leads to problems as it does not incorporate the variability, peaks or range of the indicator's distribution which could also be interesting properties.

The problem on seeking regions with similar climates for the Climate Twins Application leads to four basic research questions:

1. Which climate indicators could be used according applicability and availability?

2. Which possibilities exist to compare two regions and their climate indicators?

3. Which is a suitable uncertainty range or threshold to define similarity?

4. Which is the best way to measure similarity for this application?

The first question addresses the problem described above. The climate indicators chosen have to "describe" the climate in an appropriate and feasible way for the Climate Twins application. The second question is about the methods being used to describe similarity between the climate indicators proved to be sufficient by question 1 and is to be answered with statistical methods. As similarity always depends on subjective decisions, the (third) question arises on how to quantify and define thresholds where a "common sense" of similarity is considered. The fourth question is about combining the findings into an applicable method including the logic behind the query or the aggregation and weighting of the indicators.

The main objective of the thesis is to develop a working matching method ready to be implemented into a working Climate Twins prototype. According to the research questions given, some basic objectives have to be accomplished. First, meaningful and applicable climate indicators have to be selected. Established types of climate classification give useful inputs in selecting these indicators. To quantify similarities between two data vectors a statistical function has to be chosen which determines their relative (dis)similarity in a normalized—"unit" free—form to facilitate the combination of different indicator similarity values. A further objective is the determination of applicable thresholds and matching conditions including weighting and combination of the indicator similarity values.

For declaring similarity two measures are tested and evaluated. Both the Proportional Similarity (PD) and the Hellinger Coefficient (r_{H}) compute a similarity value between 0 (no similarity) and 1 (identical) between two statistical distributions. Based on these measures a computation process was designed and implemented into the Climate Twins Application. Because of the nature of the available test data—COSMO-CLM modeled climate data—and its raster structure the matching algorithm is applied between the corresponding source location cell with every other cell in a sequential way.

To evaluate the method, first tests were done using modeled climate data of the four sample locations Vienna, Copenhagen, Munich and Rome. In a second step a working prototype to identify similarities was developed using the current basic Climate Twins application's architecture. Therefore, the climate data had to be restructured in the data base and a new version of the matching algorithm had to be written in Java.

This thesis is structured into five chapters. The first—this—Chapter should describe the basic idea behind the Climate Twins and formulate the research questions which had been the driving forces during the last year. The second Chapter shall give an overview of the theoretical basics needed to approach the topic. It is divided into two sections, one providing definitions and ideas about climate focusing on classification, modeling and indicators and the other giving short insights on approaches of measuring similarities with statistical methods. In this chapter, also the subjects

of the first two research questions are discussed. The third Chapter is about calibrating the similarity measures and exploring their potential by testing them on modeled climate data from the cities of Vienna, Copenhagen, Munich and Rome. Furthermore the whole logical structure of the Climate Twins matching method is elaborated. In the fourth Chapter, the existing Climate Twins tool is described focusing on the technological infrastructure and functionality leading to the changes and improvements which had to be done to implement the method worked out in the prior chapters. Chapter five presents result maps, their interpretations and discusses the problems, strengths and weaknesses of the method and discusses the subjects of the last two research questions before giving a conclusion and proposing further improvements.

In addition all of the produced and used scripts including R-scripts for preparing, analyzing and rendering and the Java program implemented into the web application are to be found in the appendices. This should make the research process and its milestones transparent as the reader is enabled to retrace the single steps of this work. The programming work is meant to be open source so if there are any questions or improvements on the code, feel free to contact the author via joachim.ungar@gmail.com.

2 Theoretical framework

2.1 Climate

The term *climate* originates from the ancient Greeks and means *inclination* which refers to the angle the sunbeams hit the earth's surface. The Greeks recognized this inclination as one of the most important drivers for temperature, wind and precipitation patterns varying between the seasons. On a global scale this definition implies a meridional classification of climate types as the sunbeams have a stronger impact around the equator than towards the poles. On a first glimpse this may be correct but there are many other influences which result in climate variations between the western and eastern edges of continents, between maritime and continental zones or between highlands and lowlands. (Malberg 2002, p. 271ff.)

In the last centuries the definitions got a little bit more deliberate:

The German climatologist Köppen pointed out in 1923 that *"climate is the mean condition and ordinarily progress of the weather at a place."* (Malberg 2002, p. 272).

Malberg himself gives a more statistical definition of climate as *"the whole atmospheric conditions and processes defined by the means, the variations and the extreme values within an adequate period of time."* (Malberg 2002, p. 272)

Oliver further argues, that *"climate fluctuates on all time scales: monthly, yearly, decadally, centennially, and millennially. Thus, climate is a statistical collective. It has often been described in terms of mean values of particular climatic elements, but it encompasses a wide range of values, including occasional extremes."* (Oliver 2005, p. 272)

These statements imply that climate is a complex statistical term which strongly depends in both the time and the space scale given. In fact there are many spatial scales from where climate can be described like microclimate, local climate, mesoclimate and macroclimate. Furthermore, an atmospheric condition (which leads to the term climate) is described by a composition of many climate variables like temperature, precipitation, humidity, wind speed and others. Therefore it depends on the context in which the term climate is used as it can describe e.g. the urban climate in the summer of 2009 in Vienna or the global climate within the next twenty years.

2.1.1 Climate classification

Classification is about aggregating entities with similar characteristics described by attributes. As the term *climate* is not only defined within a highly varying temporal and spatial scale, it is also defined by a wide range of physical parameters. Some parameters are easy to measure, most of them are not. Some parameters have been considered more important in characterizing a certain region by it's climate conditions than others. Therefore, there have been many attempts to use a certain combination of parameters to characterize the climate conditions in a region.

The most obvious variable is the sun. The Greeks combined their knowledge about a spherical earth and the earth traveling through seasons, to postulate a five-zone-classification of the earth, and another classification depending on the day length. In the early 19^{th} century there were more climatic data available, so more classifications emerged depending mainly on temperature and precipitation or a combination of various variables. The main idea behind the combination of variables was to describe

the distribution of vegetation. (Oliver 2005, p. 218ff)

Besides the "mathematical" (Malberg 2002, p. 274ff) classification according to the sunbeam angles done by the Greeks, other methods arose in the last two centuries due to better data availability. The availability of more climate indicators lead to possibilities classifying by certain thresholds and the more sophisticated method of using indices.

Gaile and Willmott (1984) worked out some objectives of climate classification. The most important objective is to simplify the complexity of the climate system. Therefore it provides an intellectual shorthand where huge amounts of information can be concentrated to few simple labels. Furthermore elaborating boundaries of various climate types helps to understand the underlying physical processes and their spatial distribution. Last but not least the knowledge of the spatial distribution of climatic similarity helps to avoid expensive redundancies when building up a climatic data collection network. (Gaile and Willmott 1984, p. 82ff)

2.1.1.1 The Köppen system

One of the most important works in this area still is the classification of climate by Wladimir Köppen. He defined five main climate zones with up to three sub zones depending on temperature and precipitation. The system is based on a combination of average, minimum and maximum values and their range. The Köppen classification was steadily improved by himself and others and became so dominant that nearly no other vegetation-related systems gained recognition. (Oliver 2005, p.220ff)

Köppen used a combination of upper-case letters A (tropical) to E (polar) to name the main climate zones and lower-case letters to add some basic hydrological or thermal characteristics. A C_f climate for example has an average temperature of below 18°C and above -3°C in the coolest and an average temperature of above 10°C in the warmest month indicated by the letter C. The optional letter f means that there is at least 3 cm precipitation every month in this climate zone (Oliver 2005, p. 220). With this system it is possible to characterize many different climate types in a very structured manner.

Recently attempts are observed to rebuild the climate classification maps by using gridded climate model data by Kottek et al. (2006) and using the automatic classification system to represent climate shifts in classification maps of various periods by Rubel and Kottek (2010).

2.1.1.2 Other classic systems

Another approach was published by Thornthwaite (1948). He focused on evapotranspiration, which is the water loss by transpiration through the vegetation and evaporation from the surface. Evapotranspiration is the reverse process of precipitation and therefore the mechanism that transfers water back into the atmosphere. Thornthwaite argued that one cannot know whether the climate is moist or dry if we had no idea of the evapotranspiration rate. (Oliver 2005, p. 223ff)

Approaches to classify climates not by their effects but by their causes arose in the middle of the 20$^{\text{th}}$ century. Before, these approaches were just approximations to

the complexity of the climate system. The most popular work was done by Strahler (1951) where he differentiated three main groups:

1. *"Group I. Climates dominated by equatorial and tropical airmasses all the year.*
2. *Group II. Climates that occur between groups I and III and that are influenced by the interaction between tropical airmasses (group I) and polar airmasses (group III).*
3. *Group III. Climates controlled by polar airmasses."*

(Oliver 2005, p.224)

2.1.1.3 Numerical classification

Early classification systems were later criticized because of several fundamental disadvantages. Willmott (found in Gaile and Willmott (1984, p. 81)) discussed in 1977 the huge influences arbitrary decisions have on classification regarding

1. the number of regions,
2. the criteria used to delineate between climatic types,
3. the variables chosen to characterize climate, and
4. strategies used to develop indices out of the selected variables.

Numerical classification is a systematical approach where classification is defined by rules and done by statistical methods or certain threshold values. For example the Köppen system can also be seen as a numerical classification scheme because he defined some rules and thresholds which have to be applied. The most popular statistical methods according to Gaile and Willmott (1984) are the Principal Component Transformation, correlation coefficients and Euclidean distance measures.

However, Rohli and Vega (2007, p. 187) mentioned that no mathematician has found a method to combine atmospheric data, spatial variables and temporal variables so that all variables can be analyzed simultaneously.

2.1.2 Climate data, variables and indicators

Climate is a spatio-temporal process where the condition can be determined by splitting it into various *climate elements* which are influenced by certain *climate factors*. Climate elements are therefore spatio-temporal *variables* presented by *climate data* and can be determined by measurement, estimations (if no measurement is possible) or observation of the atmosphere (e.g. a thunderstorm). Furthermore there are various variables like the wet bulb temperature combined by the climate variables air temperature, air humidity and wind. Last but not least there are *climate parameters* or *indicators* which are mathematically or statistically combined climate elements,

e.g. the annual mean temperature or the monthly precipitation sum. (Schönwiese 2008, p. 65ff)

Some of the major climate variables are

- air temperature
- air pressure
- wind
- air humidity
- cloud cover
- precipitation
- sunshine duration

(Schönwiese 2008, p. 67ff)

2.1.3 Modeling

The climate system which has to be modeled contains an innumerable amount of system components and processes. It was defined by the World Meteorological Organization (WMO) in 1975 as the *"composition of the atmosphere, hydrosphere, cryosphere, land surface and the biosphere"* (Henderson-Sellers and McGuffie 1988, p. 4). These layers interact and exchange energy and matter, mainly water. A climate model represents the most important and influential components and processes to simulate the whole system.

A model is always a purpose-related simplification of the real world. Therefore all the results of a model have an inherent uncertainty. In climate models the main sources of uncertainty are that not all atmospheric processes are fully understood and that the observational data the models are calibrated with, are not complete and sometimes not accurate enough (see Section 2.2.4). The simplifications that have to be made can be distinguished into two sets. (1) Not all of the processes can be modeled in detail, some have to be treated in an approximate way. The main two reasons are because of our lack of understanding and the limited computer resources. (2) The limits of the resolution of the model in both space and time have a direct influence in the reliability of the results. On the other hand there are constraints in the computability and data availability as the resolution increases. Furthermore not all the modeled processes are acting the exact same way in a more detailed spatial resolution than they were designed to. (Henderson-Sellers and McGuffie 1988, p. 35)

Besides other types of climate models the GCM (either for *general circulation model* or *global climate model*) is the most complete description of the climate system as it is capable to simulate the exchange of energy and mass in all three dimensions. There are four fundamental equations solved in a GCM (Henderson-Sellers and McGuffie 1988, p. 35):

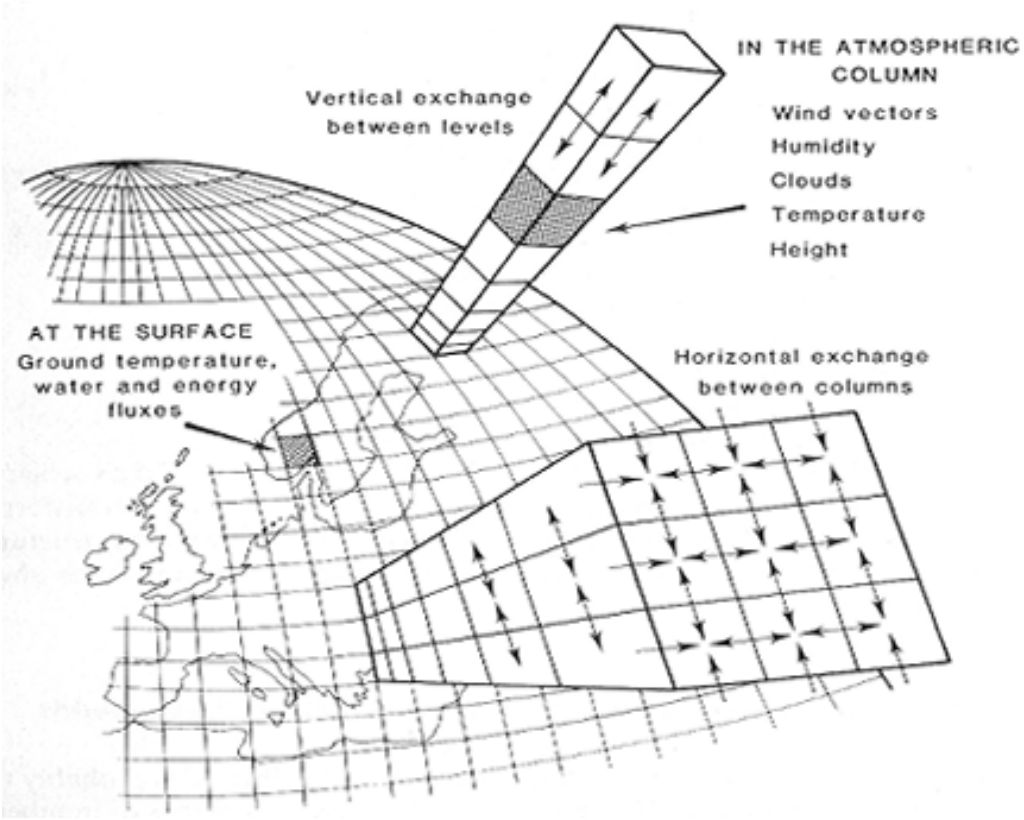


Figure 2.1: Basic structure of a cartesian grid GCM (Henderson-Sellers and McGuffie 1988, p. 138)

1. Conservation of energy: the first law of thermodynamics
2. Conservation of momentum: Newton's second law of motion
3. Conservation of mass: the continuity equation
4. Ideal gas law: an approximation to the equation of state

2.1.3.1 Cartesian grid GCM

In a cartesian grid GCM the atmospheric condition is calculated for points located on a grid. The grid includes a couple of layers or levels representing the vertical structure of the earth from the atmosphere to the deeper layers of the ocean. This structure allows to calculate both the horizontal exchange between the grid cells and the vertical exchange between the levels (Figure 2.1). At every given time step, which could be seconds to minutes, the basic atmospheric variables of every grid point is being calculated. These calculations are complex and intense for every computer system. Therefore many compromises have to be made in the spatial and temporal resolution, depending on the facilities available. (Henderson-Sellers and McGuffie 1988, p. 41)

2.1.3.2 Spectral GCM

Spectral GCMs represent the atmospheric fields not in a grid but in waves. These waves are also just an approximation of the real atmospheric states but as they are combination of sine and cosine waves it takes less computing resources than the grid approach. However, the model's surface remains as a grid and the vertical exchange transfers are also modeled in a rectangular grid. (Henderson-Sellers and McGuffie 1988, p. 140ff.)

2.1.3.3 Regional Climate Model (RCM)

GCM resolutions with grid spacing of around 100 km are relatively coarse so that local topographical effects, water bodies or regionally important circulations are unconsidered. Therefore, Regional Climate Models (RCM) simulate the atmospheric conditions in a better resolution (usually around 10 to 50 km) while receiving input data at the sub-domain's boundaries from the GCM. Regional models simulate smaller processes more accurately and produce therefore more realistic results. (India and Bonillo 2001, p. 454 and Barry and Chorley 1992, p. 168)

2.2 Statistics of time series data and spatial data

As we saw in Section 2.1, climate can be seen as a statistical collective of various climate variables. These variables are either measured or modeled in periodical time steps and therefore can be seen as a list of values. The aim of this chapter is to describe the possibilities to compare these data sets and to compute the similarity between them.

2.2.1 Measures describing datasets

A statistical distribution can be described by three main attributes: the dispersion, skewness and kurtosis. These attributes can be measured, especially for measuring the dispersion there are several methods like computing the variance, standard deviation, range or the Gini coefficient. Measuring skewness and kurtosis is much more a challenge and unfortunately the results are not always satisfactory. For example different skewness measurements of the same distribution could produce contradictory results. As climate data is rather not normally distributed and can have multiple peaks it leads to major problems in describing the distributions just by the dispersion, skewness and kurtosis with conventional methods. Furthermore the attributes have to be combined in a single indicator which would also lead to problems in weighting them in an appropriate manner.

2.2.2 Similarity conditions, indicators

Similarity is an often used, basic and intuitive concept which is hard to define and to measure. Vosniadou and Ortony differentiate between *literal similarity* and *analogy*. Literal similarity includes identical both relational properties and object attributes

of a system, an analogy only the relational properties (Vosniadou and Ortony 1989, p. 206). A more formal approach comes from Lin, where the following intuitions are stated (Lin 1998):

1. The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are.

2. The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are.

3. The maximum similarity between A and B is reached when A and B are identical, no matter how much commonality they share.

The most obvious and most used way to quantify similarity is to make choices how to measure similarity and then to define a certain threshold that divides the areas of similarity and non-similarity. Once the attributes of two or more entities are brought into a metric scale, it would be possible to measure the similarity by the distance lying between the attribute values. However there are always inherent problems caused by the subjectiveness of the attribute's definition, similarity threshold's definition and with generic basic problems of measuring anything. These uncertainties (see chapter 2.2.4) of the data therefore lead to an uncertainty of the similarity measure. All in all there may be three main challenges in declaring similarity.

1. The choice which attributes are defined and measured is a subjective process and can lead to different results.

2. The definition of the similarity threshold is subjective and therefore "arbitrary".

3. The data used, especially climate data as described here, vary within an uncertainty range and do not represent the "true" values.

2.2.3 Appropriate measures for data set comparison: similarity

Usually, statistical tests are used to confirm or reject a hypothesis. Therefore, some of them compare two distributions or samples of distributions which could help for a certain research question. Unfortunately the most used tests are not dedicated to prove or quantify similarity and before making any choices towards one or more tests a few basic requirements have to be defined.

The test should ...

1. not require a normal distribution,

2. include the basic properties of the distributions like the ranges, mean values, skewness etc.,

3. and deliver a standardized value between 0 and 1 as a result.

2.2.3.1 Descriptive statistic tests

A glimpse on the main tests in literature about descriptive statistics reveals the following problems:

t-test for computing the difference of two mean values. This test requires a normal distribution and a stochastic independence (Güßefeldt 1999, p. 206) and is therefore not suitable.

Confidence intervals of the two distribution's mean values. Here, just the quality of the mean values is being tested, not the rest of the distribution's properties. (Güßefeldt 1999, p. 203)

F-test for computing the difference of two dispersions (Güßefeldt 1999, p. 205). The F-test also compares just one property and is also not suitable.

Kolmogorov-Smirnov (KS) test . This test does not require a normal distribution. It compares the cumulated frequencies of two distributions and checks whether the maximum distance between them exceeds a certain value, the p-value (Güßefeldt 1999, p. 210). This test would satisfy the first two requirements but it is quite imprecise as it delivers only a binary result (yes or no) and does not quantify the amount of similarity.

Coefficient of determination, R^2 . The R^2 describes the goodness of a regression model between two or more variables. If there is a perfect linear relationship between two variables the R^2 has a value of 1, if there is little or no linear relationship the value goes towards 0. This method satisfies the first and third requirements but it fails on the second requirement as it does not incorporate the absolute values which means that if e.g. the temperature in region A is constantly 5°C higher than in region B, the R^2 would be 1 and therefore indicate a perfect similarity. (Hutcheson and Sofroniou 1999, p. 65)

None of the tests satisfies all of the requirements. The KS test seems to fit but it would fail in an extreme situation where two distributions are identical except for one extreme difference between two values. Here the extreme difference would exceed the p-value and the distributions would be marked as not similar.

2.2.3.2 Similarity measures

A comparative study on similarity measures of distributions was done by Jan Vegelius et al. (1986). They defined relevant criteria a similarity measure has to provide, whereas U, V are two distributions and r similarity measure:

1. $|r(U,V)| \leq 1$

   The result of r has to be a value between 0 and 1.

2. $r(U,U) = 1$

   r of two identical distributions has to be 1.

3. $r(U,V) = r(V,U)$

   the similarity measure has to lead to the same result in both directions.

4. If $r(U,V) = 1$ and $r(U,W) = 1$ then $r(V,W) = 1$ must also be equal to 1.

5. A correlation matrix based on r is positive semidefinite. Every value has to be greater than or equal to 0.

6. r is an E-Coefficient.

7. r has minimum, if and only if, $\sum_{i=1}^{C} f_{\mathrm{Ui}} * f_{\mathrm{Vi}} = 0$

8. The minimum value of r is 0.

9. If a category is divided into two, in such a way that the frequencies in these two new categories are equal to each to each other (for both distributions separately), then r should not be changed.

10. If a category is empty in both compared distributions, it may be deleted without affecting the value of r.

11. $r(U,V) = 1$, if, and only if, $f_{\mathrm{Ui}} = f_{\mathrm{Vi}}$ for each i.

Vegelius analyses the similarity measures and finds out that two measures fit to all eleven criteria. These two are the *Proportional Similarity* (PD, 2.1) and the *Hellinger Coefficient* (r_{H}, 2.2).

$$PD(U,V) = 1 - \frac{\sum_{i=1}^{C} |f_{\mathrm{Ui}} - f_{\mathrm{Vi}}|}{2} = \sum_{i=1}^{C} min(f_{\mathrm{Ui}}, f_{\mathrm{Vi}}) \tag{2.1}$$

$$r_{\mathrm{H}}(U,V) = \sum_{i=1}^{C} \sqrt{f_{\mathrm{Ui}} * f_{\mathrm{Vi}}} \tag{2.2}$$

Both measures work quite similar. Both of them calculate with the relative frequencies of predefined categories in the two distributions. The PD summarizes the smaller relative frequencies of each category, the r_{H} summarizes the square root of the both relative frequencies products per category. Vegelius advises to use the PD rather than the r_{H} because it is easier to understand and PD values are smaller than r_{H} values (Vegelius et al. 1986).

2.2.4 Uncertainty range

The term *uncertainty* is used in various different contexts. Very often it is used in the context of measuring, where there are two different types of uncertainties. Measurements never represent "true" values but only approximations where the difference between the true value and the approximation is called *accuracy*. Depending on the method of measuring, the measured values may show a slight variation which is called *precision*.

Douglas Hubbard (Hubbard 2007, S. 46) gives a viable definition of uncertainty:

> *"Uncertainty: The lack of complete certainty, that is, the existence of more than one possibility. the 'true' outcome/state/result/value is not known.*
>
> *Measurement of Uncertainty: A set of probabilities assigned to a set of possibilities. ..."*

Manfred Drosg, a physicist in Vienna, states that uncertainties are not only the fault of the measurement but the *"trademark of science"* (Drosg 2009, p. 1). As models or theories have to be used in science to approach reality but there will never be a model or a theory representing reality in all its complexity. Drosg cites Nobel Price laureate Richard P. Feynman who said *"Scientific knowledge is a body of statements of varying degree of certainty - some most unsure, some nearly sure, but none absolutely certain"* (Feynman et al. 1997). A famous example of uncertainty as an integral part of reality is the Uncertainty Principle stated by Werner Heisenberg in his work on Quantum Mechanics in 1927 (found in Heisenberg (1969)).

Climate models therefore are also to be seen in the context Drosg mentioned above. The IPCC therefore distinguishes between three simple types of uncertainties (Table 2.1). As a climate model strongly depends on input parameters and processes producing values within an uncertainty range, the results are also computed within an uncertainty range. The second part of Hubbard's definition above reveals an approach to quantify and deal with these uncertainties, named probability density functions (pdf). Various research groups attempt to evaluate and quantify climate model uncertainty (e.g. Andronova and Schlesinger (2001), Forest et al. (2002)).

In the context of this work, uncertainty is used as the range within similarity can be stated. A region is similar to another according to some selected indicators as long as its value lies within a given uncertainty range. For the Climate Twins idea it means that every region or every raster cell has a grade of similarity characterized by a similarity measure. The results are therefore from a mathematical point of view not uncertain in the sense of probable but continuous instead of discrete.

Type	Indicative example of sources	Typical approaches or considerations
Unpredictability	Projections of human behavior not easily amenable to prediction (e.g. evolution of political systems). Chaotic components of complex systems.	Use of scenarios spanning a plausible range, clearly stating assumptions, limits considered, and subjective judgments. Ranges from ensembles of model runs.
Structural uncertainty	Inadequate models, incomplete or competing conceptual frameworks, lack of agreement on model structure, ambiguous system boundaries or definitions, significant processes or relationships wrongly specified or not considered.	Specify assumptions and system definitions clearly, compare models with observations for a range of conditions, assess maturity of the underlying science and degree to which understanding is based on fundamental concepts tested in other areas.
Value uncertainty	Missing, inaccurate or non-representative data, inappropriate spatial or temporal resolution, poorly known or changing model parameters.	Analysis of statistical properties of sets of values (observations, model ensemble results, etc); bootstrap and hierarchical statistical tests; comparison of models and observations.

Table 2.1: A short typology of uncertainties (Solomon et al. 2007)

3 Methods exploration, analysis & selection, operationalization

3.1 Exploration of approaches to define similarity measures

The challenge in using the similarity measures described by Vegelius et al. (1986) in Subsection 2.2.3.2 is to examine the best working parameters in terms of the amount and ranges of "categories" the distributions are to be split into. Too many categories lower the r-value, too few rise the r-value and both ways produce an imprecise result. For example, in an extreme case where just one category is defined, the r-value will always be equal to 1 and an extremely high number of categories, the r value would shrink towards 0. Another choice has to be made in defining the range, where the categories have to be built. In all cases the range has to include all possible values. For this purpose, where the similarities of many distributions are calculated and afterwards compared, this range could be static for every single similarity measurement (minimum and maximum value of the whole data set) or dynamic defining always different ranges for every single similarity measurement (minimum and maximum values of the two current distributions).

Another problem occurs because by comparing distributions the temporal information gets lost. Two regions with the same distribution of precipitation sums over the year but with one having the peak in spring and the other in autumn would be defined similar without coping with the problem. This problem could be solved by splitting the data into seasons and compare season by season.

Oliver (2005) defines season as a *"period of time during the calendar year characterized by or associated with a set of coherent climatic activities or weather phenomena."* (p. 651). Usually these four seasons are spring, summer, autumn and winter of three months each. On page 655, Oliver (2005) also presents other concepts of splitting the year into seasons but for this application the standard classification (spring: MAM, summer: JJA, autumn: SON and winter: DJF) should be sufficient.

Combining the seasonal r values to an average value for the whole year may not be enough. Depending on the query, a certain threshold of minimum similarity for every season must be defined because there is no reason to show similarity of two locations, where one season is not similar at all.

The methods used have two main advantages to prior attempts. First, it is possible to quantify similarity by generating a value between 0 and 1. A pair of regions can be "more" or "less" similar than another one. For the cartographic representation this means, that coloring similar regions can be continuous as the similarity value can be translated to the saturation value of a certain color. Therefore the visual representation of Climate Twins can be continuous instead of discrete. The second advantage is that more r values from different points of time and even different climate indicators can be combined to an overall measure of similarity.

3.1.1 Test data

The test data and the data implemented into the application is from the COSMO-CLM (COnsortium for Small-scale MOdelling - Climate Local Model) model 2.4.11 which receives it's boundary input from the ECHAM5/MPIOM global model. The climate of the 20^{th} century was modeled three times with different initialization

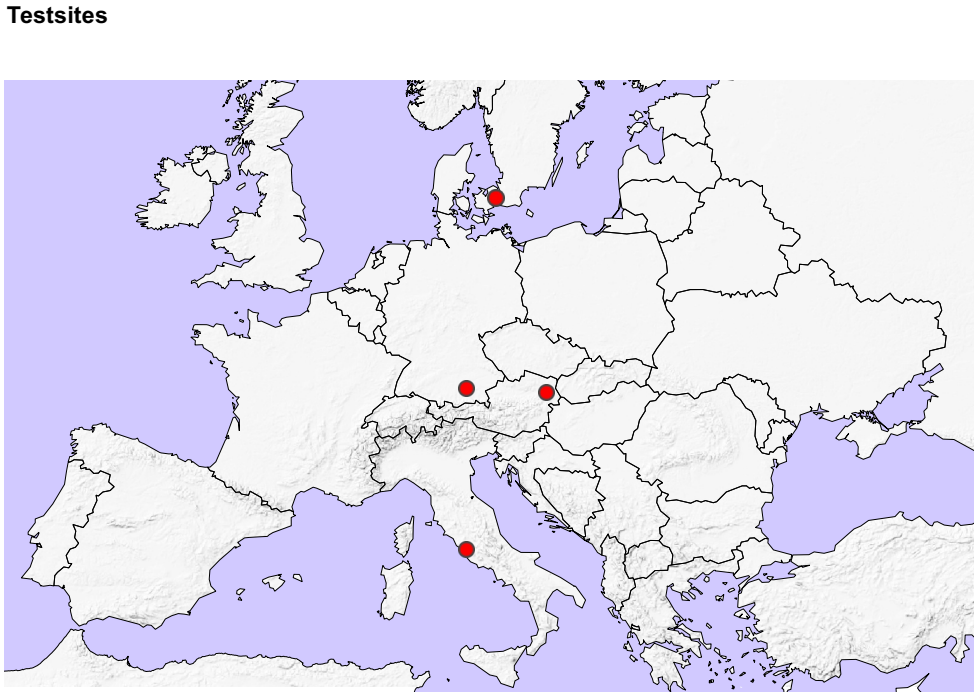


Figure 3.1: Locations used to explore similarity measures

times, the 21st century was modeled according to the A1B IPCC scenario, which is based on moderate demographical, economical and ecological assumptions. The horizontal resolution is 0.165°, or around 18 to 20 km on a rotated grid. (Lautenschlager et al. 2009)

The selected testsites (see Figure 3.1) are the cities of Vienna, Copenhagen, Munich and Rome. It is assumed that they differ in climate because of their maritime (Copenhagen, Rome) vs. continental (Vienna, Munich) and their unequal latitudinal (Copenhagen vs. Rome) position. This should affect both temperature and precipitation patterns. Because Vienna and Munich are quite close, it is also expected that they show more similarity than the other locations.

3.2 Analysis of similarity coefficients regarding performance and applicability

3.2.1 Algorithm programming - test version developed in R

The PD and the r_{H} were implemented in the open source statistic software environment R (http://www.r-project.org). Both similarity measures are not well-known and therefore no standard functions of R or any similar software exist. The Appendix Section A.1.1 contains the source code of the scripts.

As the calculation of the inices is quite similar, it was possible to carry out both within one script. Optional it is possible to "smoothen" the distributions by applying a moving average filter (`ma=x`, where `x` is the width of the filter) or a `log` filter, which

lowers extreme high values. The exact meaning and exploration of these filters is described in Subsections 3.2.4.2 and 3.2.4.3.

For developing purposes a debugging output can be created by adding the parameter `debug=TRUE`. Here, additional information to the category borders, the absolute and relative frequencies and other can be found. All of the calculations and graphs made for this thesis were calculated by applying this one script.

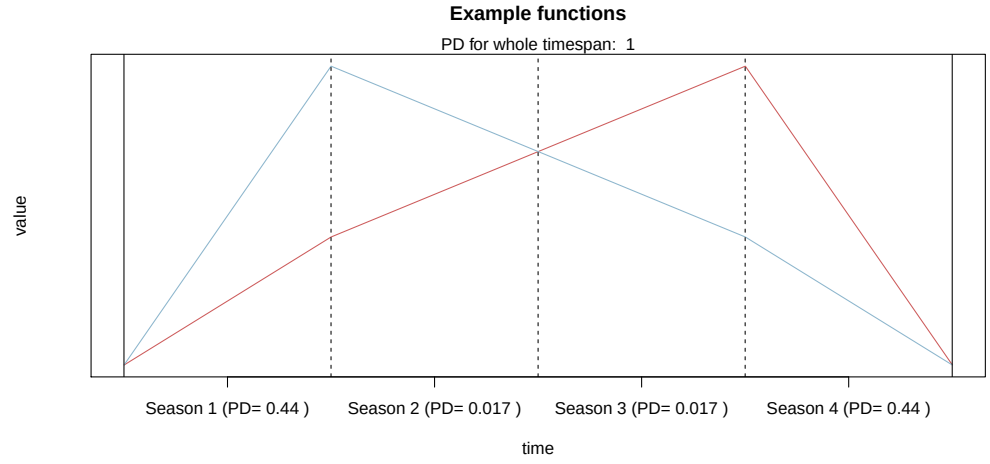
3.2.2 Generic requirements exploring similarity of distributions

As input, both functions need the two value vectors of the distributions and the number of categories. Furthermore the range to distinguish the categories has to be defined. In any case the parameters have to be selected in a way where as much categories as possible are filled with as much values as possible to achieve serious results (see theory Subsection 2.2.3.2). The choice of setting the parameters is also influenced by the decision towards a static or dynamic range. A dynamic range would better fulfill the demands above, but a static range secures the comparability of multiple similarity measures.

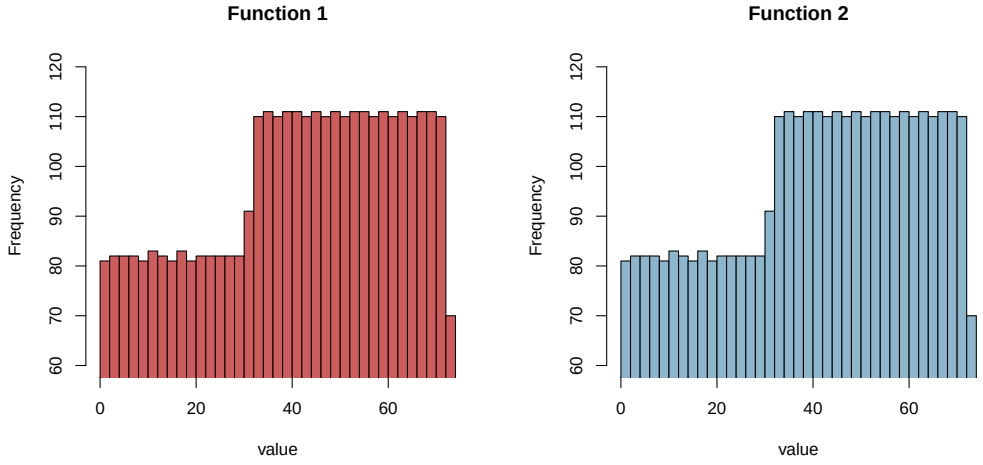
In Figure 3.2 two functions and their respective frequency distributions are shown. Although the functions are mirrored and peaks occur at different days over time, the frequency distributions are exactly the same which means that the r value derived will be 1. As mentioned before, when having two different temperature or precipitation curves over a year, a similarity would be found although the summers for example are completely different. For including sequence information, which gets lost in frequency distributions, the data has to be split in subsets of certain time-spans. For this application splitting the data into spring, summer, autumn and winter subsets gives an easy and transparent method.

Comparing the seasons reveals dissimilarity as season 1 and 4 have an r value of 0.44 and Season 2 and 3 just 0.017. Applying the approach of measuring seasonal similarity a big amount of incorrect climate twin regions is being filtered out. For the final application minimum similarity thresholds for every season have to be fulfilled and a minimum threshold of the combined r values will delimit the query results further.

The values of both data sets lie within 0 and 73.25, the PD was calculated with 50 categories between 0 and 100. This means that for every single measurement, the framework was the same. My R Script provides another option (`dyn=TRUE`) to build the categories not within fixed borders (e.g. 0 and 100) but between the minimum and the maximum value of both distributions. For this application where many r values are computed, a static framework is necessary to keep the integrity.



(a)



(b) (c)

Figure 3.2: Example: Functions and their distributions

Also, the number of categories plays an important role. Comparing Season 1 data with 50 categories revealed a value of 0.44, but computed with 2 categories the result is 0.68, with 3 it is 0.45 and with 1000 categories it is 0.43. A more exact analysis of an applicable category framework is done in the next section but before determining an applicable framework, some requirements have to be defined:

1. The framework has to provide accurate r values in a way that a slight variation on parameters must not change the r value totally.

2. The similarity measurement should be kept as less complex as possible not only because of transparency but also because of the technical implementation in the Climate Twins application, allows to compute results within an acceptable calculation time.

3. The r values produced should be dispersed widely to facilitate the query of similar regions.

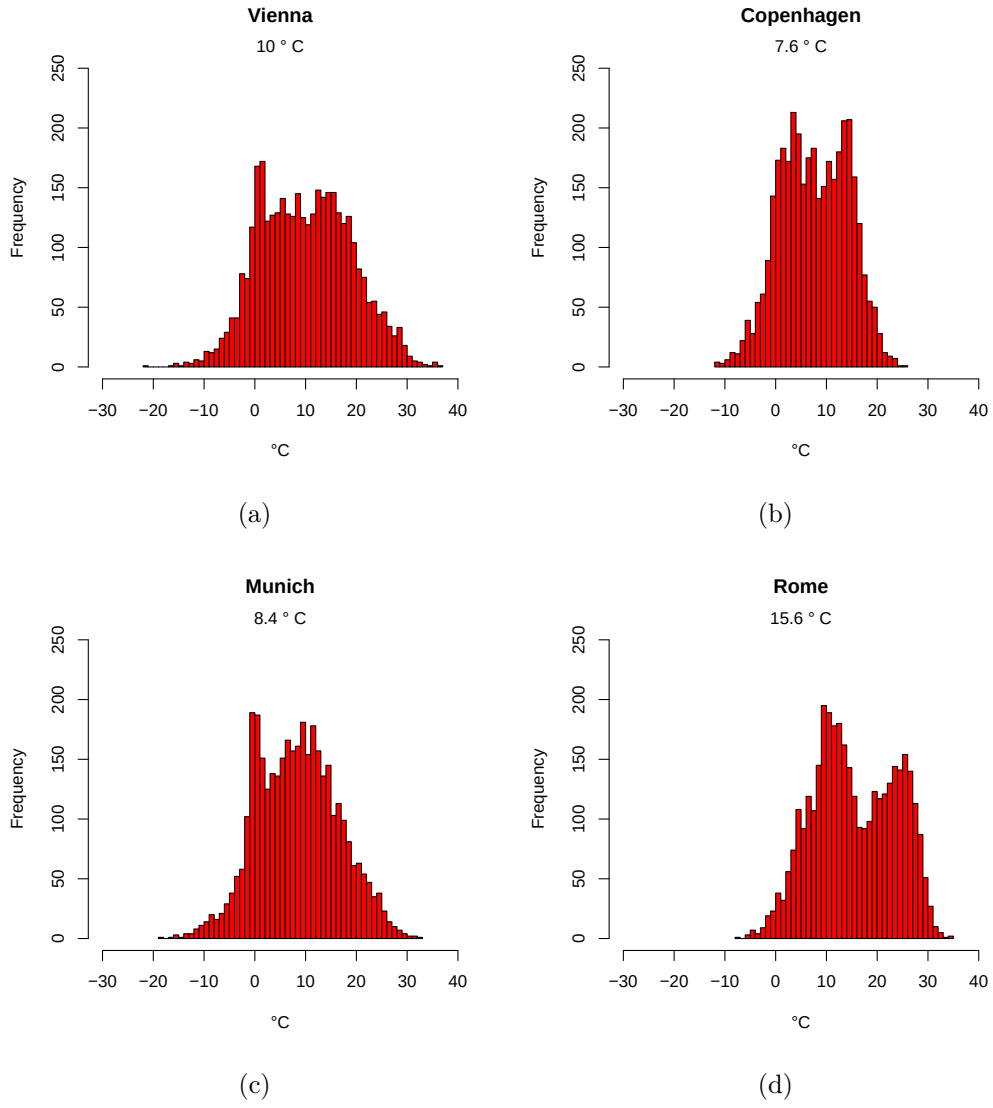


Figure 3.3: Frequency distributions of daily mean temperatures 2001 to 2010

3.2.3 Tests applying temperature data

Temperature is one of the most important climate variables. It is easy to measure and a key variable in every weather report. Temperature varies not only within a year but also within day and night. These data are modeled daily mean temperatures within 2001 and 2010. With this data it is possible to discover the amplitude within a year but not the amplitude between day and night. The difference between day and night temperatures is also an important indicator of a location's climate but only an optional step further after examining the annual temperature curve and therefore not realized in this work.

Figure 3.3 shows the frequency distribution of the modeled daily mean temperatures of the four test locations. A visual interpretation of the temperature properties can be done quickly: The more values there are distributed toward the right side, the warmer is the location (e.g. Rome). Also the annual temperature amplitude can

be read easily as it is the same as the range or width of the histograms. Therefore, Vienna and Munich have a wider range and higher annual temperature amplitude which correlates with the fact that they are located in a more continental climate zone. Rome and Copenhagen have a more maritime climate because the sea decreases temperature amplitudes. A bipolar distribution indicates two strong and distinct seasons like winter and summer with short and alternating changeovers in spring and autumn (Rome), whereas a Gauss-like distribution indicates a more homogeneous climate (Vienna, Copenhagen, Munich). A histogram of temperature data therefore can provide information of the variability, total intensity and seasonality of the climate.

3.2.3.1 Determining an usable amount of categories

As shown in Subsection 3.2.2, an appropriate number of categories must be defined to get valid results. Figure 3.4 shows the behavior of the r value depending on the number of categories.

The r values start at 1 and decrease as the number of categories increases. The reason why it starts at 1 is clear: when a frequency distribution is calculated over just one category and this category includes all the data, the frequency is 100% which leads to a similarity of 1 between two distributions. An interesting property of the curve is that it runs not steadily and fluctuates due to the values swapping the categories. The r value should approach 0 if the number of categories goes towards infinity but it should only meet the 0 line if every value is unique and "occupies" its own category. In fact with the data used here, the curve remains static applying 600 or more categories. An interesting fact is that both similarity measures in the area of the first 5-10 categories the r value drops rapidly and the fluctuation of the curve is very strong (up to 30%).

The curves can be divided into three sections: the first section is the beginning of the curve, where the PD/r_{H} drops rapidly and fluctuates strongly and in the second section it declines more steadily with fewer fluctuations and reaches after about 600 categories the third, constant section. The borders can not be drawn mathematically exact but visually. In general the PD and r_{H} curves show the same pattern, but the r_{H} seems to run more smoother and more stable in the first section.

The saisonal r values behave similar to the annual values used above. As expected, saisonal differences are greater than differences in the annual temperatures. Especially in the most recognizable seasons summer and winter, the r values disperse much and provide a good conclusion about similarity and dissimilarity. In winter, Rome behaves totally different than the other three locations, which are all three quite similar during this season. In summer the distances between the similarity curves are more regularly but provide a similar picture of Rome being less similar to the other three cities. Vienna and Munich show the most similar temperature patterns in summer and in the winter it is, depending on the category width, either Vienna and Copenhagen or Vienna and Munich again.

According to the data used here, the number of categories should not be below 20 as there would be too unstable conditions for the framework like defined in Subsection 3.2.2. Estimating minimum and maximum values, category borders of -30° and

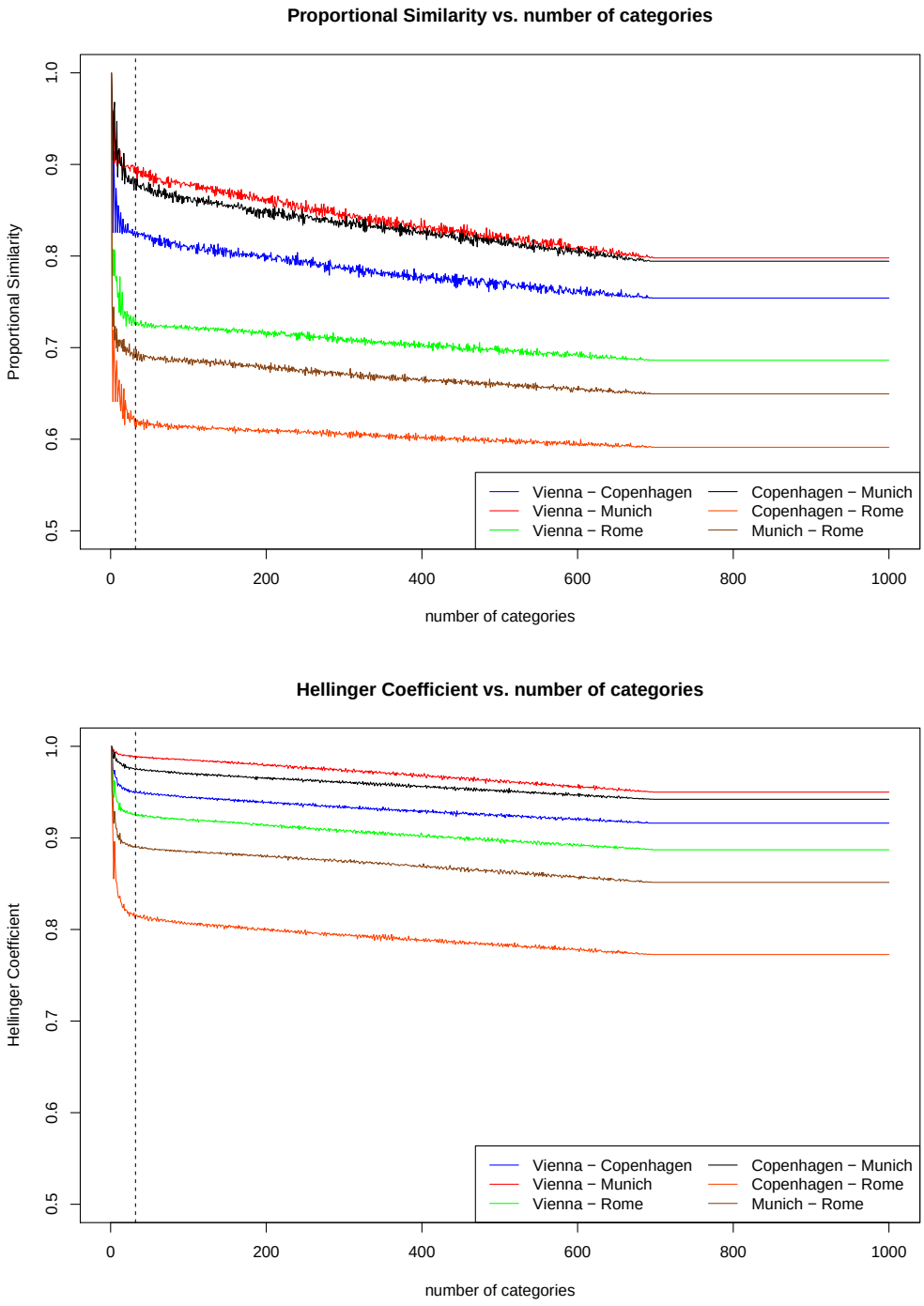


Figure 3.4: Influence of category number on similarity measures (daily mean temperatures 2001-2010): 1 to 1000 categories

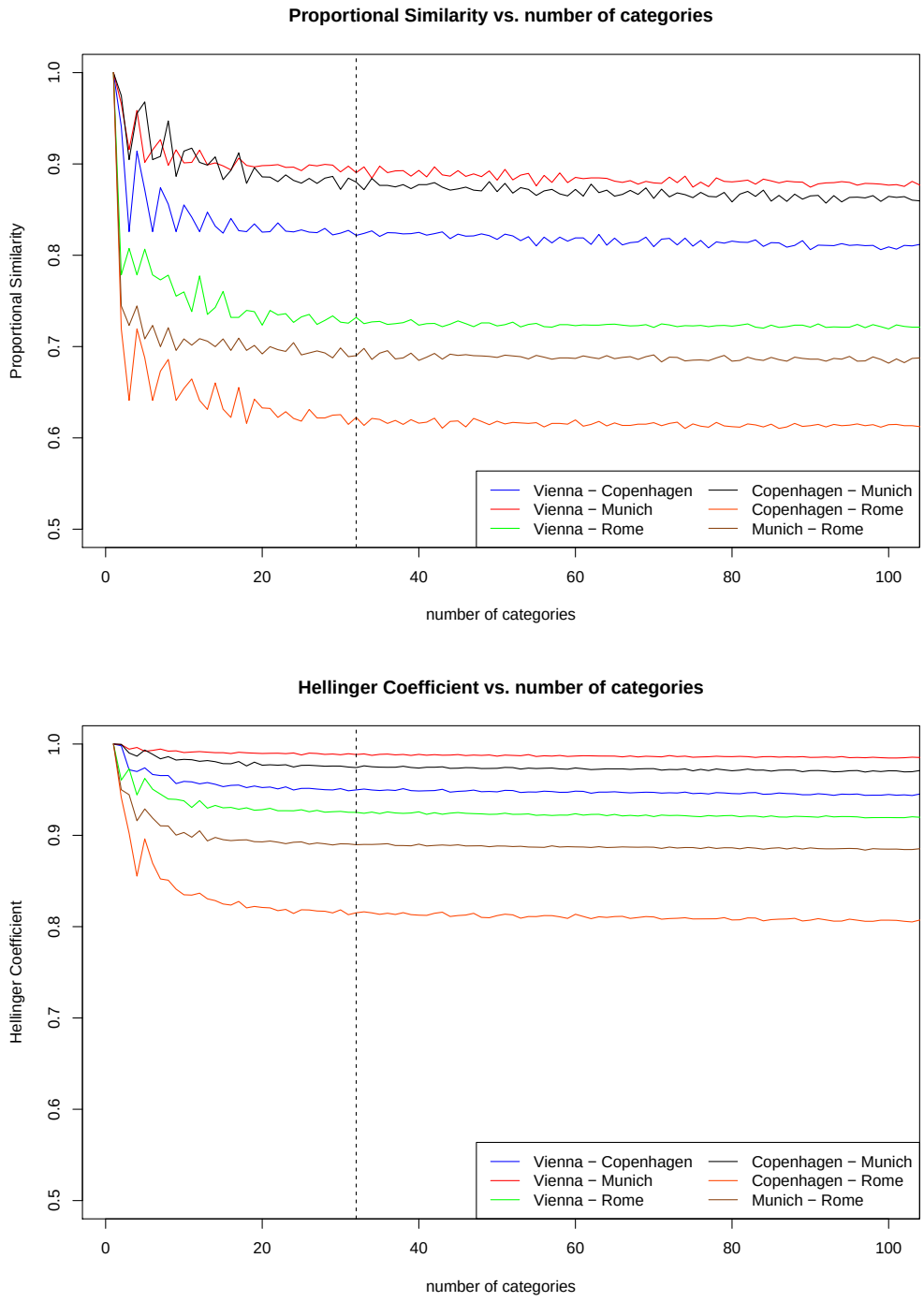


Figure 3.5: Influence of category number on similarity measures (daily mean temperatures 2001-2010): 1 to 100 categories

40° should be sufficient. For all values outside the category borders, a $< -30°$ and $> 40°$ category are introduced to be sure all values are included. Therefore a range of 70 with two lower and higher categories is set. Dividing the range into categories of 2°C width leads to 35 plus the two outer categories. The line of 37 is drawn in the figures to show that it is an applicable amount for temperature data.

The two similarity measures differ but over a small extent. In every case, the PD disperses the r values more than the r_{H} which makes the PD as a measure of similarity more applicable to temperature data. This conforms to Vegelius, who also supports PD.

3.2.4 Tests applying precipitation data

Handling precipitation data is a challenge in itself because the most interesting indices are the cumulated amount of precipitation within a fixed period and the occurrence of droughts and extreme precipitation events. Regarding vegetation the proportion of precipitation and evapotranspiration is an interesting indicator because it provides information on the water balance and therefore the growing conditions. The most common illustration of a location's climate condition is the Walter-Lieth-Diagram, merging temperature and precipitation into one diagram in which precipitation is presented by the monthly precipitation sum (Figure 3.10). Normally, the monthly average temperature curve would have to be added.

The major difference in temperature and precipitation data is that temperature is an omnipresent condition which means that at every moment the value temperature can be determined. Within this perspective precipitation at a certain point on a time line can only be determined by a binary value either "precipitation" or "no precipitation" which is not useful for most applications. Therefore quantifying precipitation is about determining the amount of rain or snow falling from the sky within a defined timespan. Hence similarity between two precipitation patterns has to be measured in another way. A glimpse on a simple histogram of the ten-year daily precipitation in Munich and Rome reveals the main problem in applying the PD and r_{H} (Figure 3.11).

Histograms of precipitation data have a disadvantageous shape for the PD and r_{H} because the first category of less than 1 mm, which is the definition of a day without rain, has exorbitant more entries than the other ones. Therefore this category has an extremely high relative frequency and if all frequency categories are weighted equally, the r value is higher and not well dispersed. Therefore, some data and category modifications have to be made.

3.2.4.1 Categorization

Both similarity measures PD and r_{H} don't require the categories having the same width as long as the sum of the categories include all of both distribution's values. Therefore it is possible to predefine the categories according to the precipitation distributions. Precipitation data is saved in millimeters with one decimal place but this is only the raw model data and in reality a measurement below one millimeter is irrelevant. However, ranges of higher and lower priority categorizations have to

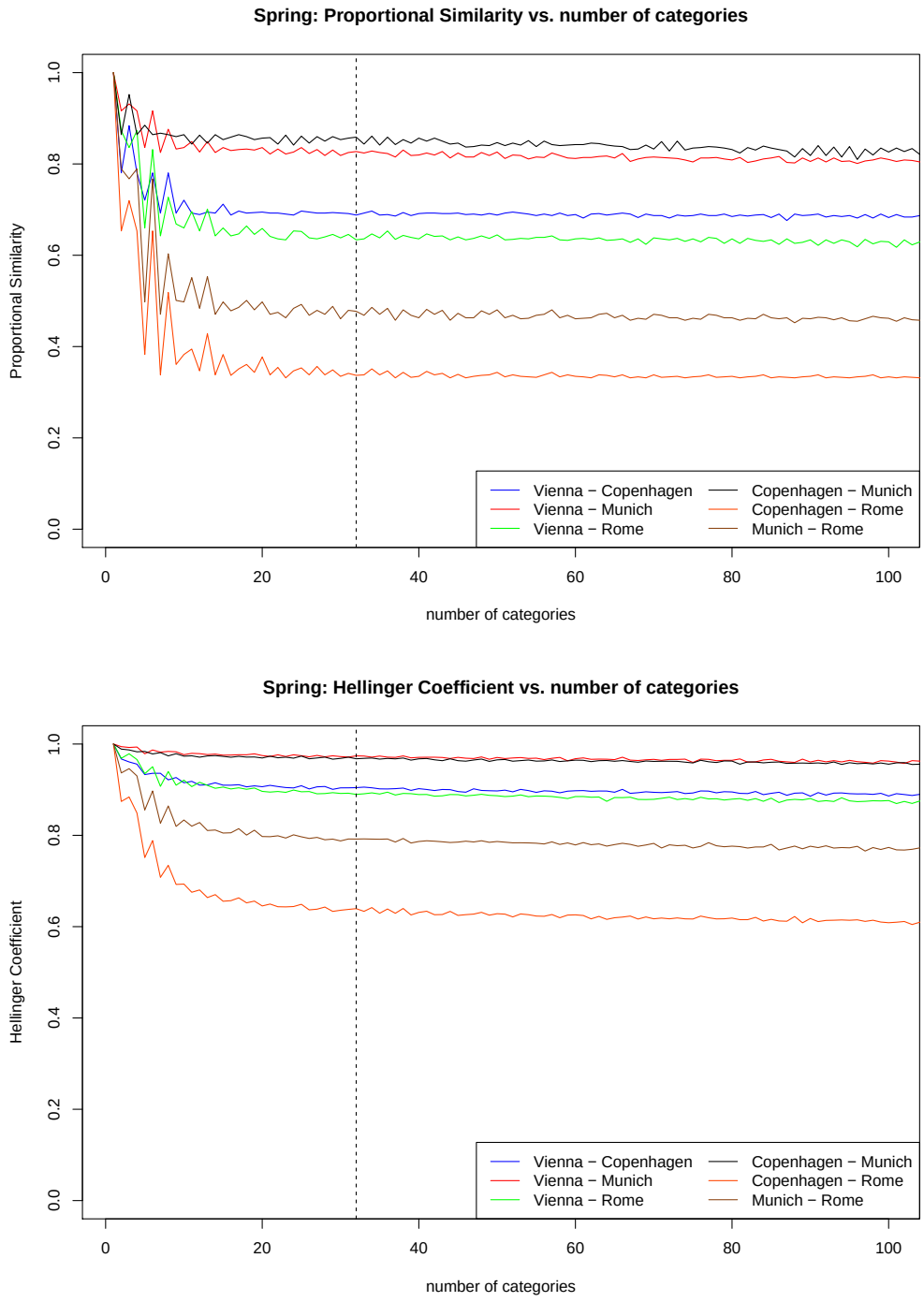


Figure 3.6: Influence of category number on similarity measures (spring daily mean temperatures 2001-2010): 1 to 100 categories

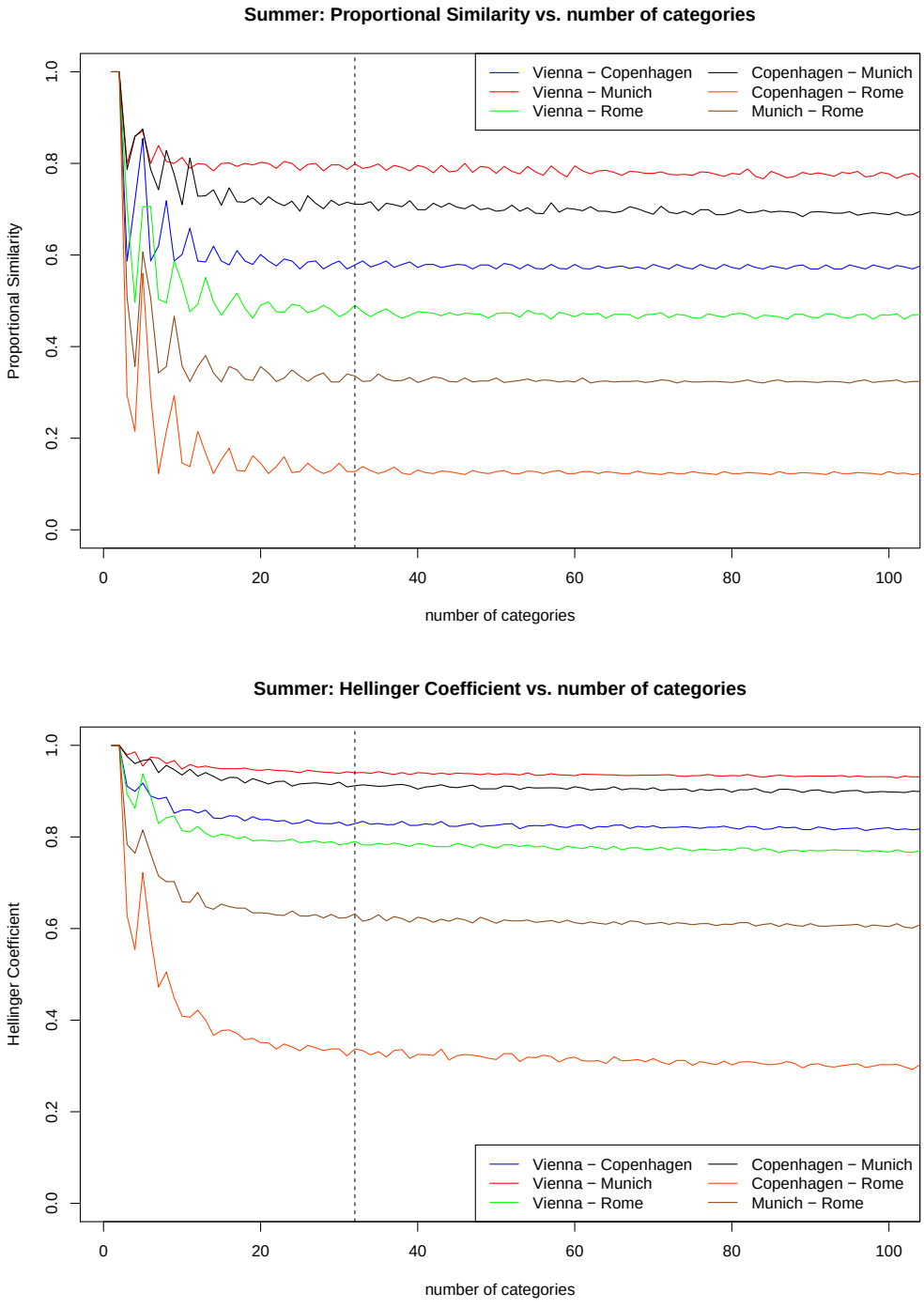


Figure 3.7: Influence of category number on similarity measures (summer daily mean temperatures 2001-2010: 1 to 100 categories

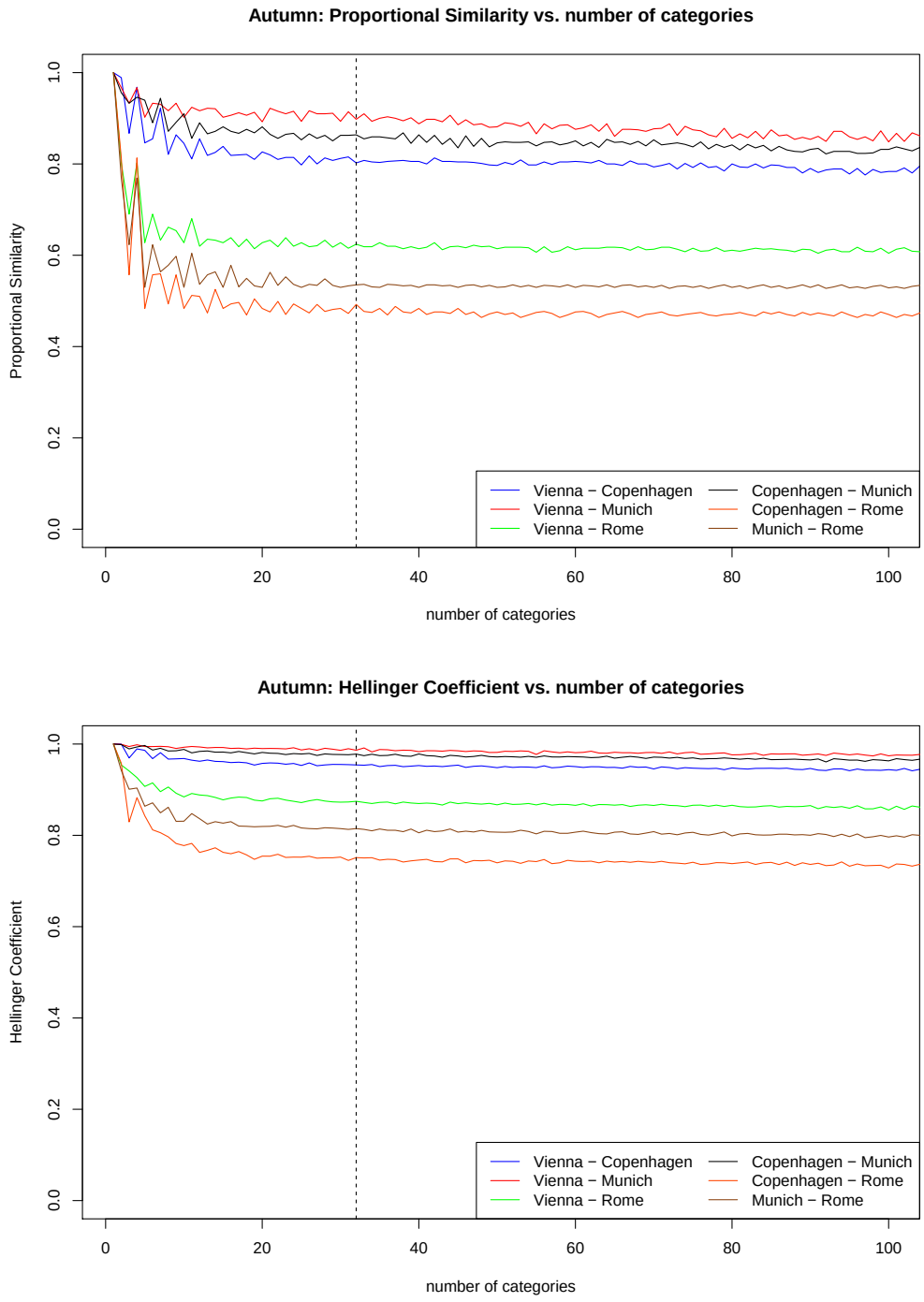


Figure 3.8: Influence of category number on similarity measures (autumn daily mean temperatures 2001-2010): 1 to 100 categories

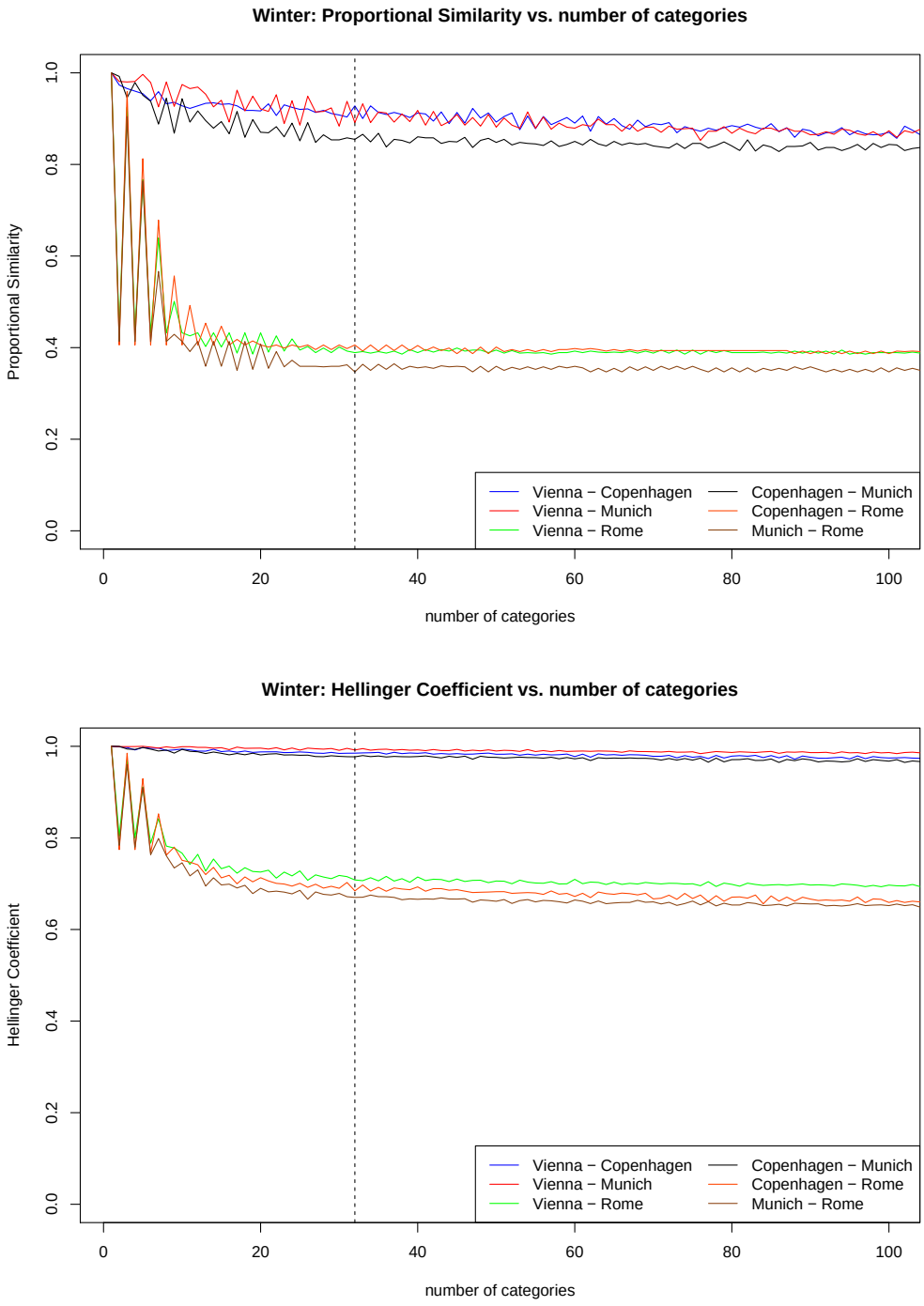


Figure 3.9: Influence of category number on similarity measures (winter daily mean temperatures 2001-2010): 1 to 100 categories

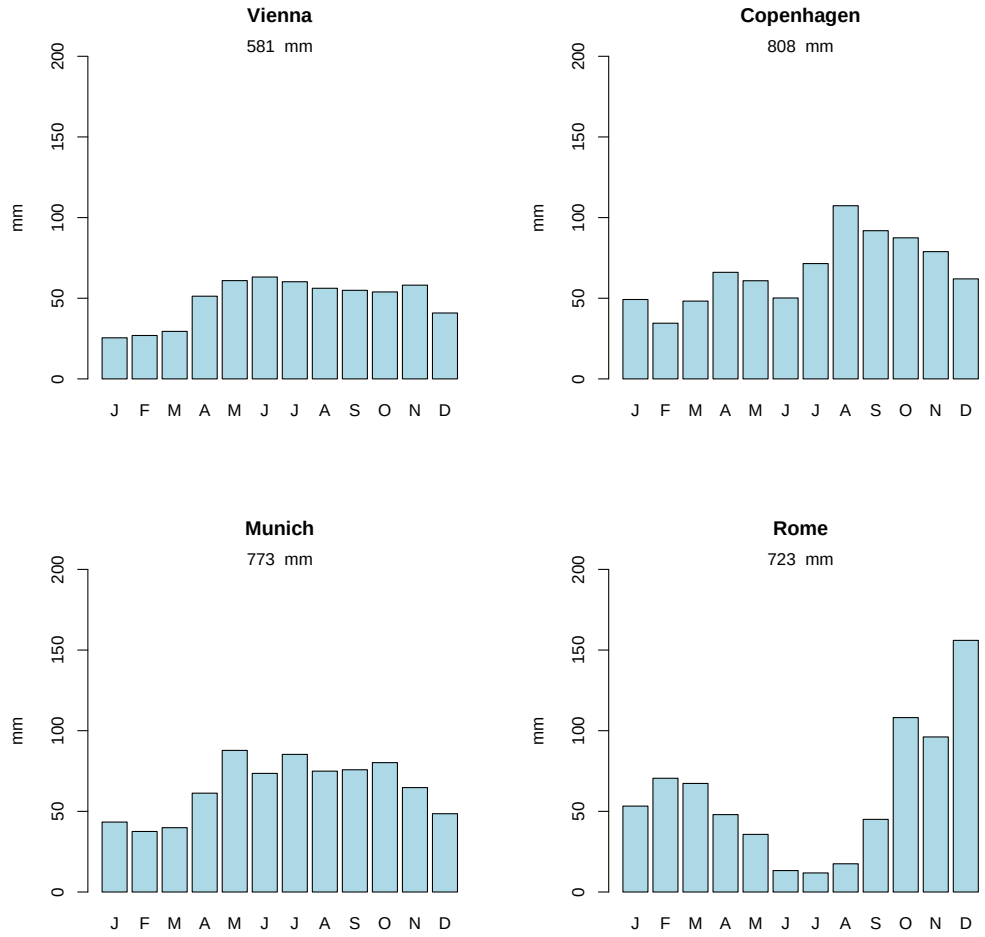


Figure 3.10: Monthly average precipitation sums, 2001 to 2010

be made which means that the most interesting part from 0 to 10 mm has to be more exact than the part of 21 mm and more. Daily precipitation over 100 mm is very sparse and can be merged into one category.

For this application the following categorization was selected (see source code in Appendix A.1.1): from 0 to 10 there are categories of 1 mm width, from 10 upwards to 100 the category width is 5 mm and for days with precipitation events exceeding 100 mm there is an extra category. In total there are 29 categories which is roughly the same as the number of temperature categories.

3.2.4.2 Moving average and moving sum filters

There are some methods to smoothen distributions, two of them are the moving average and the moving sum method. Given a certain day both methods compute either the average or the total precipitation within a given range of days before and after. The higher the range is set, the smoother the distribution occurs. Applying filters on a data set is always connected to a loss of accuracy, the uncertainty rises.

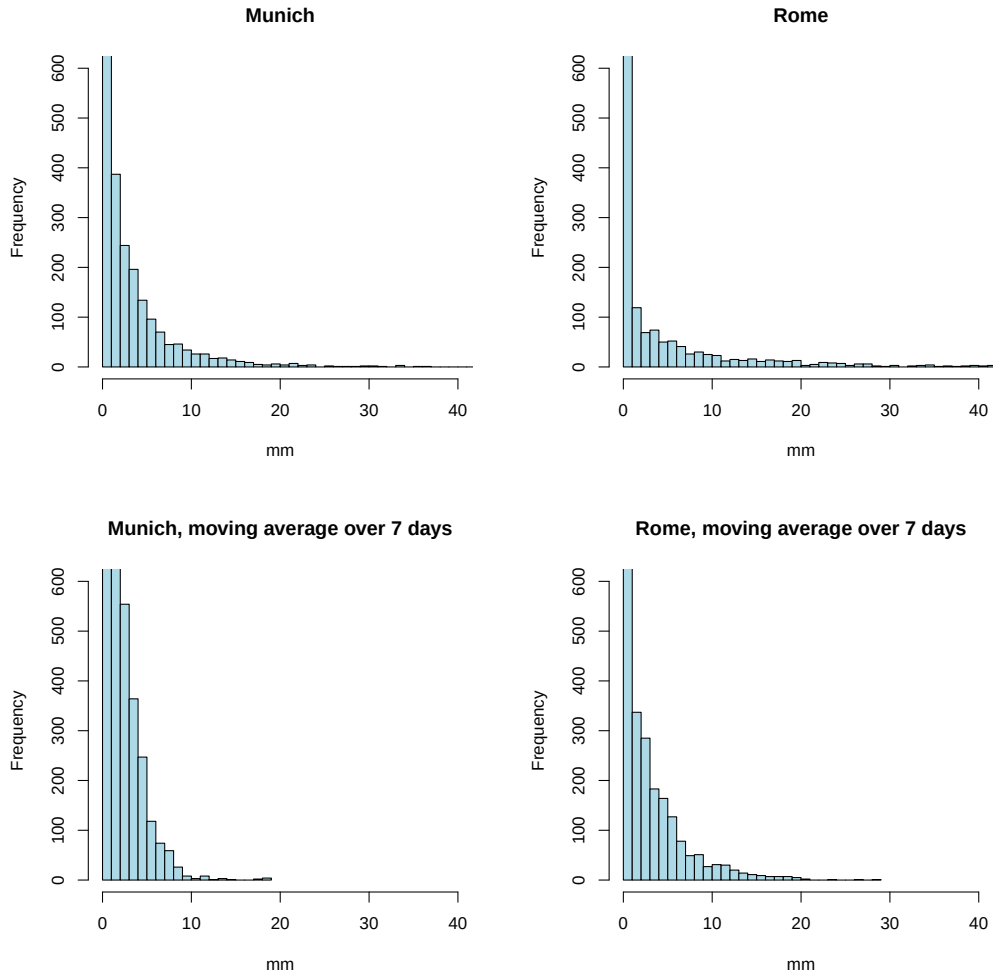


Figure 3.11: Yearly precipitation histograms of Munich and Rome, 2001 to 2010

Here, though, it is expected to emboss the characteristics as it moves the values away from the first category and leads to a wider distribution into more categories.

Using the moving sum filter won't provide any advantage as the moving average is the moving sum divided by the filter width. The values therefore would have to be multiplied by e.g. 7 (one week filter width) and afterwards the total range where the similarity measures compute the relative frequencies (now 0 - 100 mm) would also have to be extended sevenfold. Both similarity measures would lead to the same result.

3.2.4.3 Logarithmic flattening

Another possibility tried to work out precipitation characteristics was applying the `log()` function of R to the raw absolute frequencies by adding the parameter `log=TRUE`. The logarithmic function squeezes high values more than low values so it was expected to have a positive impact to the test data.

Applying the logarithmic filter seems not to improve the results at all. According

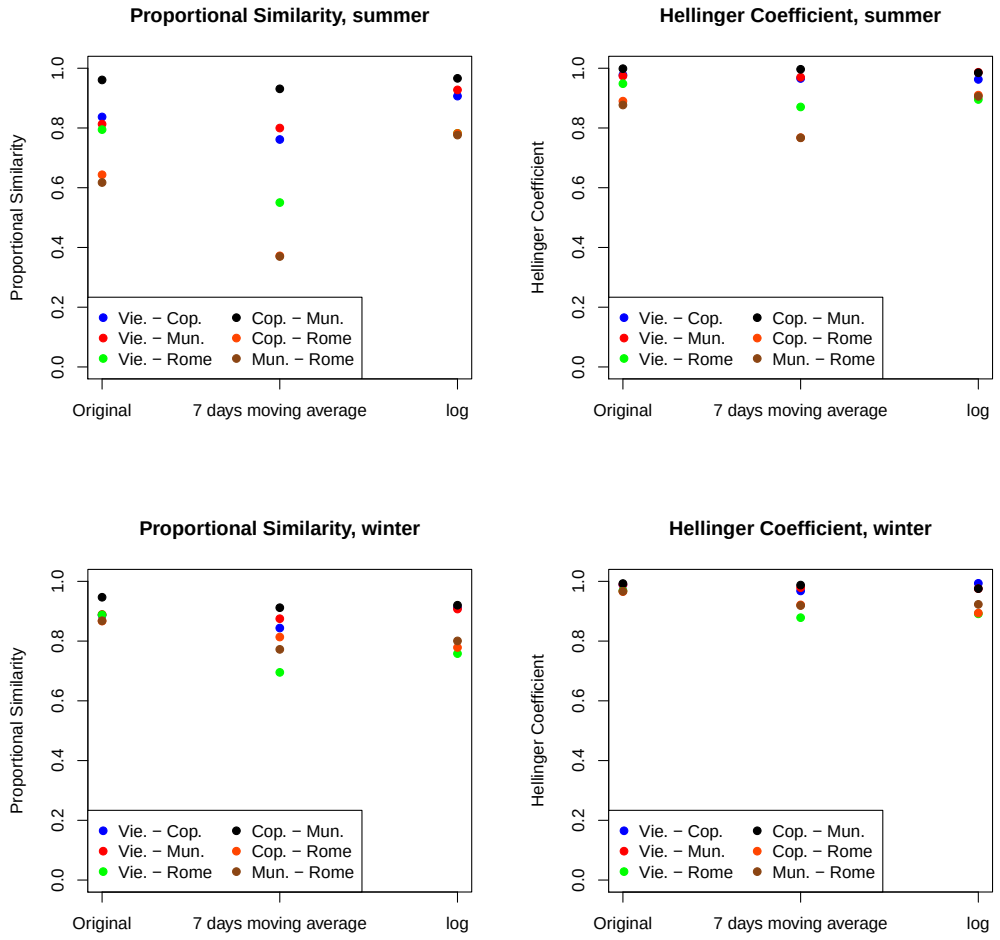


Figure 3.12: Comparison of filters and the r values of precipitation data (2001-2010)

to the test results of the four test locations in Figure 3.11, only the moving average provides a little improvement to the dispersion of the r values. There could be the assumption that the four locations could have similar precipitation patterns but according to the precipitation sum diagram in Figure 3.10 this is highly implausible as both, the sums and the temporal distribution, is highly dissimilar.

3.2.5 Combining similarity measures

As mentioned above the r values represent an "unit"-free index of similarity which allows combinations of r values from different indicators. For this application two kinds of combinations have been applied. One combines the respective saisonal values to a value for the whole year and the second one combines the values of the different climate indicators measured. In both ways the values are combined by averaging them, as an average of values between 0 and 1 again computes a value between 0 and 1 which could be easily processed further.

A problem arises when combining the values from different climate indicators.

As shown in Subsections 3.2.3 and 3.2.4 the statistical distributions of temperature and precipitation data are different and therefore the similarity indices are different. A r value of 0.8 in temperature similarity shows a higher coincidence as the same value of 0.8 for precipitation similarity because precipitation based r values seem to disperse less than temperature values. Besides there is the question whether a statistical distribution represents temperature characteristics in the same quality like it does with precipitation characteristics.

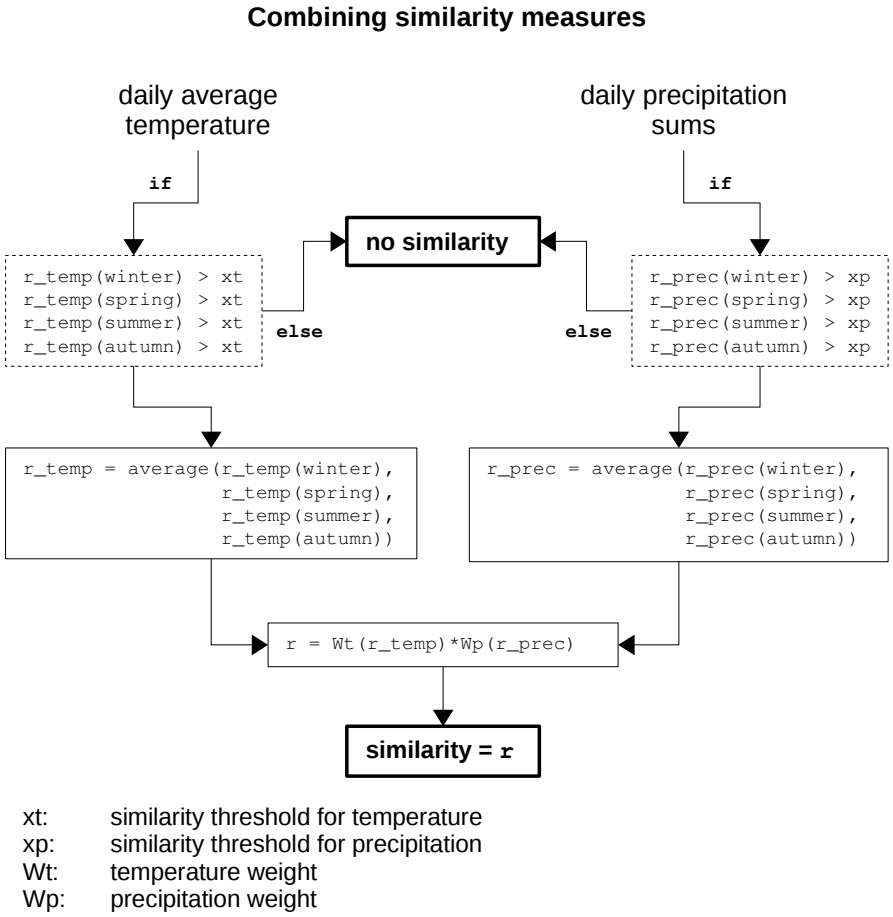


Figure 3.13: Logical structure of combining r values

If precipitation similarity values are in general higher than temperature values, they have more impact when merging them to a combined climatic similarity value. The evaluation of these similarity differences is complicated and requires more time and data to re-check the classification. For this application the weighting of the indicators can only be estimated.

Figure 3.13 shows the basic logic of the similarity exploration implemented in the Climate Twins application. First all of the seasons (temperature and precipitation data) have to show a minimum similarity according to the thresholds `xt` (temper-

ature) and `xp` (precipitation). If any condition is not resolved, the current region will not be identified as a Climate Twin region. If all seasons match, the r values of the seasons are averaged to aggregate them. After this step, there are only two r values left, one for each climate indicator. These two values are combined by multiplying them after weighting them with the factors `Wt` (temperature) and `Wp` (precipitation). The resulting r value is a value between 0 and 1 and influences the saturation of the target cell's color, thus a continuous gradient from "low similarity" to "high similarity" can be shown.

3.2.5.1 Aggregating saisonal values

Both similarity measures work well as expected in comparing saisonal temperature patterns. This reveals the possibility to implement a basic inclusion of the temporal distributions when combining the particular r values of the seasons to one year. The most simple option is to build an average r value of the four season values to compare annual temperature patterns. To assess the temporal distribution's influence on the similarity, the averaged r values are compared to the r values calculated for the whole year distribution.

The result is shown in Figure 3.14. Averaging the saisonal values provides a more exact picture of similarity than a measurement of the whole year's data as the factor time is included. As the distinct saisonal r values were computed with the same parameters like the identical categories, a combination by averaging them is a valid way. In addition introducing a filter, which excludes regions where the r value falls below a certain threshold in any of the seasons would make sense. There is no point in presenting a similar region, where one season is not similar at all and by the way the drop-out rate of potential Climate Twin regions could also be increased.

For precipitation patterns (Figure 3.15) r values derived by PD disperse more than the ones from r_{H} and therefore should be preferred. The aggregation can be done the same way as with the temperature values but a separate threshold shall be found.

3.3 Defining similarity coefficient thresholds

The thresholds at this stage can only be defined arbitrarily in a meaning that no calculation or estimation method could be found and Vegelius et al. (1986) do not recommend any threshold value. The threshold should of course be tight enough to provide a reliable similarity result but on the other hand not so tight that no Climate Twin region can be found. As mentioned above, the thresholds should also be estimated individually for every climate indicator. A glimpse on the graphs in Subsections 3.2.3 and 3.2.4 reveals that a value of around 0.9 could be sufficient. In order to let the user participate in the decision of the accuracy of his map, the actual threshold values are able to be modified by a slider in the web application's front end.

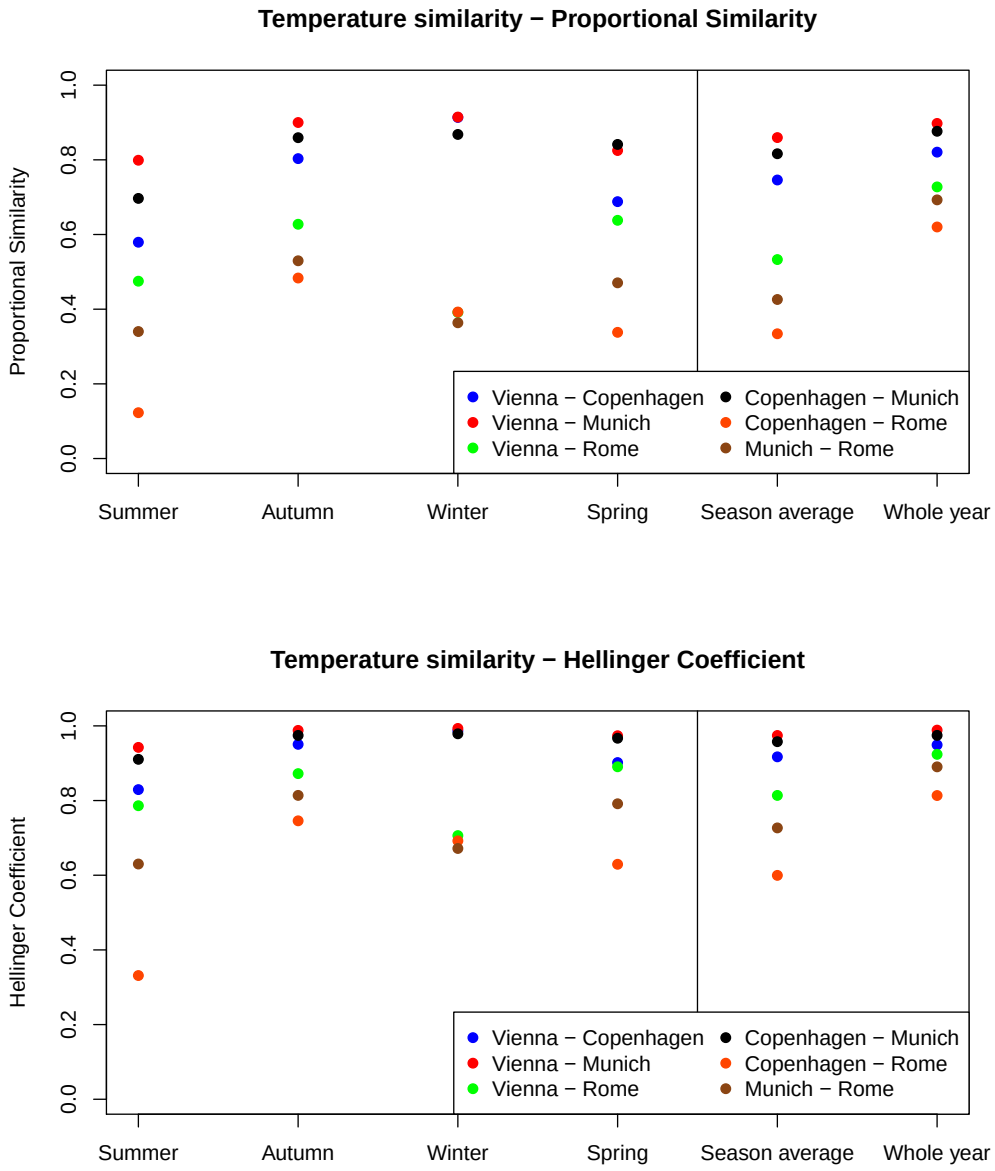


Figure 3.14: Similarity values per season, average of seasons and similarity for whole year's daily temperature

3.4 Discussion

This analysis showed that concerning temperature data, the Hellinger Coefficient is more practicable as there are less categories needed and the curves turn out smoother indicating stability. The category number according to the data used here should be at least 10 but as the different climate types of a larger (COnsortium for Small-scale MOdelling - Climate Local Model) area are expected to be more variable, a number of 20 to 40 should bring satisfying results.

However, there are some points left to describe temperature variation, namely

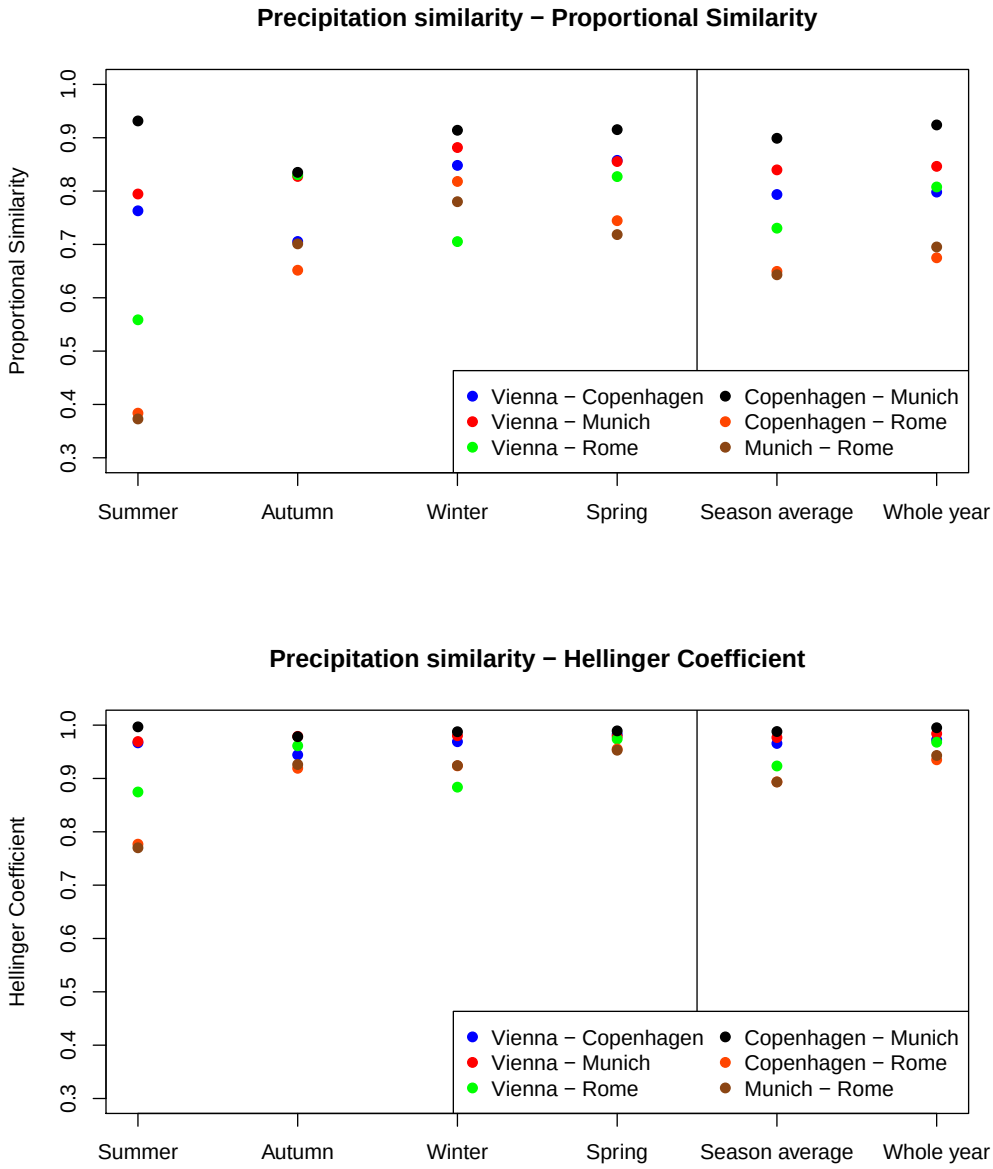


Figure 3.15: Similarity values per season, average of seasons and similarity for whole year's daily precipitation

the daily minimum and maximum values. They are important indicators as certain minimum and maximum temperatures affect vegetation. Also the daily temperature amplitude is being well recognized by people, especially when going out in summer nights. The daily temperature range should be therefore considered in describing climate. These data should behave similar to the daily mean values.

4 Prototype architecture: improving the Climate Twins tool

4.1 Climate Twins tool

The Climate Twins application is an interactive web mapping application basically consisting of climate data stored in a database, the matching algorithm and the front end with a query map and the result map. The climate data currently used are the German COSMO-CLM simulated data from 1960 to 2100 covering Europe. In a first version the matching was done by comparing the monthly mean temperature and precipitation sums. A region was defined similar if the differences between monthly mean temperatures did not exceed an uncertainty range of $\pm 4°C$ and the monthly precipitation sums $\pm 40\%$. (Loibl et al. 2010)

In future the Climate Twins Viewer should provide a broader functionality, though. As the intention is to create a tool for exploring future climate conditions and as mentioned above, modeling cannot be done without uncertainties, results from different climate models or approaches are planned to be implemented. In addition some focus has to be spent in optimizing the infrastructure because now it takes up to half a minute—depending on the similarity parameters—to get a result map. This is due to the fact that a lot of data has to be extracted out of the database, processed in a Java snippet and written to a PostGIS layer with an attribute table. Therefore the aim is to translate the similarity algorithm directly into the database query because the database engine can handle these kinds of calculations much faster. In order to optimize the database itself it could be transformed into a SOLAP (Spatial OnLine Analytical Processing) cube. An OLAP system (OnLine Analytical Processing) is designed to query and process huge and multidimensional data sets. SOLAP is a spatial extension and enhances the system with the ability to handle georeferenced data. By improving the speed of the application, unprocessed climate data could be stored within the cube on at least a daily if not an hourly basis which enhances the accuracy of the similarity measure (comparing frequencies of hourly climate data adds day and night values, hence the daily amplitude) and the possibility to add further tools like rendering climate diagrams.

4.2 Practical application

The application itself follows the basic structure of any non-static web page. The data in the background is stored in a database and is being presented via a web browser depending on some parameters given by the user. In this case it means that the database contains the results of the climate models and the front-end map shows the information depending on the parameters like the time spans, climate indicators or thresholds given by the user. However, the data has to be processed to show the desired information. In this case it is the similarity algorithm computing the similarity maps out of the stored distributions.

Figure 4.1 shows the logic behind the Climate Twins query. A click in the front-end's map selects a cell whose future climate patterns are used to compute similarity values between them and the present climate patterns of every single cell of the current climate. The result therefore is a similarity value for every cell that can be translated into a color optically representing regions of higher similarity by a darker

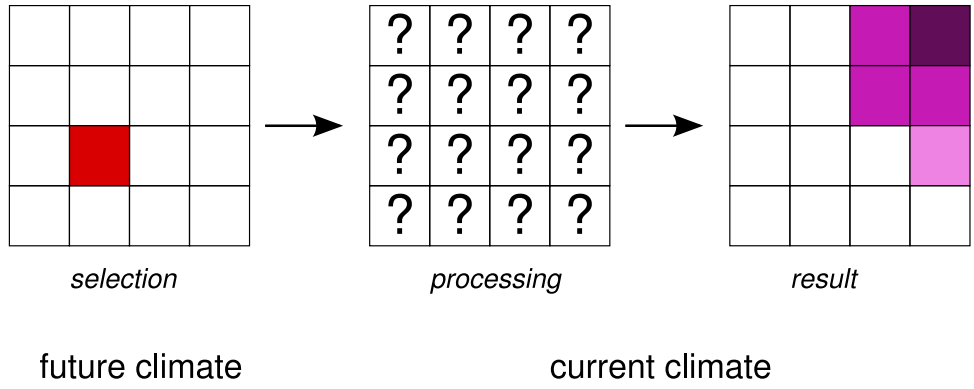


Figure 4.1: Schematic representation of the similarity matching module

color and lower similarity by a brighter color. The reason the color purple is used is because purple is a color with almost no distinct association. Most other relatively strong colors like red for example would indicate "heat" or "danger" or blue would indicate "coldness" or "moistness". The usage of a color associated strongly with other adjectives would mislead the understanding of the result map.

4.2.1 Database organization

In order to optimize computation time, the data was preprocessed inserted into the PostGIS database. The original calculation shown in chapter 3 requires daily climate data. The dataset contains 140 years which leads to over 50 000 entries for daily data. As every single cell contains daily data and there are over 60 000 cells, just the storage of the data in an effective and applicable way would be a challenge. Furthermore nearly all of the raw data is needed to calculate one Climate Twin query, so preprocessing had to be done where possible.

The most obvious way to optimize the query is to preprocess the parts of the calculations which are similar in every query. Every query requires the frequencies of predefined categories so the data inserted into the database were compressed in a way that every raster cell contains data vectors for every season combined in blocks of 10 years each. Every cell contains an array with the multiple values of the absolute frequencies. In other words there are $14 * 4 = 56$ columns (14 blocks of ten years each, e.g. 1961 to 1970 multiplied by four seasons) for both climate indicators temperature and precipitation.

ID	1961_1970_winter	1961_1970_spring	...	2091_2100_autumn
1001	temp[0,2,154,253,...]	temp[0,0,17,45,...]	...	temp[0,1,45,98,...]
...	...	...	...	...
255241	temp[2,5,94,178,...]	temp[0,3,67,125,...]	...	temp[1,4,35,74,...]

Table 4.1: Data structure

4.2.2 Algorithm integration, reprogramming in Java

The part of processing the data is being done by a Java program running on the server. According to given parameters it extracts the corresponding data from the database, executes the similarity measurement and writes the result back into the database. The parameters it needs are:

- ID of the source cell
- source region time period (one of 14 ten-year blocks)
- target regions time period (one of 14 ten-year blocks)
- temperature threshold value (`0 - 100`)
- precipitation threshold value (`0 - 100`)
- indicator weighting value (from `0` (100% temperature, 0% precipitation) to `1` (0% temperature, 100% precipitation))
- entire climate or one of the two climate indicators to be queried (`ENTIRE_CLIMATE`, `TEMP` or `PREC`)
- entire year or one of the four seasons to be queried (`ENTIRE_YEAR`, `SPRING`, `SUMMER`, `AUTUMN` or `WINTER`)
- similarity measure to be used (`PD` for the Proportional Similarity or `RH` for the Hellinger Coefficient)

The developed source code of the Java module called `ClimateConnector` can be found in the Appendix section A.2.1.

4.2.3 Climate Twins adaptation

The adaption of the new method is done by implementing the structures described above. It affects all parts of the application beginning at the data structure shown in Table 4.1, the new version of the `ClimateConnector` Java program and the updated front-end capable of providing the algorithm with required parameters. All changes did not affect the basic structure built for the first Climate Twins version although some weaknesses according the occurred which can be fixed by rebuilding the whole application considering the new challenges.

4.2.4 Application

The final application's front-end has all options implemented in a graphical user interface (GUI). Before selecting the desired source location, both time periods including the choice between a seasonal and an entire year matching, thresholds and weighting and the choice between indicators or an entire climate matching as well as the desired similarity measure has to be chosen. According to the inputs, the matching progress starts and shows the results in the map on the right side (Figure 4.2).

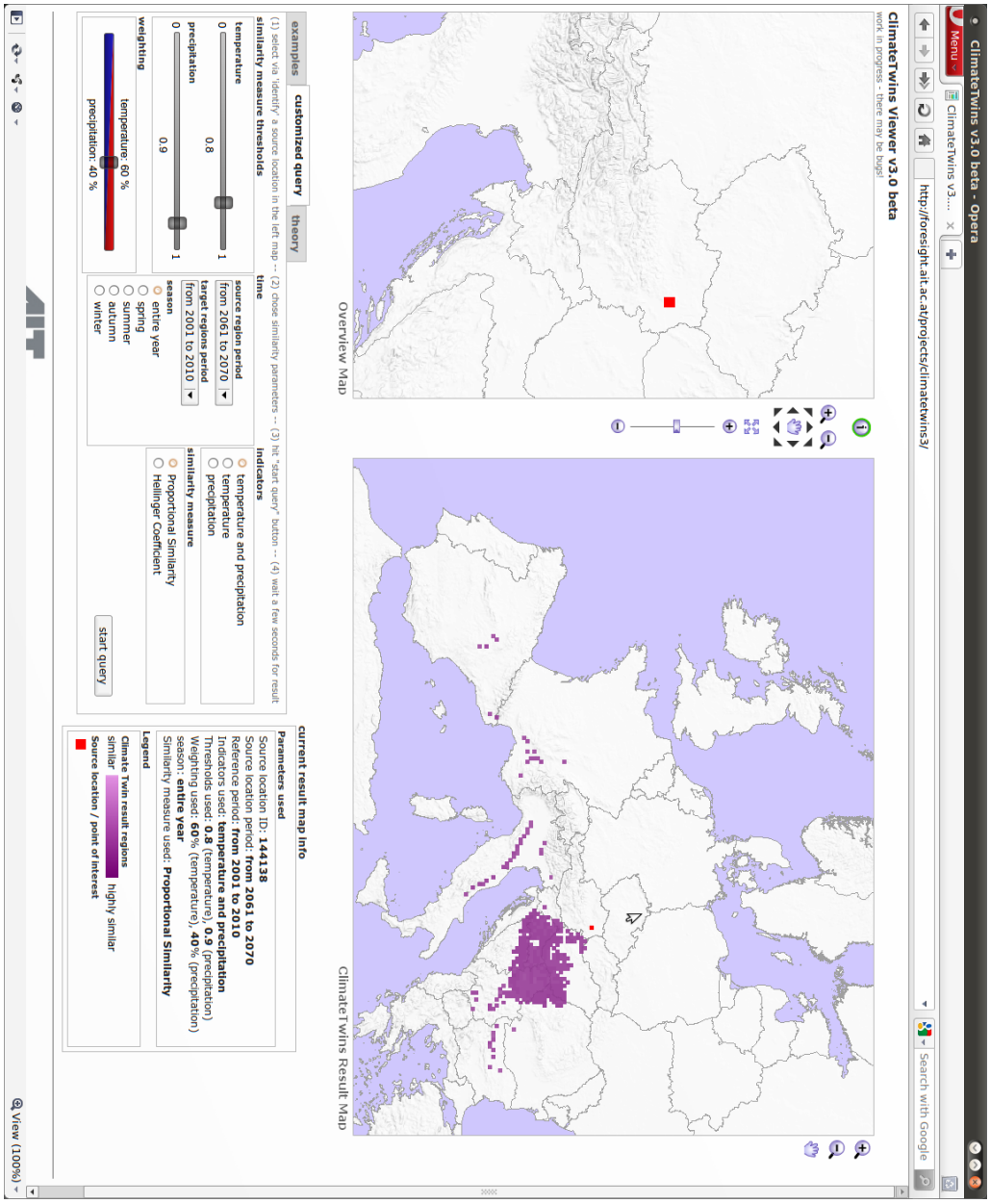


Figure 4.2: Application screenshot

5 Discussion and conclusion

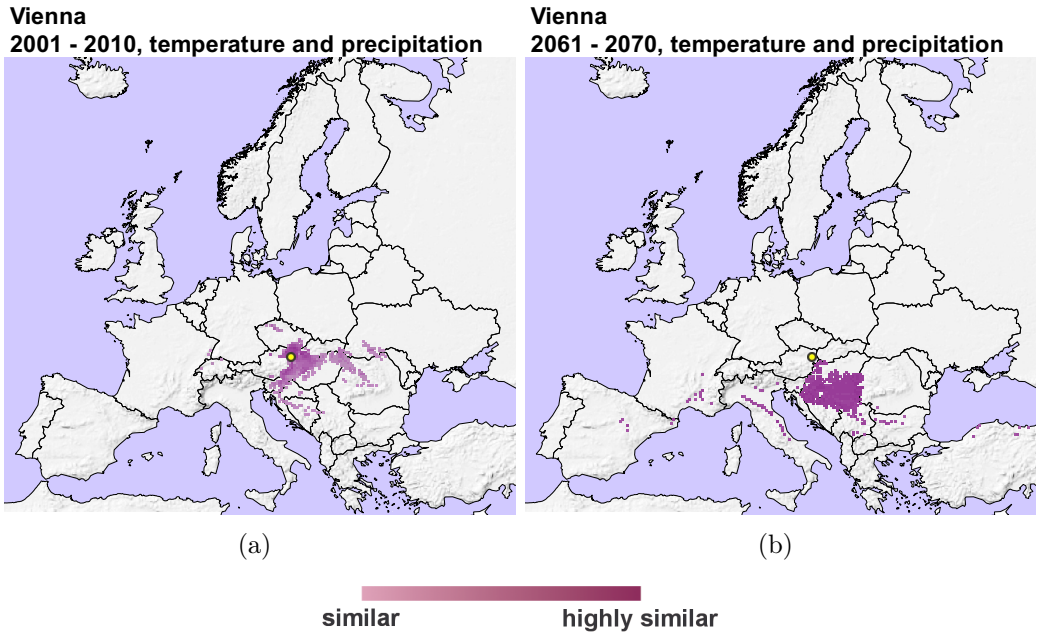


Figure 5.1: Vienna's Climate Twins now and in the period 2061 - 2070 (threshold temperature: 0.8, threshold precipitation: 0.9, PD)

5.1 Results

In this chapter the main results of the practical adaption are described and discussed. The problem in discussing a dynamic application is that a more or less random sample out of the variety on input parameters has to be drawn. Therefore this analysis is based on subjective assumptions on the similarity parameters and predilections for some test locations.

A further quantitative validation of the results was not possible because there were no research projects found answering similar questions. There are some attempts to examine future climates with modeled data as found in Rubel and Kottek (2010). Aggregating "cells" or regions with similar climate parameters as done in climate classifications (see Subsection 2.1.1) is a different approach as there is a "statically" defined framework like certain predefined minimum or maximum temperature or precipitation values. The Climate Twins method queries regions on the basis of a framework given by an example region and cannot produce an overall map of Europe showing similar climate zones. So the method used here and its results should and can only be seen as a first approximation in solving such a problem.

The results show, as expected, in general a southward shift of the Climate Twin Regions as time progresses. According to the results, major climate changes occur for the 2060s and later on as seen in Figure 5.1 where the corresponding Climate Twins of Vienna are located at the continental regions of the Balkans. In comparison, the current Climate Twin Regions (by comparing the period of 2001 to 2010) of Vienna are, of course, located in and around Vienna and its eastern adjacent regions mainly in Hungary, Slovakia and Slovenia and smaller regions in western Romania and northern Croatia due to the effect of spatial autocorrelation.

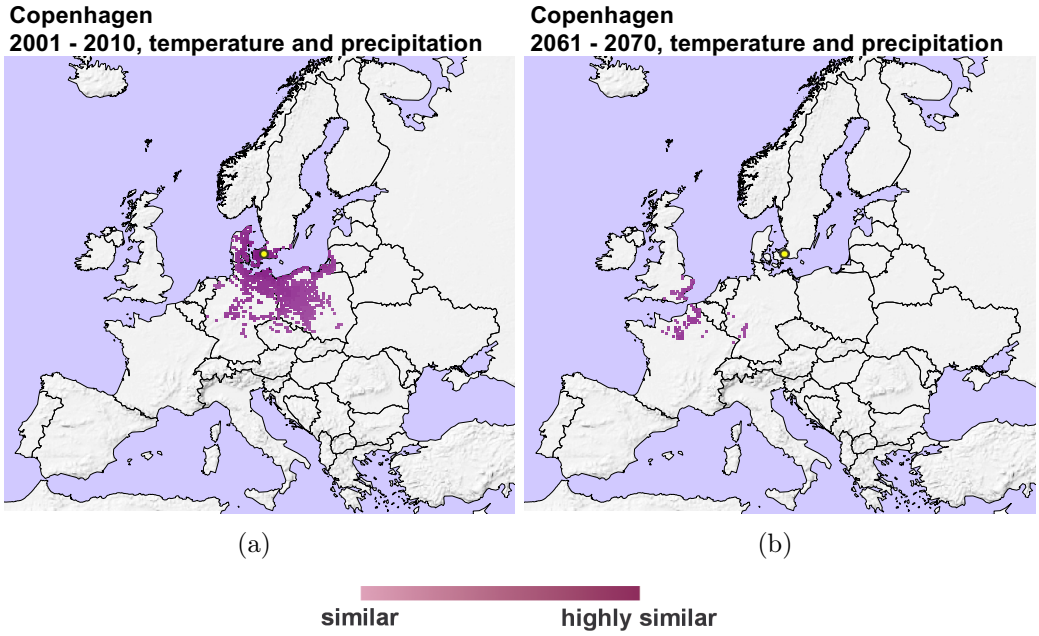


Figure 5.2: Copenhagen's Climate Twins now and in the period 2061 - 2070 (threshold temperature: 0.8, threshold precipitation: 0.9, PD)

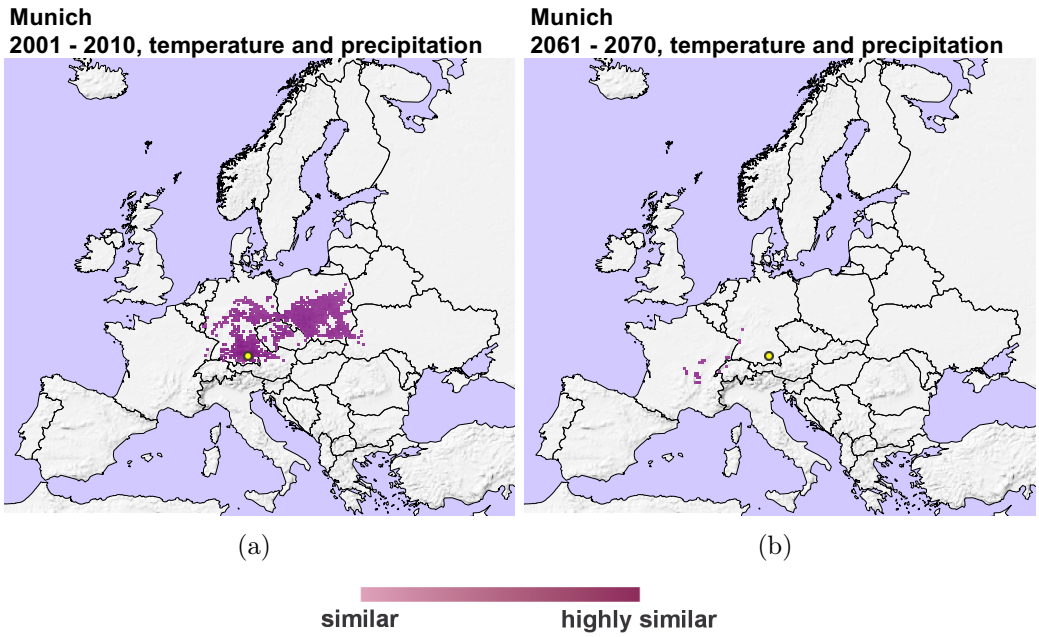


Figure 5.3: Munich's Climate Twins now and in the period 2061 - 2070 (threshold temperature: 0.8, threshold precipitation: 0.9, PD)

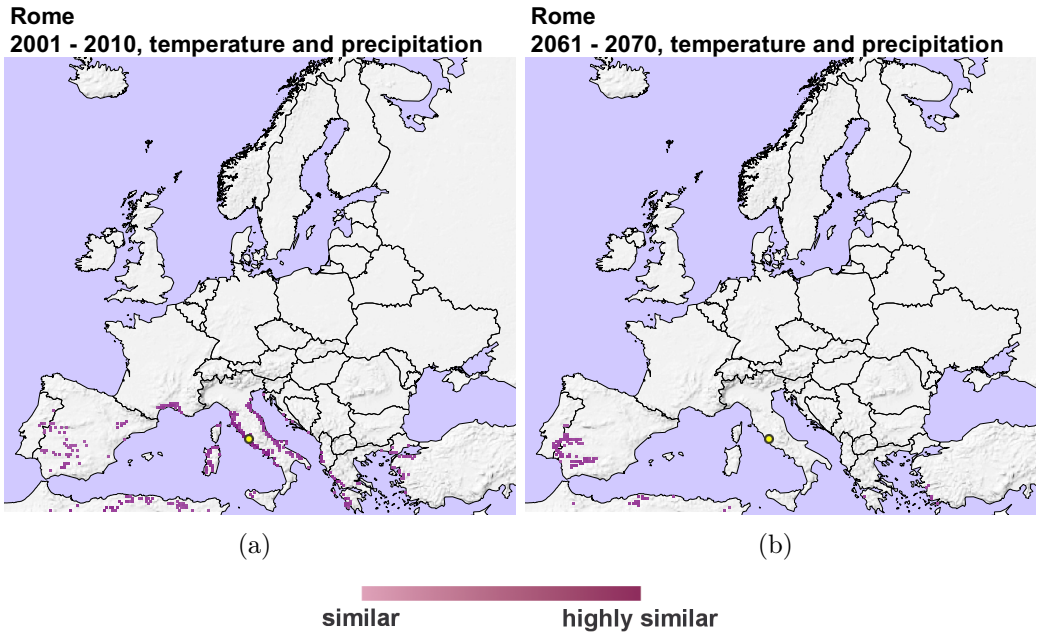


Figure 5.4: Rome's Climate Twins now and in the period 2061 - 2070 (threshold temperature: 0.8, threshold precipitation: 0.9, PD)

The biggest change of climate conditions can—also in the context of this work—be interpreted as a change that exceeds the used thresholds within this method so that the resulting regions do not cover the source region anymore. The further away a Climate Twin Region is (at least on a Europe-wide scale), the more distinct the changes are. There is some friction though: As the method uses daily average values, the diurnal amplitude of temperature is neglected so that regions located in northern Africa are marked as similar although they have colder nights and warmer days. Averaging these values leads to a daily average temperature that is the same as in a region with less extreme values. This happens for example in Figure 5.4 where some of Rome's Climate Twin regions are located in Northern Africa.

5.1.1 Climate indicators and seasonal results

To dig a little deeper into the process of generating the Climate Twins maps it makes sense to look at the intermediate results of the distinct indicator and seasonal similarities. The Climate Twin Regions are always the intersection of the regions with similar temperature and precipitation patterns but with the option to weight both climate indicators and thus change slightly the intersecting areas. The maps in Figure 5.1 were calculated with a 1:1 weight relation between temperature and precipitation. The seasonal aggregation to the annual similarity value is also equal weighted. So the process of combining the distinct results can be seen as simple GIS-like intersection of two or more layers. Comparing the overall common result maps with the single result maps reveals that more basic input parameters (e.g. winter precipitation similarity) achieve a larger coverage of matching regions. In almost all result maps the similar regions change drastically between the distinct

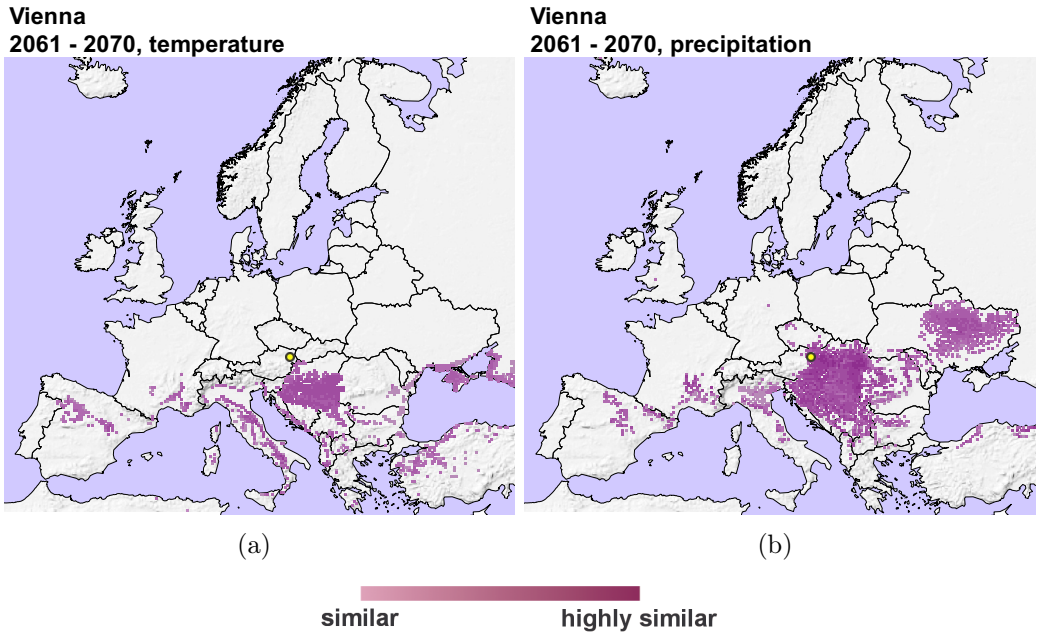


Figure 5.5: Regions with similar current temperature and precipitation patterns compared to Vienna 2061 - 2070

seasons, especially between summer and winter. Therefore the meaning of seasonal layer intersection and its influence on the final result should be accentuated.

A small example should illustrate this influence. Figure 5.5 shows both single result maps of the climate indicators for Vienna. Obviously the major similarities occur at the Balkan area but there are regions with similar climate in north central Spain, south western France, parts of Italy and south-eastern Ukraine. In these zones similarities occur in both climate indicators but just intersect in loose cells and thus are not visualized as eye-catching large Climate Twin Regions. Shifting the thresholds to widen the match range uncovers these regions and turns into Climate Twins.

5.1.2 Thresholds

The applied thresholds of 0.8 (temperature) and 0.9 (precipitation) used with the Proportional Similarity measurement seem to work well within the example of Vienna and both of the used time spans of 2001 to 2010 and 2061 to 2070 in the sense that a reasonable amount of Climate Twin areas are found. Reasonable in this context means on the one hand that there is at least one Climate Twin Region found on the one hand and on the other hand that there are not too many regions marked as Climate Twins to show characteristically similar regions.

As mentioned in Subsection 5.1.1, wider threshold ranges of 0.75 and 0.85 reveal more distinct Climate Twin regions in Spain, France, Italy, Romania and Ukraine. Within the former thresholds of 0.8 and 0.9, just a continental zone in the Balkan area was marked, within the new thresholds both continental (Spain) and maritime (Italy and areas around the Black Sea) zones are Vienna's Climate Twins in this

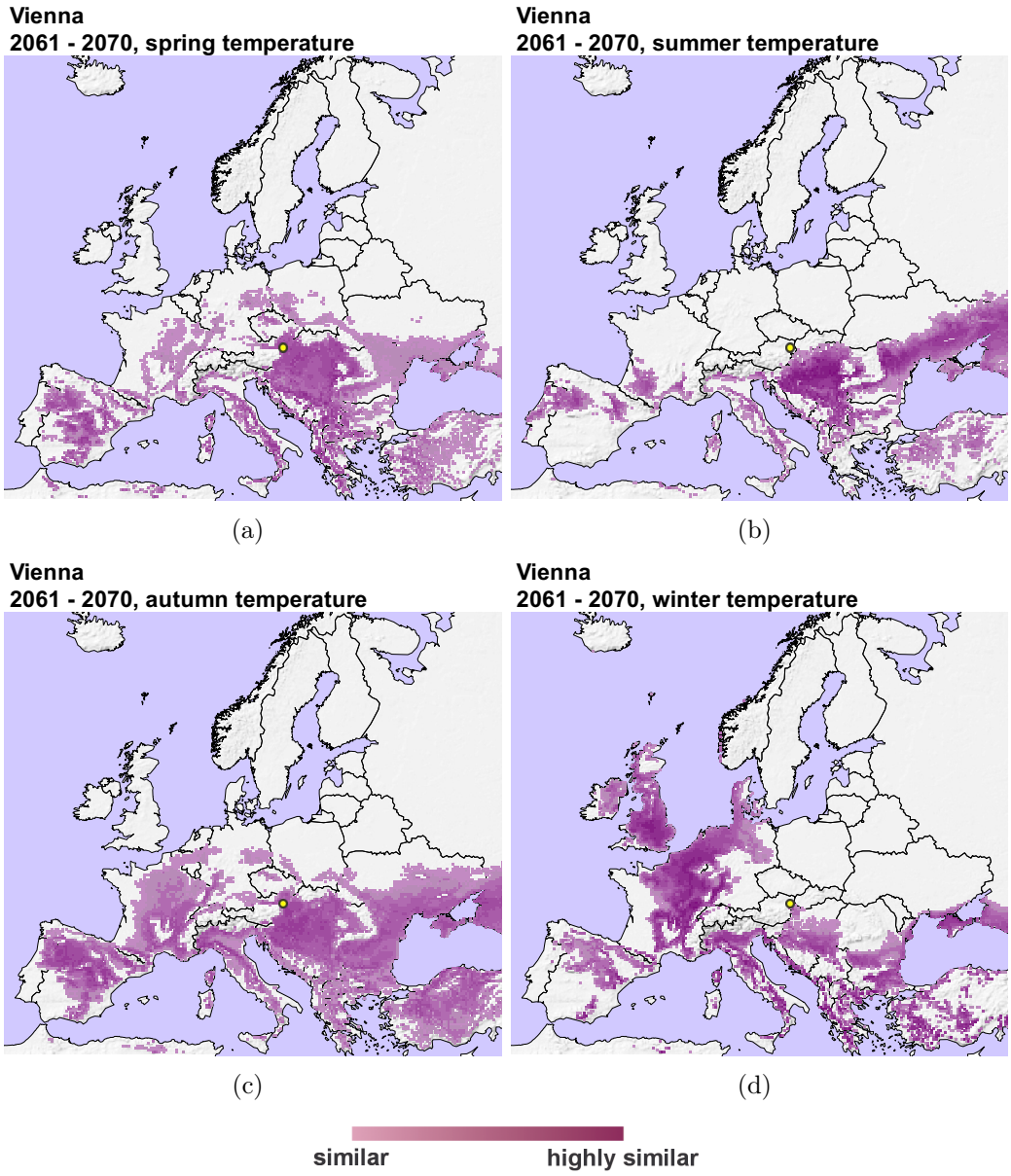


Figure 5.6: Regions with similar current seasonal temperature patterns compared to Vienna 2061 - 2070

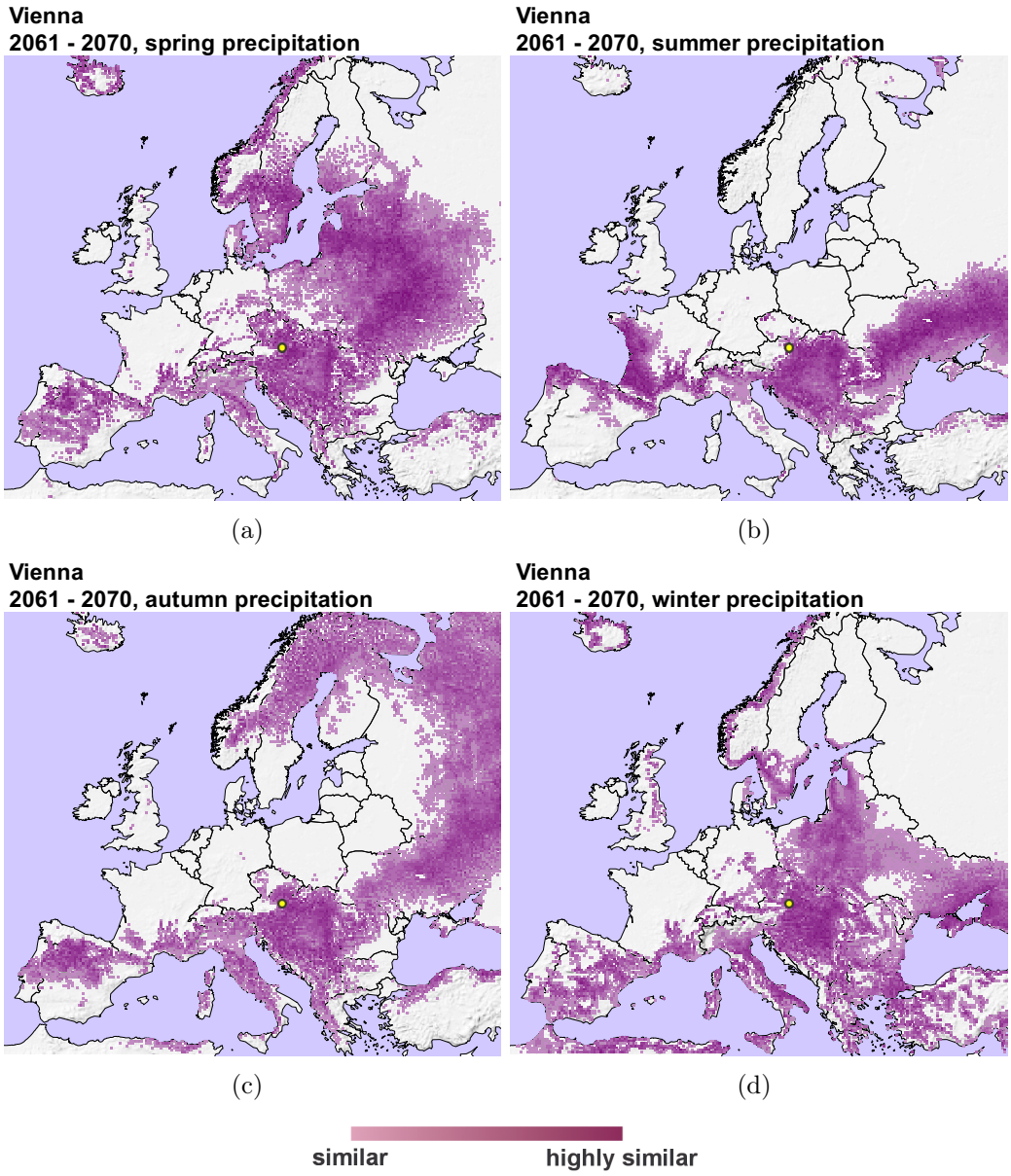


Figure 5.7: Regions with similar current seasonal precipitation patterns compared to Vienna 2061 - 2070

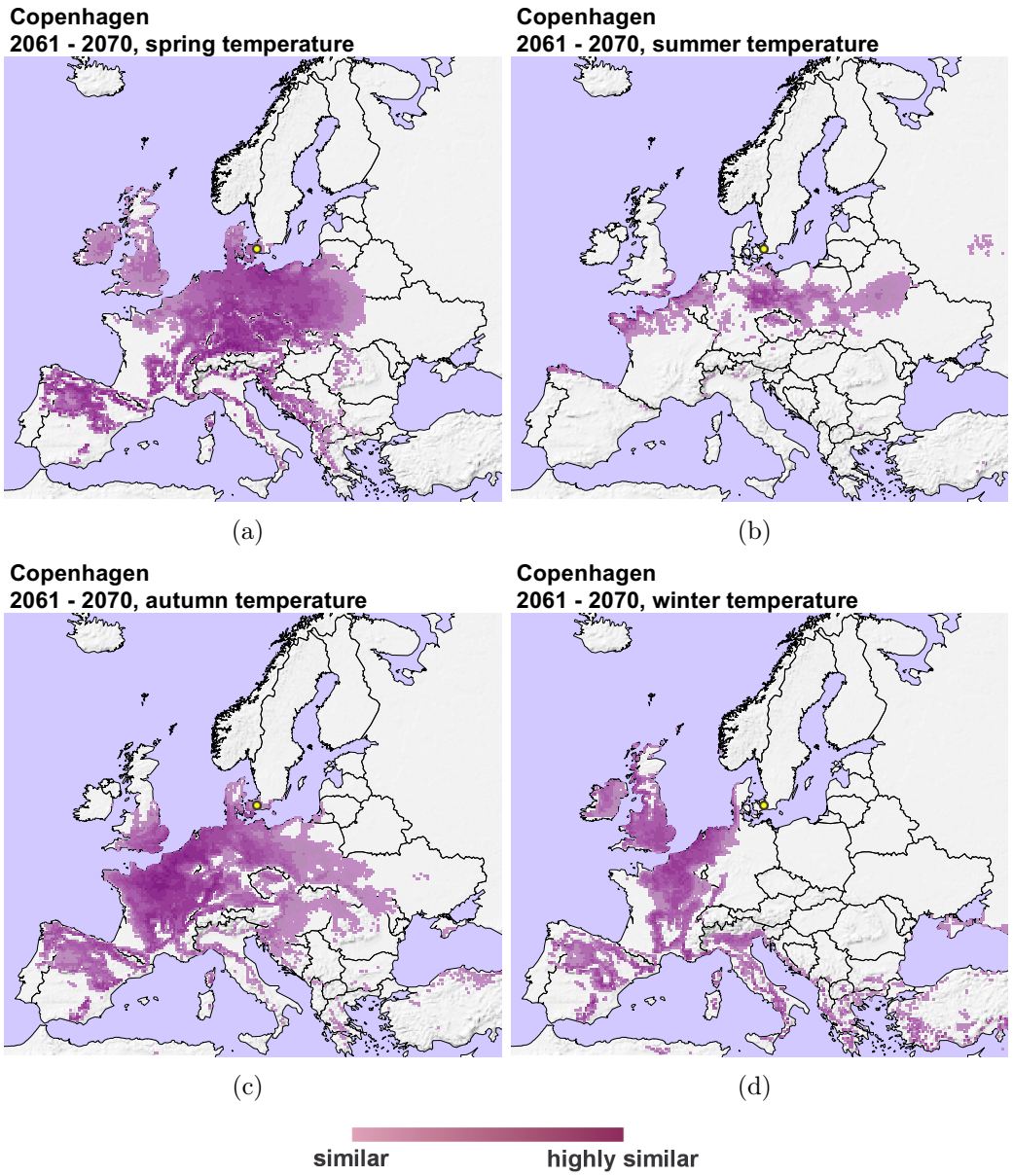


Figure 5.8: Regions with similar current seasonal temperature patterns compared to Copenhagen 2061 - 2070

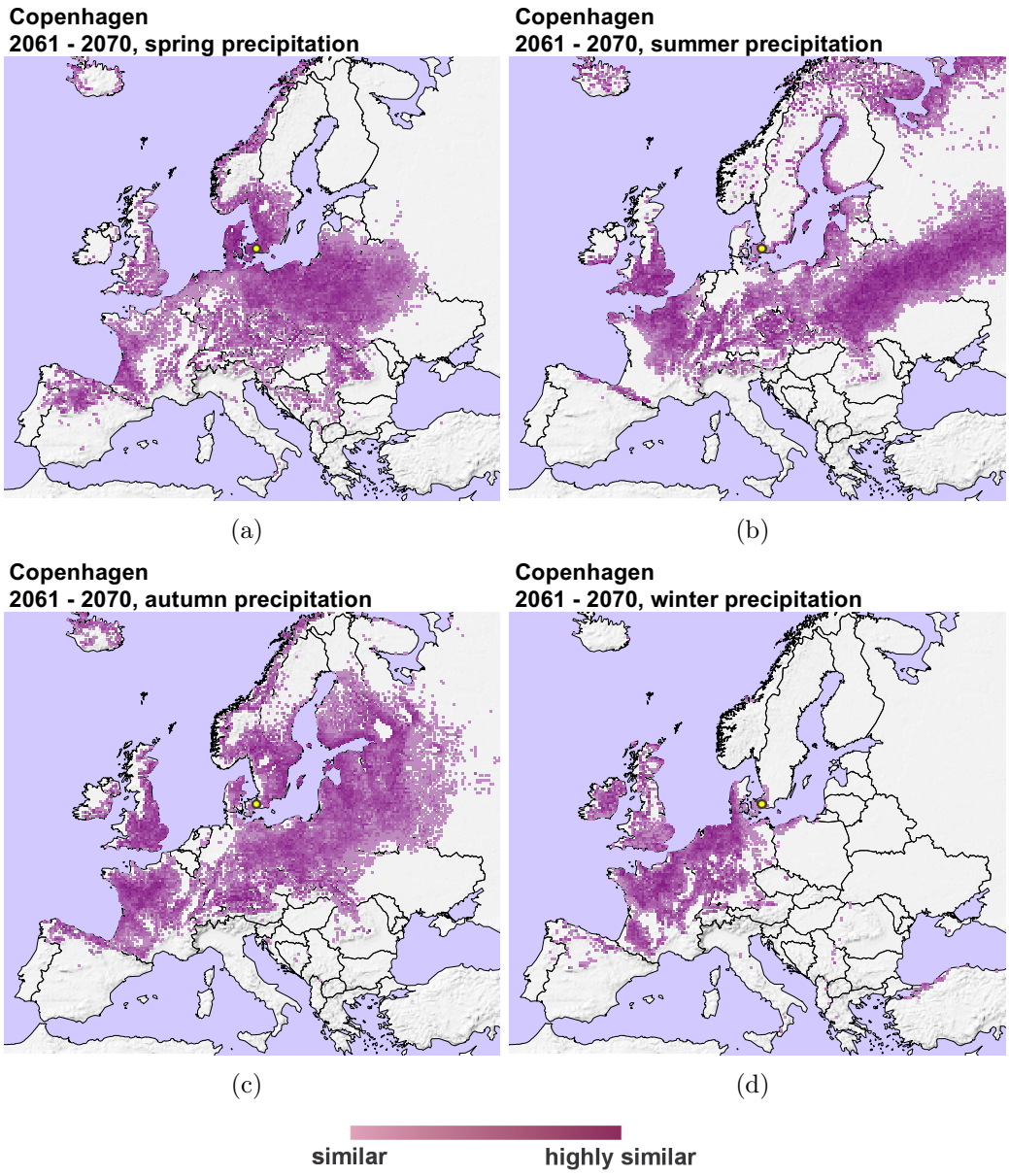


Figure 5.9: Regions with similar current seasonal precipitation patterns compared to Copenhagen 2061 - 2070

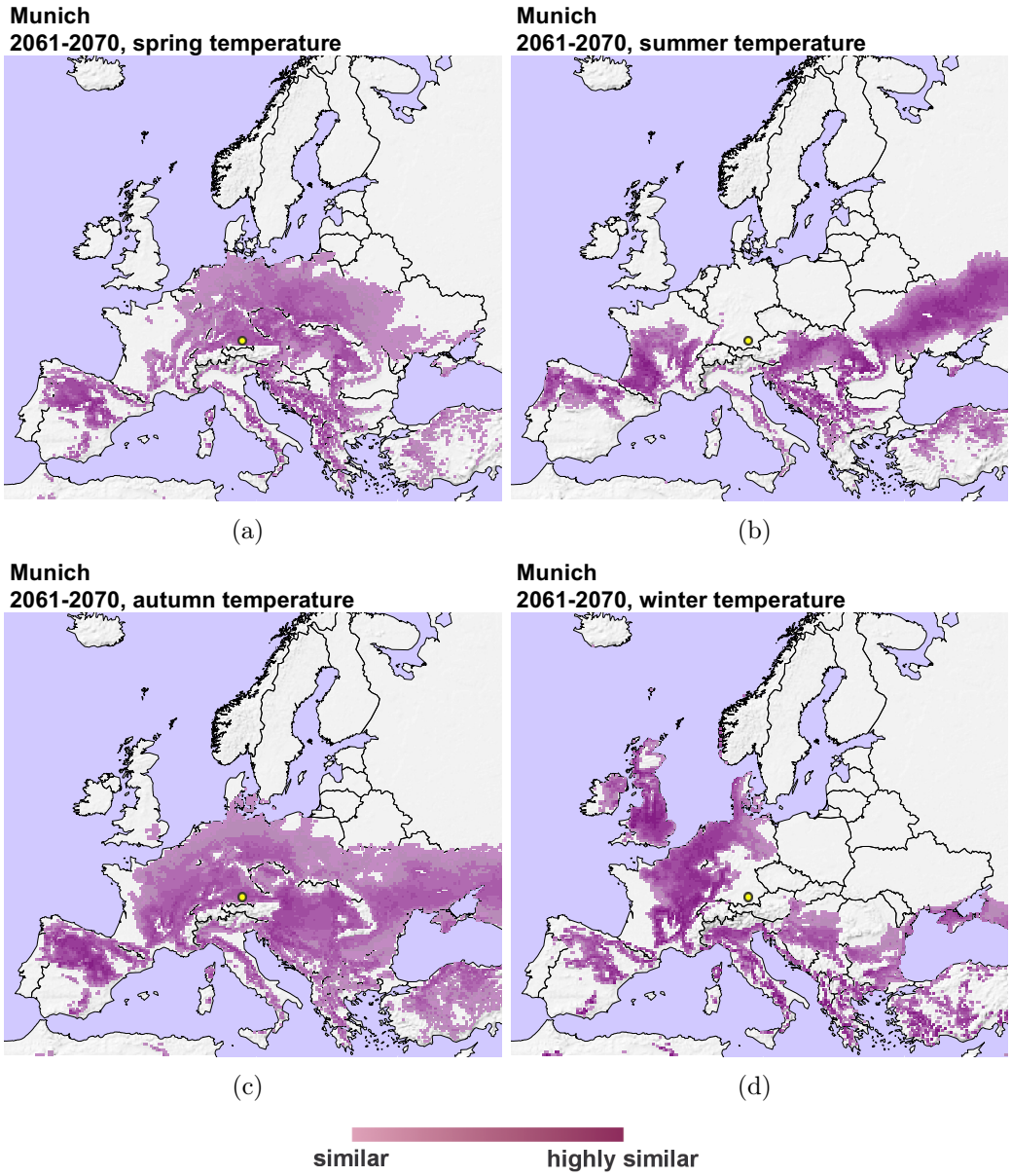


Figure 5.10: Regions with similar current seasonal temperature patterns compared to Munich 2061 - 2070

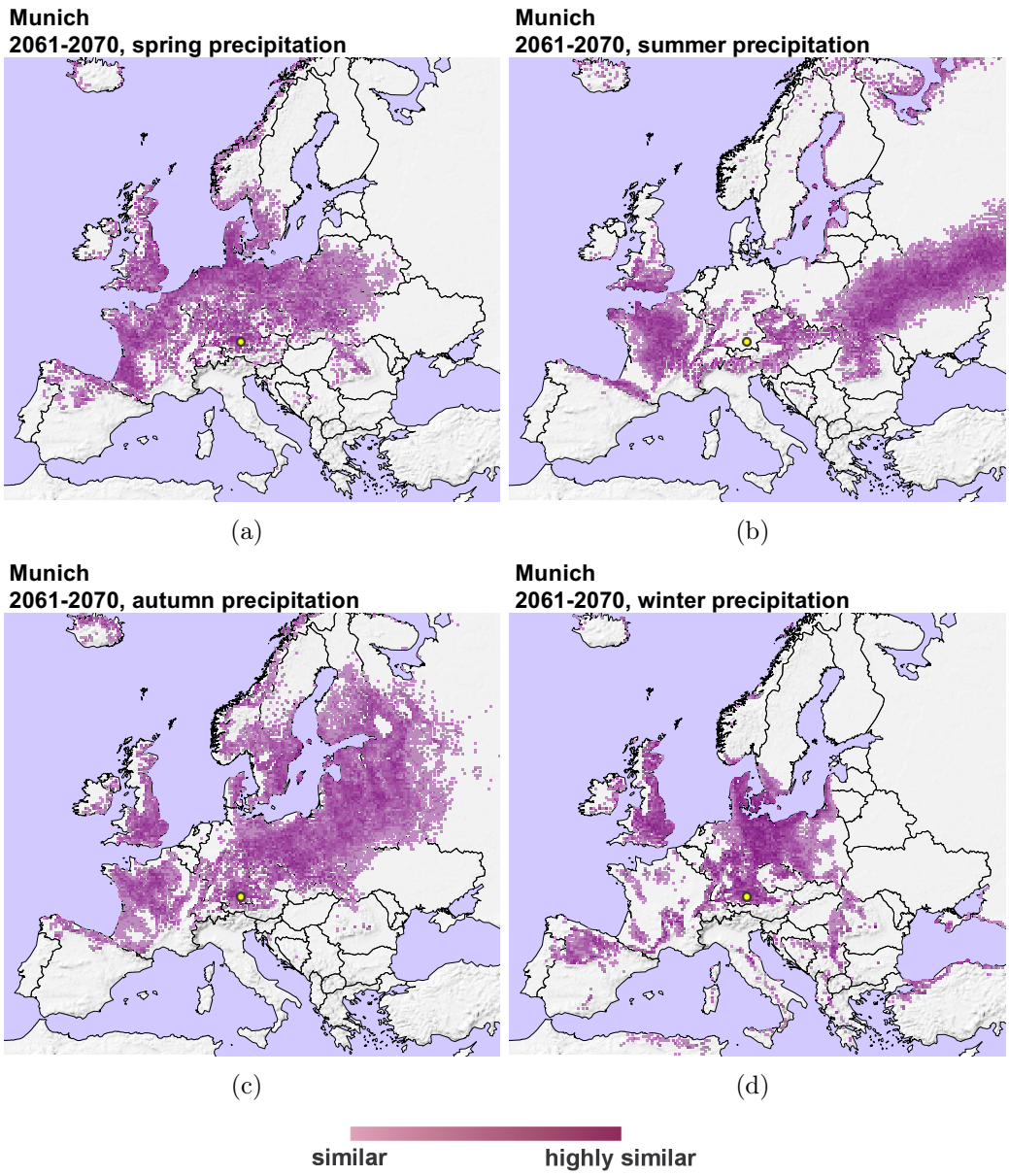


Figure 5.11: Regions with similar current seasonal precipitation patterns compared to Munich 2061 - 2070

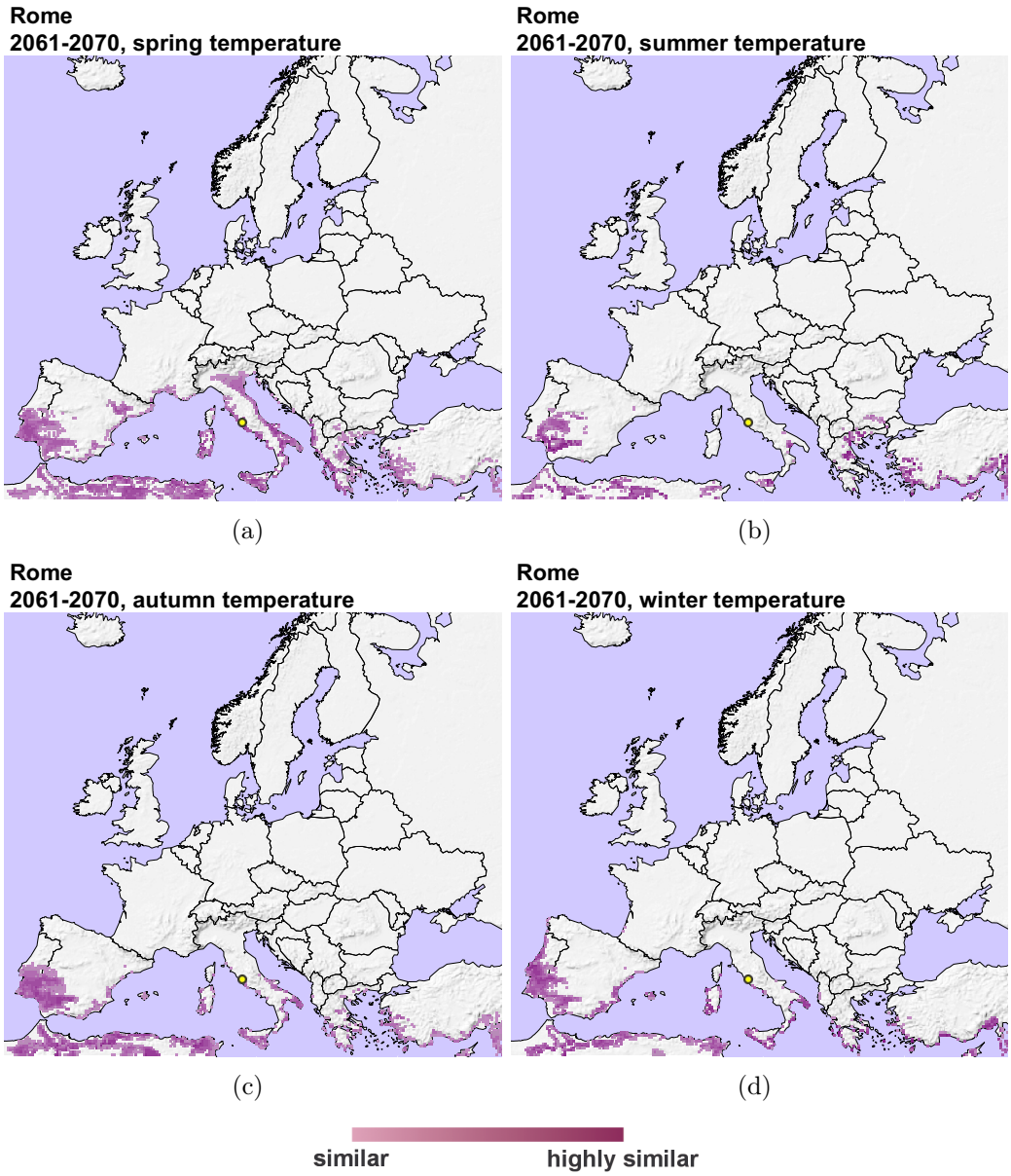


Figure 5.12: Regions with similar current seasonal temperature patterns compared to Rome 2061 - 2070

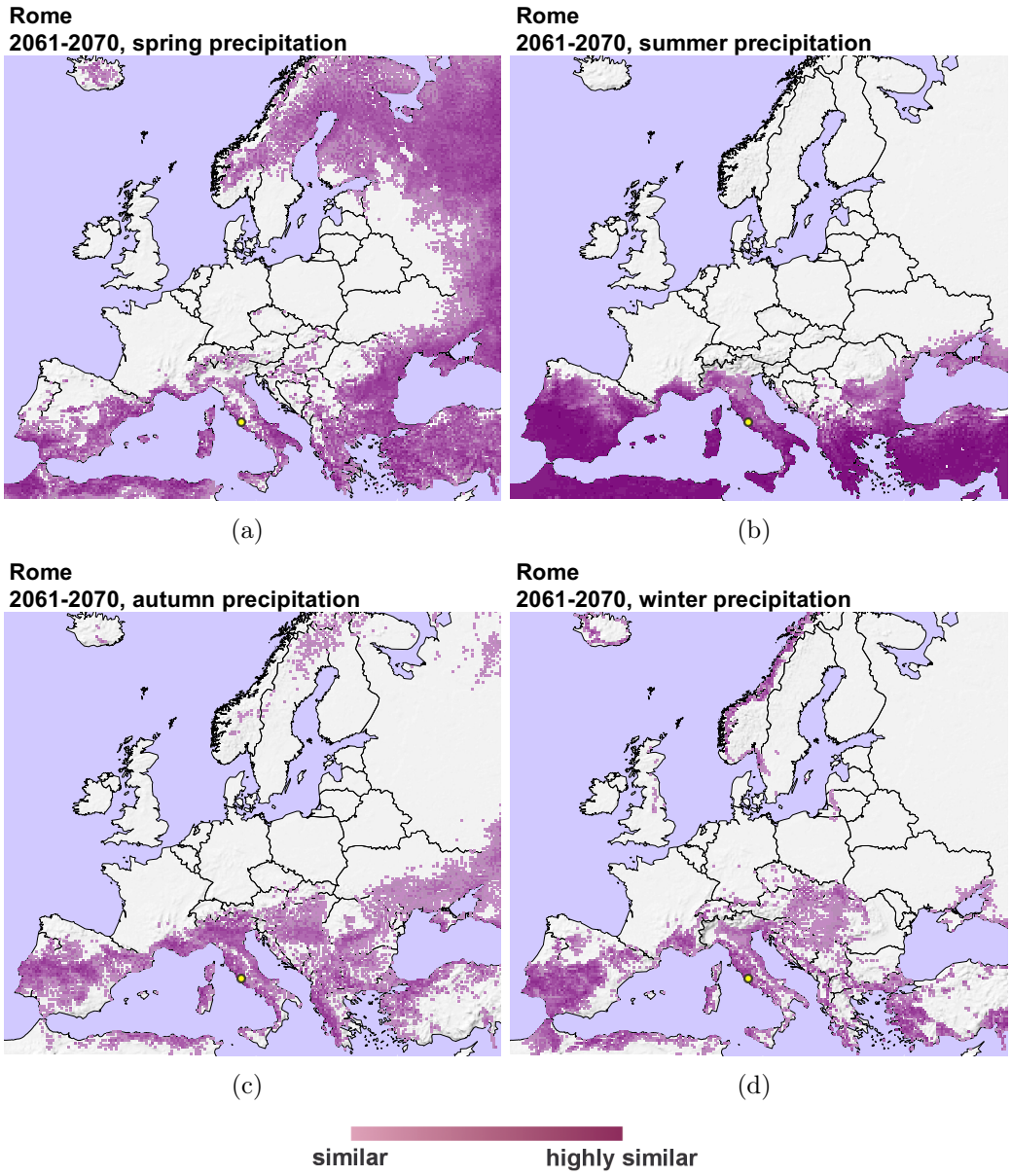


Figure 5.13: Regions with similar current seasonal precipitation patterns compared to Rome 2061 - 2070

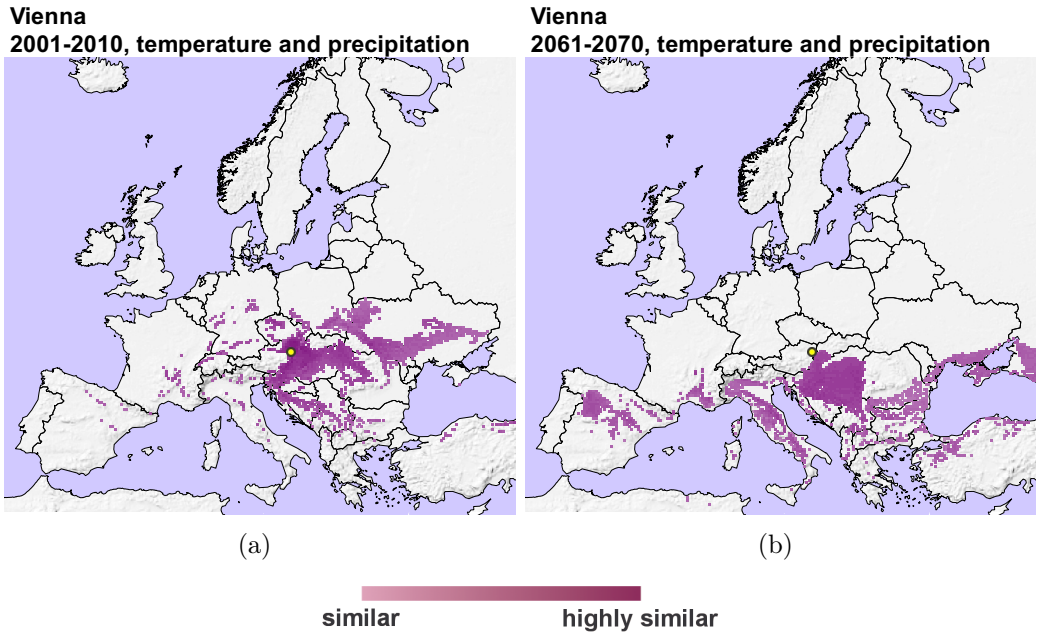


Figure 5.14: Vienna's Climate Twins now and in the period 2061 - 2070 (threshold temperature: 0.75, threshold precipitation: 0.85, PD)

time span.

For the other three examples the default thresholds do not work that well. Copenhagen has a huge future Climate Twin area covering a region between northern England to Southern France with parts in northern Spain, central Italy, Greece, Turkey and even northern parts of Algeria. Munich and Rome in contrast have almost no future Climate Twin Regions except some smaller ones in central France (Munich) and the southern border region between Portugal and Spain (Rome). One reason could be that these points have a very unique climate situation which is hard to find in Europe. Broadening the thresholds would generate more Climate Twin Regions but the threshold or uncertainty width is always related indirectly proportionally with accuracy and consequently the relevance of the result.

5.1.3 Proportional similarity vs. Hellinger Coefficient

Basically the results (Figure 5.1 and Figure 5.15) agree between both used similarity measures PD and r_{H}. Using r_{H} requires much higher thresholds of 0.95 for temperature and 0.97 for precipitation to get a more or less similar amount of Climate Twin cells. This was expected from the evaluation of the similarity measures shown in Section 3.2. As there are similar results and the PD seems to meet the requirements discussed in Subsection 3.2.2, the default measure was used for further result evaluation.

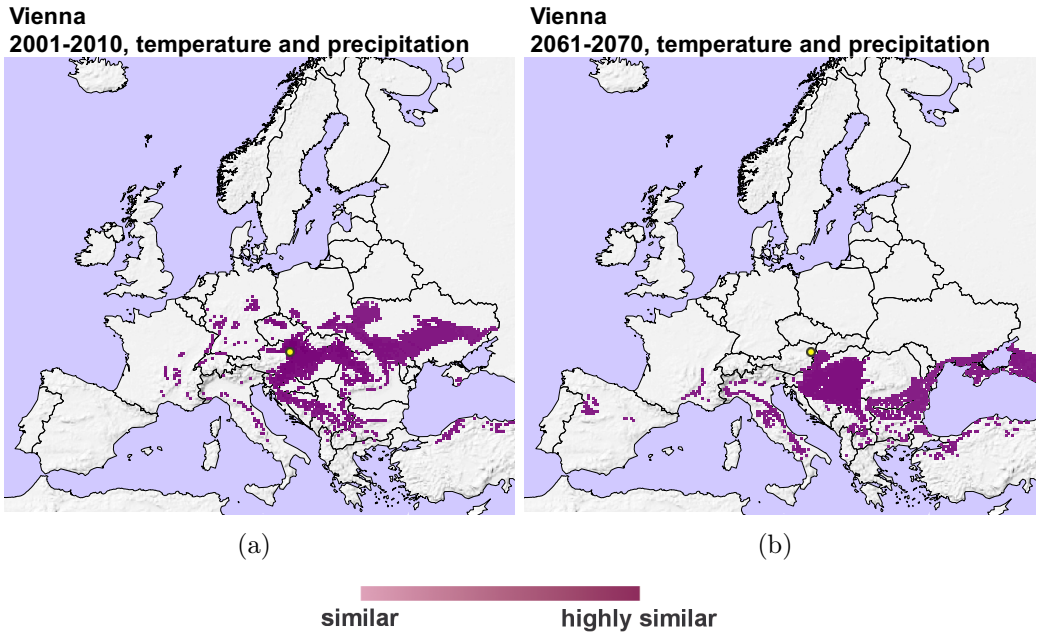


Figure 5.15: Vienna's Climate Twins now and in the period 2061 - 2070 (threshold temperature: 0.95, threshold precipitation: 0.97, r_{H})

5.2 Final statement

5.2.1 Discussion

This work evaluated the possibilities to quantify similarities between regions according to their climate conditions by measuring the similarities between statistical distributions. As there were no research projects covering a similar topic found in the literature it can be seen as a first exploration of cautious steps towards an algorithm to seek similar entities of a source entity in a complex data structure with the help of statistical similarity measures.

The first problem was to find parameters describing climate conditions which are suitable to be implemented in the Climate Twins application. Normally climate is defined by a number of climate variables like air temperature and pressure, wind or precipitation averaged over a significant timespan. Mostly a period of 30 years is covered. In this work 10 year datasets are used as a compromise between a significant minimum period and an adequate number of decadal datasets (14 altogether) to test the method and its results.

The climate indicators used are the daily mean temperatures and daily precipitation sums. The usage of daily data was expected to be more accurate than monthly or yearly mean data. The consistent spatio-temporal resolution of climate model data made the usage of daily data possible. The reason not to use an even finer temporal resolution like hourly data were the challenges of handling it within a common PC environment. More climate indicators like the diurnal temperature amplitude were left aside because daily mean temperatures and precipitation sums are the most used indicators to describe climate. A well-known example are the

almost omnipresent Walter-Lieth climate diagrams. As the aim of the work was to elaborate an example method, the two most important indicators were used to reduce the complexity and to focus on its fundamental applicability. The method's structure allows to implement more parameters, though.

The next step was to find suitable similarity measures. Common tests of descriptive statistics were not applicable because none of them could fulfill the essential requirements. Two almost identical working methods of measuring the similarity between two statistical distributions were found in the library of social sciences. The Proportional Similarity (PD) and the Hellinger Coefficient (r_{H}) compute a normalized value between 0 (not at all similar) and 1 (identical) which is perfect as it can be easily implemented in a fully automated process. Furthermore, statements like "region A is more similar to C than B is to C" can be made so the similarity values can be brought to an ordinal scale. In addition these values can be derived from several unlike indicators and be combined to an overall similarity measure.

It is a major advantage in using statistical distributions rather than derivations like mean values, ranges, deviations, etc. to parametrize climate indicators as these derivations are implicitly included. For example using only the monthly mean temperature to compare two regions would mean to ignore possible differences in the minimum and maximum values or having four days where in one region there is 10 mm precipitation per day and in the other 40 mm in the first day and the other three are dry. The daily mean precipitation occurs as the same although there are completely different climate conditions. A statistical distribution is a more comprehensive way to characterize a dataset.

There are also disadvantages. Most of all, the temporal information of the indicator (its variation through time) will be lost. Therefore the data subsets compared at the lowest level are not yearly distributions but seasonal distributions. With this solution at least the seasonal distributions of the climate indicators are taken into account. Another interesting outcome was that though the temperature distribution worked well, the precipitation comparison showed problems because a precipitation sum distribution's shape is not as characteristic as a temperature's shape and therefore produces higher similarity values. Yet, the application of the similarity measures on precipitation data is not quite satisfying. Other possible indicators have not been tested yet beyond moving averages over sums and applying a logarithmic filter on precipitation distribution.

The method was implemented by programming the algorithm in Java for the existing Climate Twins web application. The results showed that the method produces conclusive results although it strongly depends on the selected thresholds. It was not possible to completely validate the results against existing data or similar research results. Comparing it with climate classification systems would not have made sense because the fundamental idea in the background and the approach to combine similar climates within a predefined framework is a different one than to seek similar entities on the basis of characteristics of a given entity. Further research should be done to confirm the applicability or to point out serious methodological errors committed here.

5.2.2 Outlook

The Climate Twins application can be seen as an educational tool showing the mostly non-transparent process of preparing data before generating results and visualizing them with the help of maps or graphs. Giving the user the choice of specifying thresholds and measures should ensure that he/she will be aware of the method's fragility and its direct impact on the final result. In a world where an unprecedented amount of preprocessed information is available, a sort of "literacy" in the interpretation of statistics, graphs or maps seems to be an important skill. Most of the time preprocessed information regarding complex topics is being accepted by the public.

On the technical side, the IT infrastructure is yet more basic and could be improved to extend the accuracy, usability and calculation speed. The currently used structure with a common PostgreSQL database in the background and a Java program conducting the actual calculations is not the most effective way to realize this project. In the current version it takes approximately one minute to produce the query result and as long as there are possibilities to shorten the retrieval time it should be done. One possibility would be to implement a multidimensional data cube, where all of the daily data is stored for every point—up to now just the absolute frequencies are stored to optimize the processing time—and where the calculations run, rather than using Java. Such a cube can be designed to handle huge amounts of multidimensional (also spatial) data and is optimized in extracting and calculating data so it could also handle the challenges of this project. Another advantage would be adding other climate indicators and implementing additional functionality like rendering climate diagrams.

Last but not least the method could be used on measured climate data to locate similar climates and analyze the reason for the similarity but it should also be able to handle other kinds of data besides climate data. As long as it is possible to acquire enough data, for example on land usage, employment rates, criminal records, etc. the query for similar entities would work and produce interesting insights. A mayor e.g. could seek other cities with similar economical or demographical characteristics to find out how other cities deal with similar problems.

The challenge though is neither the math nor the programming but the definition and parametrization of the characteristic properties and collecting sufficient and accurate data. Focusing on the example given in this work, it means that even if the maps seem to show plausible results, a clean description of climate—if possible at all—cannot be done by just picking daily mean temperatures and precipitation sums. As Thornthwaite (1948) showed in his critique of the choice of parameters for climate classification without considering the evapotranspiration, emphasis should be put on evaluating meaningful parameters.

All in all there could be some useful applications for this method depending on the research question. Further testing especially in other scientific areas should be done to evaluate the possible potential or point out methodological weaknesses.

A Appendix

A.1 R Scripts

One of the major tools used in this thesis was R, because of it's versatility and flexibility. To make all the calculations for this thesis transparent and traceable, the original scripts used are added in this section.

A.1.1 Similarity Measures

```
# ================================================================ #
# |            similarity measures function for R                | #
# |                                                               | #
# |    tested and designed to compare daily temperature or        | #
# |    daily precipitation distributions of two locations         | #
# |                                                               | #
# |                  by Joachim Ungar 2010                        | #
# ================================================================ #
#                                                                  #
# based on the Proportional Similarity and the Hellinger           #
# Coefficient found in                                             #
#                                                                  #
# Jan Vegelius, Svante Janson, and Folke Johansson, Measures of    #
# similarity between distributions, Quality and Quantity 20,       #
# no. 4 (December 1, 1986): 437-441.                               #
#                                                                  #
# ================================================================ #
sim <- function (x, y, c = 0, rh = FALSE, prec = FALSE, log = FALSE, ma = 0, dyn =
    FALSE, debug = FALSE, min_tem = -30, max_tem = 40, min_pre = 0, max_pre = 100)
    {
  # calculating temperature data
  if (prec == FALSE) {
    # convert celsius to kelvin degrees
    xk <- x + 273.15
    yk <- y + 273.15
    if (dyn == FALSE) {
      # define static boundaries
      low <- min_tem + 273.15
      high <- max_tem + 273.15
    } else {
    # set dynamic boundaries
      low <- min(xk,yk)
      high <- max(xk,yk)
    }
  # calculating precipitation data
  } else {
    xk <- x
    yk <- y
    if (dyn == FALSE) {
      # define static boundaries
      low <- min_pre
      high <- max_pre
    } else {
      # set dynamic boundaries
      low <- min(xk,yk)
      high <- max(xk,yk)
    }
  }
  # cumulated frequency
  xk_cum <- 0
  yk_cum <- 0
  # absolute frequency
  xk_abs <- 0
  yk_abs <- 0
  # log of absolute frequency
  xk_abs_log <- 0
  yk_abs_log <- 0
  # relative frequency
```

```
  xk_rel <- 0
  yk_rel <- 0
  # current cumulated frequency
  xk_old <- 0
  yk_old <- 0
  # minimum relative frequency
  sim_min <- 0
  # r value
  sim <- 0
  # temporary value for Hellinger Coefficient
  sim_cat <- 0
  # apply optional moving average filter
  if (ma != 0) {
    xk <- na.exclude(filter(xk, rep(1/ma,ma), sides=2))
    yk <- na.exclude(filter(yk, rep(1/ma,ma), sides=2))
  }
  # values of category borders
  if (prec == TRUE) {
    category_borders <- c(c(1:9),c(2:20)*5)
    c <- length(category_borders)
  } else {
    category_borders <- 0
    # define category width
    category_width <- (high-low)/c
  }
  # other debugging variables
  xsim <- NULL
  ysim <- NULL
  #  check data and calculate if valid
  if (min(xk,yk)<low) {
    print("Error - minimum value out of bounds")
  } else {
    if (max(xk,yk)>high) {
      print("Error - maximum value out of bounds")
    } else {
      i <- 1
      # set cumulated frequency
      while(i<c+1) {
        if (prec == FALSE) {
          xk_cum[i] <- length(xk[xk<(low+i*category_width)])
          yk_cum[i] <- length(yk[yk<(low+i*category_width)])
          # debug category borders
          category_borders[i] <- (low-category_width+i*category_width)-273.15
        } else {
          xk_cum[i] <- length(xk[xk<(category_borders[i])])
          yk_cum[i] <- length(yk[yk<(category_borders[i])])
        }
        # set absolute frequency
        if (i == 1) {
          xk_abs[i] <- xk_cum[i]
          yk_abs[i] <- yk_cum[i]
        } else {
          xk_abs[i] <- xk_cum[i] - xk_old
          yk_abs[i] <- yk_cum[i] - yk_old
        }
        if (log == TRUE) {
          if (xk_abs[i] != 0) {
            xk_abs_log[i] <- log(xk_abs[i])
          } else {
            xk_abs_log[i] <- 0
          }
          if (yk_abs[i] != 0) {
            yk_abs_log[i] <- log(yk_abs[i])
          } else {
            yk_abs_log[i] <- 0
          }
        }
        # save current cumulated frequency to derive absolute frequency in next
            loop
        xk_old <- xk_cum[i]
```

```
        yk_old <- yk_cum[i]
        i <- i+1
      }
      i <- 1
      while (i<c+1) {
        if (log == TRUE) {
          # set relative frequency and apply optional log filter
          xk_rel[i] <- 100 * xk_abs_log[i] / sum(log(xk_abs[xk_abs != 0]))
          yk_rel[i] <- 100 * yk_abs_log[i] / sum(log(yk_abs[yk_abs != 0]))
        } else {
          # set relative frequency
          xk_rel[i] <- 100 * xk_abs[i] / length(xk)
          yk_rel[i] <- 100 * yk_abs[i] / length(yk)
        }
        xsim <- xsim + xk_rel[i]
        ysim <- ysim + yk_rel[i]
        # chose between Proportional Similarity and Hellinger Coefficient
        if (rh == FALSE) {
          # pick minimum relative frequency from x and y's current category
          # (the core calculation of the Proportional Similarity)
          sim_min[i] <- min(xk_rel[i],yk_rel[i])
          sim <- sim + sim_min[i]
        } else {
          # square root of the x's and y's relative frequencies product
          # (the core calculation of the Hellinger Coefficient)
         sim_cat[i] <- sqrt(xk_rel[i] * yk_rel[i])
         sim <- sim + sim_cat[i]
        }
        i <- i+1
      }
      # print r value (sim) with or without debugging information
      if (debug == TRUE) {
        return(list(sim, xk_abs, yk_abs, xk_abs_log, yk_abs_log, xk_rel, yk_rel,
            category_borders, sum(xk_abs_log), sum(yk_abs_log)))
      } else {
        return(sim/100)
      }
    }
  }
}
```

A.1.2 CPU intensive calculations

```
# ============================================================================
# read input files
# ============================================================================

## data
climate <- read.csv(file="csv/rr_tm_day_2001_2010.csv")

## functions
source("csv/similarity.r")

# ============================================================================
# set variables
# ============================================================================

## precipitation
vie_pre <- climate$vie_pre
cop_pre <- climate$cop_pre
mun_pre <- climate$mun_pre
rom_pre <- climate$rom_pre
sp_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("3","4","5"))
sp_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("3","4","5"))
sp_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("3","4","5"))
sp_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("3","4","5"))
su_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("6","7","8"))
```

```
su_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("6","7","8"))
su_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("6","7","8"))
su_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("6","7","8"))
au_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("9","10","11"))
au_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("9","10","11"))
au_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("9","10","11"))
au_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("9","10","11"))
wi_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("12","1","2"))
wi_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("12","1","2"))
wi_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("12","1","2"))
wi_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("12","1","2"))

## temperature
vie_tem <- climate$vie_tem
cop_tem <- climate$cop_tem
mun_tem <- climate$mun_tem
rom_tem <- climate$rom_tem
sp_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("3","4","5"))
sp_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("3","4","5"))
sp_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("3","4","5"))
sp_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("3","4","5"))
su_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("6","7","8"))
su_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("6","7","8"))
su_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("6","7","8"))
su_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("6","7","8"))
au_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("9","10","11"))
au_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("9","10","11"))
au_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("9","10","11"))
au_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("9","10","11"))
wi_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("12","1","2"))
wi_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("12","1","2"))
wi_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("12","1","2"))
wi_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("12","1","2"))

## calculated variables
pd_vie_cop <- NULL
pd_vie_mun <- NULL
pd_vie_rom <- NULL
pd_cop_mun <- NULL
pd_cop_rom <- NULL
pd_mun_rom <- NULL
### spring
sp_pd_vie_cop <- NULL
sp_pd_vie_mun <- NULL
sp_pd_vie_rom <- NULL
sp_pd_cop_mun <- NULL
sp_pd_cop_rom <- NULL
sp_pd_mun_rom <- NULL
### summer
su_pd_vie_cop <- NULL
su_pd_vie_mun <- NULL
su_pd_vie_rom <- NULL
su_pd_cop_mun <- NULL
su_pd_cop_rom <- NULL
su_pd_mun_rom <- NULL
### autumn
au_pd_vie_cop <- NULL
au_pd_vie_mun <- NULL
au_pd_vie_rom <- NULL
au_pd_cop_mun <- NULL
au_pd_cop_rom <- NULL
au_pd_mun_rom <- NULL
### winter
wi_pd_vie_cop <- NULL
wi_pd_vie_mun <- NULL
wi_pd_vie_rom <- NULL
wi_pd_cop_mun <- NULL
wi_pd_cop_rom <- NULL
wi_pd_mun_rom <- NULL
## rh
```

```
rh_vie_cop <- NULL
rh_vie_mun <- NULL
rh_vie_rom <- NULL
rh_cop_mun <- NULL
rh_cop_rom <- NULL
rh_mun_rom <- NULL
### spring
sp_rh_vie_cop <- NULL
sp_rh_vie_mun <- NULL
sp_rh_vie_rom <- NULL
sp_rh_cop_mun <- NULL
sp_rh_cop_rom <- NULL
sp_rh_mun_rom <- NULL
### summer
su_rh_vie_cop <- NULL
su_rh_vie_mun <- NULL
su_rh_vie_rom <- NULL
su_rh_cop_mun <- NULL
su_rh_cop_rom <- NULL
su_rh_mun_rom <- NULL
### autumn
au_rh_vie_cop <- NULL
au_rh_vie_mun <- NULL
au_rh_vie_rom <- NULL
au_rh_cop_mun <- NULL
au_rh_cop_rom <- NULL
au_rh_mun_rom <- NULL
### winter
wi_rh_vie_cop <- NULL
wi_rh_vie_mun <- NULL
wi_rh_vie_rom <- NULL
wi_rh_cop_mun <- NULL
wi_rh_cop_rom <- NULL
wi_rh_mun_rom <- NULL
## output variable
r_cat1000 <- NULL

# ============================================================================
# calculate
# ============================================================================

## r values vs. categories

### pd
for (i in 1:1000) { pd_vie_cop[i] <- sim(vie_tem,cop_tem,i) }
for (i in 1:1000) { pd_vie_mun[i] <- sim(vie_tem,mun_tem,i) }
for (i in 1:1000) { pd_vie_rom[i] <- sim(vie_tem,rom_tem,i) }
for (i in 1:1000) { pd_cop_mun[i] <- sim(cop_tem,mun_tem,i) }
for (i in 1:1000) { pd_cop_rom[i] <- sim(cop_tem,rom_tem,i) }
for (i in 1:1000) { pd_mun_rom[i] <- sim(mun_tem,rom_tem,i) }
#### spring
for (i in 1:1000) { sp_pd_vie_cop[i] <- sim(sp_vie_tem,sp_cop_tem,i) }
for (i in 1:1000) { sp_pd_vie_mun[i] <- sim(sp_vie_tem,sp_mun_tem,i) }
for (i in 1:1000) { sp_pd_vie_rom[i] <- sim(sp_vie_tem,sp_rom_tem,i) }
for (i in 1:1000) { sp_pd_cop_mun[i] <- sim(sp_cop_tem,sp_mun_tem,i) }
for (i in 1:1000) { sp_pd_cop_rom[i] <- sim(sp_cop_tem,sp_rom_tem,i) }
for (i in 1:1000) { sp_pd_mun_rom[i] <- sim(sp_mun_tem,sp_rom_tem,i) }
#### summer
for (i in 1:1000) { su_pd_vie_cop[i] <- sim(su_vie_tem,su_cop_tem,i) }
for (i in 1:1000) { su_pd_vie_mun[i] <- sim(su_vie_tem,su_mun_tem,i) }
for (i in 1:1000) { su_pd_vie_rom[i] <- sim(su_vie_tem,su_rom_tem,i) }
for (i in 1:1000) { su_pd_cop_mun[i] <- sim(su_cop_tem,su_mun_tem,i) }
for (i in 1:1000) { su_pd_cop_rom[i] <- sim(su_cop_tem,su_rom_tem,i) }
for (i in 1:1000) { su_pd_mun_rom[i] <- sim(su_mun_tem,su_rom_tem,i) }
#### autumn
for (i in 1:1000) { au_pd_vie_cop[i] <- sim(au_vie_tem,au_cop_tem,i) }
for (i in 1:1000) { au_pd_vie_mun[i] <- sim(au_vie_tem,au_mun_tem,i) }
for (i in 1:1000) { au_pd_vie_rom[i] <- sim(au_vie_tem,au_rom_tem,i) }
for (i in 1:1000) { au_pd_cop_mun[i] <- sim(au_cop_tem,au_mun_tem,i) }
for (i in 1:1000) { au_pd_cop_rom[i] <- sim(au_cop_tem,au_rom_tem,i) }
```

```
for (i in 1:1000) { au_pd_mun_rom[i] <- sim(au_mun_tem,au_rom_tem,i) }
#### winter
for (i in 1:1000) { wi_pd_vie_cop[i] <- sim(wi_vie_tem,wi_cop_tem,i) }
for (i in 1:1000) { wi_pd_vie_mun[i] <- sim(wi_vie_tem,wi_mun_tem,i) }
for (i in 1:1000) { wi_pd_vie_rom[i] <- sim(wi_vie_tem,wi_rom_tem,i) }
for (i in 1:1000) { wi_pd_cop_mun[i] <- sim(wi_cop_tem,wi_mun_tem,i) }
for (i in 1:1000) { wi_pd_cop_rom[i] <- sim(wi_cop_tem,wi_rom_tem,i) }
for (i in 1:1000) { wi_pd_mun_rom[i] <- sim(wi_mun_tem,wi_rom_tem,i) }

### rh
for (i in 1:1000) { rh_vie_cop[i] <- sim(rh=TRUE,vie_tem,cop_tem,i) }
for (i in 1:1000) { rh_vie_mun[i] <- sim(rh=TRUE,vie_tem,mun_tem,i) }
for (i in 1:1000) { rh_vie_rom[i] <- sim(rh=TRUE,vie_tem,rom_tem,i) }
for (i in 1:1000) { rh_cop_mun[i] <- sim(rh=TRUE,cop_tem,mun_tem,i) }
for (i in 1:1000) { rh_cop_rom[i] <- sim(rh=TRUE,cop_tem,rom_tem,i) }
for (i in 1:1000) { rh_mun_rom[i] <- sim(rh=TRUE,mun_tem,rom_tem,i) }
#### spring
for (i in 1:1000) { sp_rh_vie_cop[i] <- sim(rh=TRUE,sp_vie_tem,sp_cop_tem,i) }
for (i in 1:1000) { sp_rh_vie_mun[i] <- sim(rh=TRUE,sp_vie_tem,sp_mun_tem,i) }
for (i in 1:1000) { sp_rh_vie_rom[i] <- sim(rh=TRUE,sp_vie_tem,sp_rom_tem,i) }
for (i in 1:1000) { sp_rh_cop_mun[i] <- sim(rh=TRUE,sp_cop_tem,sp_mun_tem,i) }
for (i in 1:1000) { sp_rh_cop_rom[i] <- sim(rh=TRUE,sp_cop_tem,sp_rom_tem,i) }
for (i in 1:1000) { sp_rh_mun_rom[i] <- sim(rh=TRUE,sp_mun_tem,sp_rom_tem,i) }
#### summer
for (i in 1:1000) { su_rh_vie_cop[i] <- sim(rh=TRUE,su_vie_tem,su_cop_tem,i) }
for (i in 1:1000) { su_rh_vie_mun[i] <- sim(rh=TRUE,su_vie_tem,su_mun_tem,i) }
for (i in 1:1000) { su_rh_vie_rom[i] <- sim(rh=TRUE,su_vie_tem,su_rom_tem,i) }
for (i in 1:1000) { su_rh_cop_mun[i] <- sim(rh=TRUE,su_cop_tem,su_mun_tem,i) }
for (i in 1:1000) { su_rh_cop_rom[i] <- sim(rh=TRUE,su_cop_tem,su_rom_tem,i) }
for (i in 1:1000) { su_rh_mun_rom[i] <- sim(rh=TRUE,su_mun_tem,su_rom_tem,i) }
#### autumn
for (i in 1:1000) { au_rh_vie_cop[i] <- sim(rh=TRUE,au_vie_tem,au_cop_tem,i) }
for (i in 1:1000) { au_rh_vie_mun[i] <- sim(rh=TRUE,au_vie_tem,au_mun_tem,i) }
for (i in 1:1000) { au_rh_vie_rom[i] <- sim(rh=TRUE,au_vie_tem,au_rom_tem,i) }
for (i in 1:1000) { au_rh_cop_mun[i] <- sim(rh=TRUE,au_cop_tem,au_mun_tem,i) }
for (i in 1:1000) { au_rh_cop_rom[i] <- sim(rh=TRUE,au_cop_tem,au_rom_tem,i) }
for (i in 1:1000) { au_rh_mun_rom[i] <- sim(rh=TRUE,au_mun_tem,au_rom_tem,i) }
#### winter
for (i in 1:1000) { wi_rh_vie_cop[i] <- sim(rh=TRUE,wi_vie_tem,wi_cop_tem,i) }
for (i in 1:1000) { wi_rh_vie_mun[i] <- sim(rh=TRUE,wi_vie_tem,wi_mun_tem,i) }
for (i in 1:1000) { wi_rh_vie_rom[i] <- sim(rh=TRUE,wi_vie_tem,wi_rom_tem,i) }
for (i in 1:1000) { wi_rh_cop_mun[i] <- sim(rh=TRUE,wi_cop_tem,wi_mun_tem,i) }
for (i in 1:1000) { wi_rh_cop_rom[i] <- sim(rh=TRUE,wi_cop_tem,wi_rom_tem,i) }
for (i in 1:1000) { wi_rh_mun_rom[i] <- sim(rh=TRUE,wi_mun_tem,wi_rom_tem,i) }

# ============================================================================
# save and output
# ============================================================================

r_cat1000$x <- c(1:1000)
## pd
r_cat1000$pd_vie_cop <- pd_vie_cop
r_cat1000$pd_vie_mun <- pd_vie_mun
r_cat1000$pd_vie_rom <- pd_vie_rom
r_cat1000$pd_cop_mun <- pd_cop_mun
r_cat1000$pd_cop_rom <- pd_cop_rom
r_cat1000$pd_mun_rom <- pd_mun_rom
### spring
r_cat1000$sp_pd_vie_cop <- sp_pd_vie_cop
r_cat1000$sp_pd_vie_mun <- sp_pd_vie_mun
r_cat1000$sp_pd_vie_rom <- sp_pd_vie_rom
r_cat1000$sp_pd_cop_mun <- sp_pd_cop_mun
r_cat1000$sp_pd_cop_rom <- sp_pd_cop_rom
r_cat1000$sp_pd_mun_rom <- sp_pd_mun_rom
### summer
r_cat1000$su_pd_vie_cop <- su_pd_vie_cop
r_cat1000$su_pd_vie_mun <- su_pd_vie_mun
r_cat1000$su_pd_vie_rom <- su_pd_vie_rom
r_cat1000$su_pd_cop_mun <- su_pd_cop_mun
r_cat1000$su_pd_cop_rom <- su_pd_cop_rom
```

```
r_cat1000$su_pd_mun_rom <- su_pd_mun_rom
### autumn
r_cat1000$au_pd_vie_cop <- au_pd_vie_cop
r_cat1000$au_pd_vie_mun <- au_pd_vie_mun
r_cat1000$au_pd_vie_rom <- au_pd_vie_rom
r_cat1000$au_pd_cop_mun <- au_pd_cop_mun
r_cat1000$au_pd_cop_rom <- au_pd_cop_rom
r_cat1000$au_pd_mun_rom <- au_pd_mun_rom
### winter
r_cat1000$wi_pd_vie_cop <- wi_pd_vie_cop
r_cat1000$wi_pd_vie_mun <- wi_pd_vie_mun
r_cat1000$wi_pd_vie_rom <- wi_pd_vie_rom
r_cat1000$wi_pd_cop_mun <- wi_pd_cop_mun
r_cat1000$wi_pd_cop_rom <- wi_pd_cop_rom
r_cat1000$wi_pd_mun_rom <- wi_pd_mun_rom

## rh
r_cat1000$rh_vie_cop <- rh_vie_cop
r_cat1000$rh_vie_mun <- rh_vie_mun
r_cat1000$rh_vie_rom <- rh_vie_rom
r_cat1000$rh_cop_mun <- rh_cop_mun
r_cat1000$rh_cop_rom <- rh_cop_rom
r_cat1000$rh_mun_rom <- rh_mun_rom
### spring
r_cat1000$sp_rh_vie_cop <- sp_rh_vie_cop
r_cat1000$sp_rh_vie_mun <- sp_rh_vie_mun
r_cat1000$sp_rh_vie_rom <- sp_rh_vie_rom
r_cat1000$sp_rh_cop_mun <- sp_rh_cop_mun
r_cat1000$sp_rh_cop_rom <- sp_rh_cop_rom
r_cat1000$sp_rh_mun_rom <- sp_rh_mun_rom
### summer
r_cat1000$su_rh_vie_cop <- su_rh_vie_cop
r_cat1000$su_rh_vie_mun <- su_rh_vie_mun
r_cat1000$su_rh_vie_rom <- su_rh_vie_rom
r_cat1000$su_rh_cop_mun <- su_rh_cop_mun
r_cat1000$su_rh_cop_rom <- su_rh_cop_rom
r_cat1000$su_rh_mun_rom <- su_rh_mun_rom
### autumn
r_cat1000$au_rh_vie_cop <- au_rh_vie_cop
r_cat1000$au_rh_vie_mun <- au_rh_vie_mun
r_cat1000$au_rh_vie_rom <- au_rh_vie_rom
r_cat1000$au_rh_cop_mun <- au_rh_cop_mun
r_cat1000$au_rh_cop_rom <- au_rh_cop_rom
r_cat1000$au_rh_mun_rom <- au_rh_mun_rom
### winter
r_cat1000$wi_rh_vie_cop <- wi_rh_vie_cop
r_cat1000$wi_rh_vie_mun <- wi_rh_vie_mun
r_cat1000$wi_rh_vie_rom <- wi_rh_vie_rom
r_cat1000$wi_rh_cop_mun <- wi_rh_cop_mun
r_cat1000$wi_rh_cop_rom <- wi_rh_cop_rom
r_cat1000$wi_rh_mun_rom <- wi_rh_mun_rom

write.csv(r_cat1000,file="csv/r_cat.csv")
```

A.1.3 Graph generator

```
# ============================================================================
# read input files
# ============================================================================

## data
climate <- read.csv(file="csv/rr_tm_day_2001_2010.csv")
r_cat <- read.csv(file="csv/r_cat.csv")

## functions
source("csv/similarity.r")

```

```
# ==========================================================================
# set variables
# ==========================================================================

## precipitation
vie_pre <- climate$vie_pre
cop_pre <- climate$cop_pre
mun_pre <- climate$mun_pre
rom_pre <- climate$rom_pre
sp_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("3","4","5"))
sp_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("3","4","5"))
sp_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("3","4","5"))
sp_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("3","4","5"))
su_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("6","7","8"))
su_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("6","7","8"))
su_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("6","7","8"))
su_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("6","7","8"))
au_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("9","10","11"))
au_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("9","10","11"))
au_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("9","10","11"))
au_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("9","10","11"))
wi_vie_pre <- subset(climate$vie_pre, climate$mm %in% c("12","1","2"))
wi_cop_pre <- subset(climate$cop_pre, climate$mm %in% c("12","1","2"))
wi_mun_pre <- subset(climate$mun_pre, climate$mm %in% c("12","1","2"))
wi_rom_pre <- subset(climate$rom_pre, climate$mm %in% c("12","1","2"))

## temperature
vie_tem <- climate$vie_tem
cop_tem <- climate$cop_tem
mun_tem <- climate$mun_tem
rom_tem <- climate$rom_tem
sp_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("3","4","5"))
sp_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("3","4","5"))
sp_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("3","4","5"))
sp_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("3","4","5"))
su_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("6","7","8"))
su_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("6","7","8"))
su_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("6","7","8"))
su_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("6","7","8"))
au_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("9","10","11"))
au_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("9","10","11"))
au_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("9","10","11"))
au_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("9","10","11"))
wi_vie_tem <- subset(climate$vie_tem, climate$mm %in% c("12","1","2"))
wi_cop_tem <- subset(climate$cop_tem, climate$mm %in% c("12","1","2"))
wi_mun_tem <- subset(climate$mun_tem, climate$mm %in% c("12","1","2"))
wi_rom_tem <- subset(climate$rom_tem, climate$mm %in% c("12","1","2"))

## calculated variables
pd_vie_cop <- NULL
pd_vie_mun <- NULL
pd_vie_rom <- NULL
pd_cop_mun <- NULL
pd_cop_rom <- NULL
pd_mun_rom <- NULL
rh_vie_cop <- NULL
rh_vie_mun <- NULL
rh_vie_rom <- NULL
rh_cop_mun <- NULL
rh_cop_rom <- NULL
rh_mun_rom <- NULL

## precipitation sums
vie_pre_sum <- NULL
cop_pre_sum <- NULL
mun_pre_sum <- NULL
rom_pre_sum <- NULL

# ==========================================================================
# calculate
```

```
# ==========================================================================
83
## generate example functions
85 t1 <- c(c(1:900)*1.5,c(1351:3150),c(900:1)*3.5)/43
t2 <- c(c(1:900)*3.5,c(3150:1351),c(900:1)*1.5)/43
87
## moving average filter
89 vie_pre_ma7 <- na.exclude(filter(vie_pre, rep(1/7,7), sides=2))
cop_pre_ma7 <- na.exclude(filter(cop_pre, rep(1/7,7), sides=2))
91 mun_pre_ma7 <- na.exclude(filter(mun_pre, rep(1/7,7), sides=2))
rom_pre_ma7 <- na.exclude(filter(rom_pre, rep(1/7,7), sides=2))
93 ### spring
sp_vie_pre_ma7 <- na.exclude(subset(filter(climate$vie_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("3","4","5")))
95 sp_cop_pre_ma7 <- na.exclude(subset(filter(climate$cop_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("3","4","5")))
sp_mun_pre_ma7 <- na.exclude(subset(filter(climate$mun_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("3","4","5")))
97 sp_rom_pre_ma7 <- na.exclude(subset(filter(climate$rom_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("3","4","5")))
### summer
99 su_vie_pre_ma7 <- na.exclude(subset(filter(climate$vie_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("6","7","8")))
su_cop_pre_ma7 <- na.exclude(subset(filter(climate$cop_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("6","7","8")))
101 su_mun_pre_ma7 <- na.exclude(subset(filter(climate$mun_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("6","7","8")))
su_rom_pre_ma7 <- na.exclude(subset(filter(climate$rom_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("6","7","8")))
103 ### autumn
au_vie_pre_ma7 <- na.exclude(subset(filter(climate$vie_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("9","10","11")))
105 au_cop_pre_ma7 <- na.exclude(subset(filter(climate$cop_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("9","10","11")))
au_mun_pre_ma7 <- na.exclude(subset(filter(climate$mun_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("9","10","11")))
107 au_rom_pre_ma7 <- na.exclude(subset(filter(climate$rom_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("9","10","11")))
### winter
109 wi_vie_pre_ma7 <- na.exclude(subset(filter(climate$vie_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("12","1","2")))
wi_cop_pre_ma7 <- na.exclude(subset(filter(climate$cop_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("12","1","2")))
111 wi_mun_pre_ma7 <- na.exclude(subset(filter(climate$mun_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("12","1","2")))
wi_rom_pre_ma7 <- na.exclude(subset(filter(climate$rom_pre, rep(1/7,7), sides=2),
    climate$mm %in% c("12","1","2")))
113
## combining r values
115 c <- 35
### temperature
117 #### pd
pd_tem_vie_cop <- c(sim(su_vie_tem,su_cop_tem,c),sim(au_vie_tem,au_cop_tem,c),sim(
    wi_vie_tem,wi_cop_tem,c),sim(sp_vie_tem,sp_cop_tem,c))
119 pd_tem_vie_cop[5] <- mean(pd_tem_vie_cop[1:4])
pd_tem_vie_cop[6] <- sim(vie_tem,cop_tem,c)
121 #
pd_tem_vie_mun <- c(sim(su_vie_tem,su_mun_tem,c),sim(au_vie_tem,au_mun_tem,c),sim(
    wi_vie_tem,wi_mun_tem,c),sim(sp_vie_tem,sp_mun_tem,c))
123 pd_tem_vie_mun[5] <- mean(pd_tem_vie_mun[1:4])
pd_tem_vie_mun[6] <- sim(vie_tem,mun_tem,c)
125 #
pd_tem_vie_rom <- c(sim(su_vie_tem,su_rom_tem,c),sim(au_vie_tem,au_rom_tem,c),sim(
    wi_vie_tem,wi_rom_tem,c),sim(sp_vie_tem,sp_rom_tem,c))
127 pd_tem_vie_rom[5] <- mean(pd_tem_vie_rom[1:4])
pd_tem_vie_rom[6] <- sim(vie_tem,rom_tem,c)
129 #
pd_tem_cop_mun <- c(sim(su_cop_tem,su_mun_tem,c),sim(au_cop_tem,au_mun_tem,c),sim(
    wi_cop_tem,wi_mun_tem,c),sim(sp_cop_tem,sp_mun_tem,c))
131 pd_tem_cop_mun[5] <- mean(pd_tem_cop_mun[1:4])
```

```
pd_tem_cop_mun[6] <- sim(cop_tem,mun_tem,c)
#
pd_tem_cop_rom <- c(sim(su_cop_tem,su_rom_tem,c),sim(au_cop_tem,au_rom_tem,c),sim(
    wi_cop_tem,wi_rom_tem,c),sim(sp_cop_tem,sp_rom_tem,c))
pd_tem_cop_rom[5] <- mean(pd_tem_cop_rom[1:4])
pd_tem_cop_rom[6] <- sim(cop_tem,rom_tem,c)
#
pd_tem_mun_rom <- c(sim(su_mun_tem,su_rom_tem,c),sim(au_mun_tem,au_rom_tem,c),sim(
    wi_mun_tem,wi_rom_tem,c),sim(sp_mun_tem,sp_rom_tem,c))
pd_tem_mun_rom[5] <- mean(pd_tem_mun_rom[1:4])
pd_tem_mun_rom[6] <- sim(mun_tem,rom_tem,c)
### rh
rh_tem_vie_cop <- c(sim(rh=TRUE,su_vie_tem,su_cop_tem,c),sim(rh=TRUE,au_vie_tem,au_
    cop_tem,c),sim(rh=TRUE,wi_vie_tem,wi_cop_tem,c),sim(rh=TRUE,sp_vie_tem,sp_cop_
    tem,c))
rh_tem_vie_cop[5] <- mean(rh_tem_vie_cop[1:4])
rh_tem_vie_cop[6] <- sim(rh=TRUE,vie_tem,cop_tem,c)
#
rh_tem_vie_mun <- c(sim(rh=TRUE,su_vie_tem,su_mun_tem,c),sim(rh=TRUE,au_vie_tem,au_
    mun_tem,c),sim(rh=TRUE,wi_vie_tem,wi_mun_tem,c),sim(rh=TRUE,sp_vie_tem,sp_mun_
    tem,c))
rh_tem_vie_mun[5] <- mean(rh_tem_vie_mun[1:4])
rh_tem_vie_mun[6] <- sim(rh=TRUE,vie_tem,mun_tem,c)
#
rh_tem_vie_rom <- c(sim(rh=TRUE,su_vie_tem,su_rom_tem,c),sim(rh=TRUE,au_vie_tem,au_
    rom_tem,c),sim(rh=TRUE,wi_vie_tem,wi_rom_tem,c),sim(rh=TRUE,sp_vie_tem,sp_rom_
    tem,c))
rh_tem_vie_rom[5] <- mean(rh_tem_vie_rom[1:4])
rh_tem_vie_rom[6] <- sim(rh=TRUE,vie_tem,rom_tem,c)
#
rh_tem_cop_mun <- c(sim(rh=TRUE,su_cop_tem,su_mun_tem,c),sim(rh=TRUE,au_cop_tem,au_
    mun_tem,c),sim(rh=TRUE,wi_cop_tem,wi_mun_tem,c),sim(rh=TRUE,sp_cop_tem,sp_mun_
    tem,c))
rh_tem_cop_mun[5] <- mean(rh_tem_cop_mun[1:4])
rh_tem_cop_mun[6] <- sim(rh=TRUE,cop_tem,mun_tem,c)
#
rh_tem_cop_rom <- c(sim(rh=TRUE,su_cop_tem,su_rom_tem,c),sim(rh=TRUE,au_cop_tem,au_
    rom_tem,c),sim(rh=TRUE,wi_cop_tem,wi_rom_tem,c),sim(rh=TRUE,sp_cop_tem,sp_rom_
    tem,c))
rh_tem_cop_rom[5] <- mean(rh_tem_cop_rom[1:4])
rh_tem_cop_rom[6] <- sim(rh=TRUE,cop_tem,rom_tem,c)
#
rh_tem_mun_rom <- c(sim(rh=TRUE,su_mun_tem,su_rom_tem,c),sim(rh=TRUE,au_mun_tem,au_
    rom_tem,c),sim(rh=TRUE,wi_mun_tem,wi_rom_tem,c),sim(rh=TRUE,sp_mun_tem,sp_rom_
    tem,c))
rh_tem_mun_rom[5] <- mean(rh_tem_mun_rom[1:4])
rh_tem_mun_rom[6] <- sim(rh=TRUE,mun_tem,rom_tem,c)

### precipitation
#### pd
pd_pre_vie_cop <- c(sim(prec=TRUE,su_vie_pre_ma7,su_cop_pre_ma7),sim(prec=TRUE,au_
    vie_pre_ma7,au_cop_pre_ma7),sim(prec=TRUE,wi_vie_pre_ma7,wi_cop_pre_ma7),sim(
    prec=TRUE,sp_vie_pre_ma7,sp_cop_pre_ma7))
pd_pre_vie_cop[5] <- mean(pd_pre_vie_cop[1:4])
pd_pre_vie_cop[6] <- sim(prec=TRUE,vie_pre_ma7,cop_pre_ma7)
#
pd_pre_vie_mun <- c(sim(prec=TRUE,su_vie_pre_ma7,su_mun_pre_ma7),sim(prec=TRUE,au_
    vie_pre_ma7,au_mun_pre_ma7),sim(prec=TRUE,wi_vie_pre_ma7,wi_mun_pre_ma7),sim(
    prec=TRUE,sp_vie_pre_ma7,sp_mun_pre_ma7))
pd_pre_vie_mun[5] <- mean(pd_pre_vie_mun[1:4])
pd_pre_vie_mun[6] <- sim(prec=TRUE,vie_pre_ma7,mun_pre_ma7)
#
pd_pre_vie_rom <- c(sim(prec=TRUE,su_vie_pre_ma7,su_rom_pre_ma7),sim(prec=TRUE,au_
    vie_pre_ma7,au_rom_pre_ma7),sim(prec=TRUE,wi_vie_pre_ma7,wi_rom_pre_ma7),sim(
    prec=TRUE,sp_vie_pre_ma7,sp_rom_pre_ma7))
pd_pre_vie_rom[5] <- mean(pd_pre_vie_rom[1:4])
pd_pre_vie_rom[6] <- sim(prec=TRUE,vie_pre_ma7,rom_pre_ma7)
#
pd_pre_cop_mun <- c(sim(prec=TRUE,su_cop_pre_ma7,su_mun_pre_ma7),sim(prec=TRUE,au_
    cop_pre_ma7,au_mun_pre_ma7),sim(prec=TRUE,wi_cop_pre_ma7,wi_mun_pre_ma7),sim(
```

```
    prec=TRUE,sp_cop_pre_ma7,sp_mun_pre_ma7))
pd_pre_cop_mun[5] <- mean(pd_pre_cop_mun[1:4])
pd_pre_cop_mun[6] <- sim(prec=TRUE,cop_pre_ma7,mun_pre_ma7)
#
pd_pre_cop_rom <- c(sim(prec=TRUE,su_cop_pre_ma7,su_rom_pre_ma7),sim(prec=TRUE,au_
    cop_pre_ma7,au_rom_pre_ma7),sim(prec=TRUE,wi_cop_pre_ma7,wi_rom_pre_ma7),sim(
    prec=TRUE,sp_cop_pre_ma7,sp_rom_pre_ma7))
pd_pre_cop_rom[5] <- mean(pd_pre_cop_rom[1:4])
pd_pre_cop_rom[6] <- sim(prec=TRUE,cop_pre_ma7,rom_pre_ma7)
#
pd_pre_mun_rom <- c(sim(prec=TRUE,su_mun_pre_ma7,su_rom_pre_ma7),sim(prec=TRUE,au_
    mun_pre_ma7,au_rom_pre_ma7),sim(prec=TRUE,wi_mun_pre_ma7,wi_rom_pre_ma7),sim(
    prec=TRUE,sp_mun_pre_ma7,sp_rom_pre_ma7))
pd_pre_mun_rom[5] <- mean(pd_pre_mun_rom[1:4])
pd_pre_mun_rom[6] <- sim(prec=TRUE,mun_pre_ma7,rom_pre_ma7)
### rh
rh_pre_vie_cop <- c(sim(prec=TRUE,rh=TRUE,su_vie_pre_ma7,su_cop_pre_ma7),sim(prec=
    TRUE,rh=TRUE,au_vie_pre_ma7,au_cop_pre_ma7),sim(prec=TRUE,rh=TRUE,wi_vie_pre_
    ma7,wi_cop_pre_ma7),sim(prec=TRUE,rh=TRUE,sp_vie_pre_ma7,sp_cop_pre_ma7))
rh_pre_vie_cop[5] <- mean(rh_pre_vie_cop[1:4])
rh_pre_vie_cop[6] <- sim(prec=TRUE,rh=TRUE,vie_pre_ma7,cop_pre_ma7)
#
rh_pre_vie_mun <- c(sim(prec=TRUE,rh=TRUE,su_vie_pre_ma7,su_mun_pre_ma7),sim(prec=
    TRUE,rh=TRUE,au_vie_pre_ma7,au_mun_pre_ma7),sim(prec=TRUE,rh=TRUE,wi_vie_pre_
    ma7,wi_mun_pre_ma7),sim(prec=TRUE,rh=TRUE,sp_vie_pre_ma7,sp_mun_pre_ma7))
rh_pre_vie_mun[5] <- mean(rh_pre_vie_mun[1:4])
rh_pre_vie_mun[6] <- sim(prec=TRUE,rh=TRUE,vie_pre_ma7,mun_pre_ma7)
#
rh_pre_vie_rom <- c(sim(prec=TRUE,rh=TRUE,su_vie_pre_ma7,su_rom_pre_ma7),sim(prec=
    TRUE,rh=TRUE,au_vie_pre_ma7,au_rom_pre_ma7),sim(prec=TRUE,rh=TRUE,wi_vie_pre_
    ma7,wi_rom_pre_ma7),sim(prec=TRUE,rh=TRUE,sp_vie_pre_ma7,sp_rom_pre_ma7))
rh_pre_vie_rom[5] <- mean(rh_pre_vie_rom[1:4])
rh_pre_vie_rom[6] <- sim(prec=TRUE,rh=TRUE,vie_pre_ma7,rom_pre_ma7)
#
rh_pre_cop_mun <- c(sim(prec=TRUE,rh=TRUE,su_cop_pre_ma7,su_mun_pre_ma7),sim(prec=
    TRUE,rh=TRUE,au_cop_pre_ma7,au_mun_pre_ma7),sim(prec=TRUE,rh=TRUE,wi_cop_pre_
    ma7,wi_mun_pre_ma7),sim(prec=TRUE,rh=TRUE,sp_cop_pre_ma7,sp_mun_pre_ma7))
rh_pre_cop_mun[5] <- mean(rh_pre_cop_mun[1:4])
rh_pre_cop_mun[6] <- sim(prec=TRUE,rh=TRUE,cop_pre_ma7,mun_pre_ma7)
#
rh_pre_cop_rom <- c(sim(prec=TRUE,rh=TRUE,su_cop_pre_ma7,su_rom_pre_ma7),sim(prec=
    TRUE,rh=TRUE,au_cop_pre_ma7,au_rom_pre_ma7),sim(prec=TRUE,rh=TRUE,wi_cop_pre_
    ma7,wi_rom_pre_ma7),sim(prec=TRUE,rh=TRUE,sp_cop_pre_ma7,sp_rom_pre_ma7))
rh_pre_cop_rom[5] <- mean(rh_pre_cop_rom[1:4])
rh_pre_cop_rom[6] <- sim(prec=TRUE,rh=TRUE,cop_pre_ma7,rom_pre_ma7)
#
rh_pre_mun_rom <- c(sim(prec=TRUE,rh=TRUE,su_mun_pre_ma7,su_rom_pre_ma7),sim(prec=
    TRUE,rh=TRUE,au_mun_pre_ma7,au_rom_pre_ma7),sim(prec=TRUE,rh=TRUE,wi_mun_pre_
    ma7,wi_rom_pre_ma7),sim(prec=TRUE,rh=TRUE,sp_mun_pre_ma7,sp_rom_pre_ma7))
rh_pre_mun_rom[5] <- mean(rh_pre_mun_rom[1:4])
rh_pre_mun_rom[6] <- sim(prec=TRUE,rh=TRUE,mun_pre_ma7,rom_pre_ma7)

## monthly precipitation sums
for (i in 1:12) { vie_pre_sum[i] <- sum(subset(climate$vie_pre, climate$mm == i))/
    10 }
for (i in 1:12) { cop_pre_sum[i] <- sum(subset(climate$cop_pre, climate$mm == i))/
    10 }
for (i in 1:12) { mun_pre_sum[i] <- sum(subset(climate$mun_pre, climate$mm == i))/
    10 }
for (i in 1:12) { rom_pre_sum[i] <- sum(subset(climate$rom_pre, climate$mm == i))/
    10 }

### r values comparing normal, ma7 and log applied precipitation data
#### pd
##### summer
r_pd_su_vie_cop <- c(sim(prec=TRUE,su_vie_pre,su_cop_pre),sim(prec=TRUE,ma=7,su_vie
    _pre,su_cop_pre),sim(prec=TRUE,log=TRUE,su_vie_pre,su_cop_pre))
r_pd_su_vie_mun <- c(sim(prec=TRUE,su_vie_pre,su_mun_pre),sim(prec=TRUE,ma=7,su_vie
    _pre,su_mun_pre),sim(prec=TRUE,log=TRUE,su_vie_pre,su_mun_pre))
```

```
227 r_pd_su_vie_rom <- c(sim(prec=TRUE, su_vie_pre, su_rom_pre), sim(prec=TRUE, ma=7, su_vie
        _pre, su_rom_pre), sim(prec=TRUE, log=TRUE, su_vie_pre, su_rom_pre))
    r_pd_su_cop_mun <- c(sim(prec=TRUE, su_cop_pre, su_mun_pre), sim(prec=TRUE, ma=7, su_cop
        _pre, su_mun_pre), sim(prec=TRUE, log=TRUE, su_cop_pre, su_mun_pre))
229 r_pd_su_cop_rom <- c(sim(prec=TRUE, su_cop_pre, su_rom_pre), sim(prec=TRUE, ma=7, su_cop
        _pre, su_rom_pre), sim(prec=TRUE, log=TRUE, su_cop_pre, su_rom_pre))
    r_pd_su_mun_rom <- c(sim(prec=TRUE, su_mun_pre, su_rom_pre), sim(prec=TRUE, ma=7, su_mun
        _pre, su_rom_pre), sim(prec=TRUE, log=TRUE, su_mun_pre, su_rom_pre))
231 ##### winter
    r_pd_wi_vie_cop <- c(sim(prec=TRUE, wi_vie_pre, wi_cop_pre), sim(prec=TRUE, ma=7, wi_vie
        _pre, wi_cop_pre), sim(prec=TRUE, log=TRUE, wi_vie_pre, wi_cop_pre))
233 r_pd_wi_vie_mun <- c(sim(prec=TRUE, wi_vie_pre, wi_mun_pre), sim(prec=TRUE, ma=7, wi_vie
        _pre, wi_mun_pre), sim(prec=TRUE, log=TRUE, wi_vie_pre, wi_mun_pre))
    r_pd_wi_vie_rom <- c(sim(prec=TRUE, wi_vie_pre, wi_rom_pre), sim(prec=TRUE, ma=7, wi_vie
        _pre, wi_rom_pre), sim(prec=TRUE, log=TRUE, wi_vie_pre, wi_rom_pre))
235 r_pd_wi_cop_mun <- c(sim(prec=TRUE, wi_cop_pre, wi_mun_pre), sim(prec=TRUE, ma=7, wi_cop
        _pre, wi_mun_pre), sim(prec=TRUE, log=TRUE, wi_cop_pre, wi_mun_pre))
    r_pd_wi_cop_rom <- c(sim(prec=TRUE, wi_cop_pre, wi_rom_pre), sim(prec=TRUE, ma=7, wi_cop
        _pre, wi_rom_pre), sim(prec=TRUE, log=TRUE, wi_cop_pre, wi_rom_pre))
237 r_pd_wi_mun_rom <- c(sim(prec=TRUE, wi_mun_pre, wi_rom_pre), sim(prec=TRUE, ma=7, wi_mun
        _pre, wi_rom_pre), sim(prec=TRUE, log=TRUE, wi_mun_pre, wi_rom_pre))
    #### rh
239 ##### summer
    r_rh_su_vie_cop <- c(sim(rh=TRUE, prec=TRUE, su_vie_pre, su_cop_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, su_vie_pre, su_cop_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, su_vie_pre, su_
        cop_pre))
241 r_rh_su_vie_mun <- c(sim(rh=TRUE, prec=TRUE, su_vie_pre, su_mun_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, su_vie_pre, su_mun_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, su_vie_pre, su_
        mun_pre))
    r_rh_su_vie_rom <- c(sim(rh=TRUE, prec=TRUE, su_vie_pre, su_rom_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, su_vie_pre, su_rom_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, su_vie_pre, su_
        rom_pre))
243 r_rh_su_cop_mun <- c(sim(rh=TRUE, prec=TRUE, su_cop_pre, su_mun_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, su_cop_pre, su_mun_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, su_cop_pre, su_
        mun_pre))
    r_rh_su_cop_rom <- c(sim(rh=TRUE, prec=TRUE, su_cop_pre, su_rom_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, su_cop_pre, su_rom_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, su_cop_pre, su_
        rom_pre))
245 r_rh_su_mun_rom <- c(sim(rh=TRUE, prec=TRUE, su_mun_pre, su_rom_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, su_mun_pre, su_rom_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, su_mun_pre, su_
        rom_pre))
    ##### winter
247 r_rh_wi_vie_cop <- c(sim(rh=TRUE, prec=TRUE, wi_vie_pre, wi_cop_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, wi_vie_pre, wi_cop_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, wi_vie_pre, wi_
        cop_pre))
    r_rh_wi_vie_mun <- c(sim(rh=TRUE, prec=TRUE, wi_vie_pre, wi_mun_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, wi_vie_pre, wi_mun_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, wi_vie_pre, wi_
        mun_pre))
249 r_rh_wi_vie_rom <- c(sim(rh=TRUE, prec=TRUE, wi_vie_pre, wi_rom_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, wi_vie_pre, wi_rom_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, wi_vie_pre, wi_
        rom_pre))
    r_rh_wi_cop_mun <- c(sim(rh=TRUE, prec=TRUE, wi_cop_pre, wi_mun_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, wi_cop_pre, wi_mun_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, wi_cop_pre, wi_
        mun_pre))
251 r_rh_wi_cop_rom <- c(sim(rh=TRUE, prec=TRUE, wi_cop_pre, wi_rom_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, wi_cop_pre, wi_rom_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, wi_cop_pre, wi_
        rom_pre))
    r_rh_wi_mun_rom <- c(sim(rh=TRUE, prec=TRUE, wi_mun_pre, wi_rom_pre), sim(rh=TRUE, prec=
        TRUE, ma=7, wi_mun_pre, wi_rom_pre), sim(rh=TRUE, prec=TRUE, log=TRUE, wi_mun_pre, wi_
        rom_pre))
253

255 # ============================================================================
    # plot
257 # ============================================================================

259 ## example functions
    ### functions
261 pdf("img/example.pdf", width=10, height=5)
```

```
plot(t1,type="l",xlab="time",ylab="value",main="Example functions",col="indianred",
    xaxt="n",yaxt="n")
lines(t2,col="lightskyblue3")
abline(v=0,lty="solid")
abline(v=900,lty="dashed")
abline(v=1800,lty="dashed")
abline(v=2700,lty="dashed")
abline(v=3600,lty="solid")
axis(1, at=c(0.5:3.5)*900, labels=c(paste("Season 1 (PD=",format(sim(min_tem=0,max_
    tem=100,t1[c(1:900)],t2[c(1:900)],50),digits=2),")"),paste("Season 2 (PD=",
    format(sim(min_tem=0,max_tem=100,t1[c(901:1800)],t2[c(901:1800)],50),digits=2),
    ")"),paste("Season 3 (PD=",format(sim(min_tem=0,max_tem=100,t1[c(1801:2700)],t2
    [c(1801:2700)],50),digits=2),")"),paste("Season 4 (PD=",format(sim(min_tem=0,
    max_tem=100,t1[c(2701:3600)],t2[c(2701:3600)],50),digits=2),")")), las=0)
mtext(paste("PD for whole timespan: ",sim(t1,t2,50,min_tem=0,max_tem=100,)),side=3,
    line=0)

### histograms
pdf("img/ex_hist_t1.pdf",width=5,height=5)
hist(t1,main="Function 1",col="indianred",xlab="value",ylim=c(60,120),breaks=40)
pdf("img/ex_hist_t2.pdf",width=5,height=5)
hist(t2,main="Function 2",col="lightskyblue3",xlab="value",ylim=c(60,120),breaks
    =40)

## histograms
### temperature
pdf("img/hist_vie_tem.pdf",width=5,height=5)
hist(vie_tem,main="Vienna",col="red",xlab="C",xlim=c(-30,40),ylim=c(0,250),breaks
    =50)
mtext(paste(round(mean(vie_tem),1), " C"),side=3,line=0)
pdf("img/hist_cop_tem.pdf",width=5,height=5)
hist(cop_tem,main="Copenhagen",col="red",xlab="C",xlim=c(-30,40),ylim=c(0,250),
    breaks=50)
mtext(paste(round(mean(cop_tem),1), " C"),side=3,line=0)
pdf("img/hist_mun_tem.pdf",width=5,height=5)
hist(mun_tem,main="Munich",col="red",xlab="C",xlim=c(-30,40),ylim=c(0,250),breaks
    =50)
mtext(paste(round(mean(mun_tem),1), " C"),side=3,line=0)
pdf("img/hist_rom_tem.pdf",width=5,height=5)
hist(rom_tem,main="Rome",col="red",xlab="C",xlim=c(-30,40),ylim=c(0,250),breaks=50)
mtext(paste(round(mean(rom_tem),1), " C"),side=3,line=0)

### precipitation
#pdf("img/hist_wi_vie_pre.pdf",width=5,height=5)
#hist(wi_vie_pre,main="Vienna",col="lightblue",xlab="mm",xlim=c(0,20),ylim=c(0,600)
    ,breaks=60)
#pdf("img/hist_wi_cop_pre.pdf",width=5,height=5)
#hist(wi_cop_pre,main="Copenhagen",col="lightblue",xlab="mm",xlim=c(0,20),ylim=c
    (0,600),breaks=60)
pdf("img/hist_mun_pre.pdf",width=5,height=5)
hist(mun_pre,main="Munich",col="lightblue",xlab="mm",xlim=c(0,40),ylim=c(0,600),
    breaks=60)
pdf("img/hist_rom_pre.pdf",width=5,height=5)
hist(rom_pre,main="Rome",col="lightblue",xlab="mm",xlim=c(0,40),ylim=c(0,600),
    breaks=80)
#### moving average
#pdf("img/hist_wi_vie_pre_ma7.pdf",width=5,height=5)
#hist(wi_vie_pre_ma7,main="Vienna, moving average over 7 days",col="lightblue",xlab
    ="mm",xlim=c(0,20),ylim=c(0,600),breaks=10)
#pdf("img/hist_wi_cop_pre_ma7.pdf",width=5,height=5)
#hist(wi_cop_pre_ma7,main="Copenhagen, moving average over 7 days",col="lightblue",
    xlab="mm",xlim=c(0,20),ylim=c(0,600),breaks=10)
pdf("img/hist_mun_pre_ma7.pdf",width=5,height=5)
hist(mun_pre_ma7,main="Munich, moving average over 7 days",col="lightblue",xlab="mm
    ",xlim=c(0,40),ylim=c(0,600),breaks=20)
pdf("img/hist_rom_pre_ma7.pdf",width=5,height=5)
hist(rom_pre_ma7,main="Rome, moving average over 7 days",col="lightblue",xlab="mm",
    xlim=c(0,40),ylim=c(0,600),breaks=40)

### precipitation monthly sum
```

```
313 pdf("img/vie_pre_sum.pdf",width=5,height=5)
    barplot(vie_pre_sum,main="Vienna",col="lightblue",xlab="",ylab="mm",ylim=c(0,200),
        names.arg=c("J","F","M","A","M","J","J","A","S","O","N","D"))
315 mtext(paste(round(sum(vie_pre_sum)), " mm"),side=3,line=0)
    #
317 pdf("img/cop_pre_sum.pdf",width=5,height=5)
    barplot(cop_pre_sum,main="Copenhagen",col="lightblue",xlab="",ylab="mm",ylim=c
        (0,200),names.arg=c("J","F","M","A","M","J","J","A","S","O","N","D"))
319 mtext(paste(round(sum(cop_pre_sum)), " mm"),side=3,line=0)
    #
321 pdf("img/mun_pre_sum.pdf",width=5,height=5)
    barplot(mun_pre_sum,main="Munich",col="lightblue",xlab="",ylab="mm",ylim=c(0,200),
        names.arg=c("J","F","M","A","M","J","J","A","S","O","N","D"))
323 mtext(paste(round(sum(mun_pre_sum)), " mm"),side=3,line=0)
    #
325 pdf("img/rom_pre_sum.pdf",width=5,height=5)
    barplot(rom_pre_sum,main="Rome",col="lightblue",xlab="",ylab="mm",ylim=c(0,200),
        names.arg=c("J","F","M","A","M","J","J","A","S","O","N","D"))
327 mtext(paste(round(sum(rom_pre_sum)), " mm"),side=3,line=0)

329 ## r values vs categories
    ### pd 100
331 pdf("img/pd100.pdf",width=10,height=7)
    plot(r_cat$x,r_cat$pd_vie_cop,type="l",xlab="number of categories",ylab="
        Proportional Similarity",main="Proportional Similarity vs. number of categories
        ",ylim=c(0.5,1),xlim=c(1,100),col="blue")
333 lines(r_cat$x,r_cat$pd_vie_mun,col="red")
    lines(r_cat$x,r_cat$pd_vie_rom,col="green")
335 lines(r_cat$x,r_cat$pd_cop_mun,col="black")
    lines(r_cat$x,r_cat$pd_cop_rom,col="orangered1")
337 lines(r_cat$x,r_cat$pd_mun_rom,col="chocolate4")
    abline(v=32,lty="dashed")
339 legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
        Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
        blue","red","green","black","orangered1","chocolate4"),ncol=2)
    #### spring
341 pdf("img/sp_pd100.pdf",width=10,height=7)
    plot(r_cat$x,r_cat$sp_pd_vie_cop,type="l",xlab="number of categories",ylab="
        Proportional Similarity",main="Spring: Proportional Similarity vs. number of
        categories",ylim=c(0,1),xlim=c(1,100),col="blue")
343 lines(r_cat$x,r_cat$sp_pd_vie_mun,col="red")
    lines(r_cat$x,r_cat$sp_pd_vie_rom,col="green")
345 lines(r_cat$x,r_cat$sp_pd_cop_mun,col="black")
    lines(r_cat$x,r_cat$sp_pd_cop_rom,col="orangered1")
347 lines(r_cat$x,r_cat$sp_pd_mun_rom,col="chocolate4")
    abline(v=32,lty="dashed")
349 legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
        Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
        blue","red","green","black","orangered1","chocolate4"),ncol=2)
    #### summer
351 pdf("img/su_pd100.pdf",width=10,height=7)
    plot(r_cat$x,r_cat$su_pd_vie_cop,type="l",xlab="number of categories",ylab="
        Proportional Similarity",main="Summer: Proportional Similarity vs. number of
        categories",ylim=c(0,1),xlim=c(1,100),col="blue")
353 lines(r_cat$x,r_cat$su_pd_vie_mun,col="red")
    lines(r_cat$x,r_cat$su_pd_vie_rom,col="green")
355 lines(r_cat$x,r_cat$su_pd_cop_mun,col="black")
    lines(r_cat$x,r_cat$su_pd_cop_rom,col="orangered1")
357 lines(r_cat$x,r_cat$su_pd_mun_rom,col="chocolate4")
    abline(v=32,lty="dashed")
359 legend("topright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
        Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
        blue","red","green","black","orangered1","chocolate4"),ncol=2)
    #### autumn
361 pdf("img/au_pd100.pdf",width=10,height=7)
    plot(r_cat$x,r_cat$au_pd_vie_cop,type="l",xlab="number of categories",ylab="
        Proportional Similarity",main="Autumn: Proportional Similarity vs. number of
        categories",ylim=c(0,1),xlim=c(1,100),col="blue")
363 lines(r_cat$x,r_cat$au_pd_vie_mun,col="red")
    lines(r_cat$x,r_cat$au_pd_vie_rom,col="green")
```

```
lines(r_cat$x,r_cat$au_pd_cop_mun,col="black")
lines(r_cat$x,r_cat$au_pd_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$au_pd_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)
#### winter
pdf("img/wi_pd100.pdf",width=10,height=7)
plot(r_cat$x,r_cat$wi_pd_vie_cop,type="l",xlab="number of categories",ylab="
    Proportional Similarity",main="Winter: Proportional Similarity vs. number of
    categories",ylim=c(0,1),xlim=c(1,100),col="blue")
lines(r_cat$x,r_cat$wi_pd_vie_mun,col="red")
lines(r_cat$x,r_cat$wi_pd_vie_rom,col="green")
lines(r_cat$x,r_cat$wi_pd_cop_mun,col="black")
lines(r_cat$x,r_cat$wi_pd_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$wi_pd_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)

### pd 1000
pdf("img/pd1000.pdf",width=10,height=7)
plot(r_cat$x,r_cat$pd_vie_cop,type="l",xlab="number of categories",ylab="
    Proportional Similarity",main="Proportional Similarity vs. number of categories
    ",ylim=c(0.5,1),xlim=c(1,1000),col="blue")
lines(r_cat$x,r_cat$pd_vie_mun,col="red")
lines(r_cat$x,r_cat$pd_vie_rom,col="green")
lines(r_cat$x,r_cat$pd_cop_mun,col="black")
lines(r_cat$x,r_cat$pd_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$pd_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)

### rh 100
pdf("img/rh100.pdf",width=10,height=7)
plot(r_cat$x,r_cat$rh_vie_cop,type="l",xlab="number of categories",ylab="Hellinger
    Coefficient",main="Hellinger Coefficient vs. number of categories",ylim=c
    (0.5,1),xlim=c(1,100),col="blue")
lines(r_cat$x,r_cat$rh_vie_mun,col="red")
lines(r_cat$x,r_cat$rh_vie_rom,col="green")
lines(r_cat$x,r_cat$rh_cop_mun,col="black")
lines(r_cat$x,r_cat$rh_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$rh_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)
#### spring
pdf("img/sp_rh100.pdf",width=10,height=7)
plot(r_cat$x,r_cat$sp_rh_vie_cop,type="l",xlab="number of categories",ylab="
    Hellinger Coefficient",main="Spring: Hellinger Coefficient vs. number of
    categories",ylim=c(0,1),xlim=c(1,100),col="blue")
lines(r_cat$x,r_cat$sp_rh_vie_mun,col="red")
lines(r_cat$x,r_cat$sp_rh_vie_rom,col="green")
lines(r_cat$x,r_cat$sp_rh_cop_mun,col="black")
lines(r_cat$x,r_cat$sp_rh_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$sp_rh_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)
#### summer
pdf("img/su_rh100.pdf",width=10,height=7)
plot(r_cat$x,r_cat$su_rh_vie_cop,type="l",xlab="number of categories",ylab="
    Hellinger Coefficient",main="Summer: Hellinger Coefficient vs. number of
    categories",ylim=c(0,1),xlim=c(1,100),col="blue")
```

```
lines(r_cat$x,r_cat$su_rh_vie_mun,col="red")
lines(r_cat$x,r_cat$su_rh_vie_rom,col="green")
lines(r_cat$x,r_cat$su_rh_cop_mun,col="black")
lines(r_cat$x,r_cat$su_rh_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$su_rh_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)
#### autumn
pdf("img/au_rh100.pdf",width=10,height=7)
plot(r_cat$x,r_cat$au_rh_vie_cop,type="l",xlab="number of categories",ylab="
    Hellinger Coefficient",main="Autumn: Hellinger Coefficient vs. number of
    categories",ylim=c(0,1),xlim=c(1,100),col="blue")
lines(r_cat$x,r_cat$au_rh_vie_mun,col="red")
lines(r_cat$x,r_cat$au_rh_vie_rom,col="green")
lines(r_cat$x,r_cat$au_rh_cop_mun,col="black")
lines(r_cat$x,r_cat$au_rh_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$au_rh_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)
#### winter
pdf("img/wi_rh100.pdf",width=10,height=7)
plot(r_cat$x,r_cat$wi_rh_vie_cop,type="l",xlab="number of categories",ylab="
    Hellinger Coefficient",main="Winter: Hellinger Coefficient vs. number of
    categories",ylim=c(0,1),xlim=c(1,100),col="blue")
lines(r_cat$x,r_cat$wi_rh_vie_mun,col="red")
lines(r_cat$x,r_cat$wi_rh_vie_rom,col="green")
lines(r_cat$x,r_cat$wi_rh_cop_mun,col="black")
lines(r_cat$x,r_cat$wi_rh_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$wi_rh_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)

### rh 1000
pdf("img/rh1000.pdf",width=10,height=7)
plot(r_cat$x,r_cat$rh_vie_cop,type="l",xlab="number of categories",ylab="Hellinger
    Coefficient",main="Hellinger Coefficient vs. number of categories",ylim=c
    (0.5,1),xlim=c(1,1000),col="blue")
lines(r_cat$x,r_cat$rh_vie_mun,col="red")
lines(r_cat$x,r_cat$rh_vie_rom,col="green")
lines(r_cat$x,r_cat$rh_cop_mun,col="black")
lines(r_cat$x,r_cat$rh_cop_rom,col="orangered1")
lines(r_cat$x,r_cat$rh_mun_rom,col="chocolate4")
abline(v=32,lty="dashed")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),lty="solid",col=c("
    blue","red","green","black","orangered1","chocolate4"),ncol=2)

## combine r values
### temperature
#### pd
pdf("img/pd_temp.pdf",width=8,height=5)
plot(pd_tem_vie_cop,ylim=c(0,1),col="blue",xaxt="n",ylab="Proportional Similarity",
    xlab="",pch=19,main="Temperature similarity - Proportional Similarity")
points(pd_tem_vie_mun,col="red",pch=19)
points(pd_tem_vie_rom,col="green",pch=19)
points(pd_tem_cop_mun,col="black",pch=19)
points(pd_tem_cop_rom,col="orangered1",pch=19)
points(pd_tem_mun_rom,col="chocolate4",pch=19)
abline(v=4.5,lty="solid")
#abline(v=5.75,lty="solid")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),col=c("blue","red","
    green","black","orangered1","chocolate4"),ncol=2,pch=19,bg="white")
```

```
axis(1, at=c(1:6), labels=c("Summer","Autumn","Winter","Spring","Season average","
    Whole year"), las=0)
#### rh
pdf("img/rh_temp.pdf",width=8,height=5)
plot(rh_tem_vie_cop,ylim=c(0,1),col="blue",xaxt="n",ylab="Hellinger Coefficient",
    xlab="",pch=19,main="Temperature similarity - Hellinger Coefficient")
points(rh_tem_vie_mun,col="red",pch=19)
points(rh_tem_vie_rom,col="green",pch=19)
points(rh_tem_cop_mun,col="black",pch=19)
points(rh_tem_cop_rom,col="orangered1",pch=19)
points(rh_tem_mun_rom,col="chocolate4",pch=19)
abline(v=4.5,lty="solid")
#abline(v=5.75,lty="solid")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),col=c("blue","red","
    green","black","orangered1","chocolate4"),ncol=2,pch=19,bg="white")
axis(1, at=c(1:6), labels=c("Summer","Autumn","Winter","Spring","Season average","
    Whole year"), las=0)

## precipitation
### pd
pdf("img/pd_prec.pdf",width=8,height=5)
plot(pd_pre_vie_cop,ylim=c(0.3,1),col="blue",xaxt="n",ylab="Proportional Similarity
    ",xlab="",pch=19,main="Precipitation similarity - Proportional Similarity")
points(pd_pre_vie_mun,col="red",pch=19)
points(pd_pre_vie_rom,col="green",pch=19)
points(pd_pre_cop_mun,col="black",pch=19)
points(pd_pre_cop_rom,col="orangered1",pch=19)
points(pd_pre_mun_rom,col="chocolate4",pch=19)
abline(v=4.5,lty="solid")
#abline(v=5.75,lty="solid")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),col=c("blue","red","
    green","black","orangered1","chocolate4"),ncol=2,pch=19,bg="white")
axis(1, at=c(1:6), labels=c("Summer","Autumn","Winter","Spring","Season average","
    Whole year"), las=0)
### rh
pdf("img/rh_prec.pdf",width=8,height=5)
plot(rh_pre_vie_cop,ylim=c(0.3,1),col="blue",xaxt="n",ylab="Hellinger Coefficient",
    xlab="",pch=19,main="Precipitation similarity - Hellinger Coefficient")
points(rh_pre_vie_mun,col="red",pch=19)
points(rh_pre_vie_rom,col="green",pch=19)
points(rh_pre_cop_mun,col="black",pch=19)
points(rh_pre_cop_rom,col="orangered1",pch=19)
points(rh_pre_mun_rom,col="chocolate4",pch=19)
abline(v=4.5,lty="solid")
#abline(v=5.75,lty="solid")
legend("bottomright",c("Vienna - Copenhagen","Vienna - Munich","Vienna - Rome","
    Copenhagen - Munich","Copenhagen - Rome","Munich - Rome"),col=c("blue","red","
    green","black","orangered1","chocolate4"),ncol=2,pch=19,bg="white")
axis(1, at=c(1:6), labels=c("Summer","Autumn","Winter","Spring","Season average","
    Whole year"), las=0)

# precipitation filters
## pd
### summer
pdf("img/pd_pre_su_filters.pdf",width=5,height=5)
plot(r_pd_su_vie_cop,ylim=c(0,1),col="blue",xaxt="n",ylab="Proportional Similarity"
    ,xlab="",pch=19,main="Proportional Similarity, summer")
points(r_pd_su_vie_mun,col="red",pch=19)
points(r_pd_su_vie_rom,col="green",pch=19)
points(r_pd_su_cop_mun,col="black",pch=19)
points(r_pd_su_cop_rom,col="orangered1",pch=19)
points(r_pd_su_mun_rom,col="chocolate4",pch=19)
legend("bottomleft",c("Vie. - Cop.","Vie. - Mun.","Vie. - Rome","Cop. - Mun.","Cop.
     - Rome","Mun. - Rome"),col=c("blue","red","green","black","orangered1","
    chocolate4"),ncol=2,pch=19,bg="white")
axis(1, at=c(1:3), labels=c("Original","7 days moving average","log"), las=0)
### winter
pdf("img/pd_pre_wi_filters.pdf",width=5,height=5)
```

```
521 plot(r_pd_wi_vie_cop,ylim=c(0,1),col="blue",xaxt="n",ylab="Proportional Similarity"
    ,xlab="",pch=19,main="Proportional Similarity, winter")
    points(r_pd_wi_vie_mun,col="red",pch=19)
523 points(r_pd_wi_vie_rom,col="green",pch=19)
    points(r_pd_wi_cop_mun,col="black",pch=19)
525 points(r_pd_wi_cop_rom,col="orangered1",pch=19)
    points(r_pd_wi_mun_rom,col="chocolate4",pch=19)
527 legend("bottomleft",c("Vie. - Cop.","Vie. - Mun.","Vie. - Rome","Cop. - Mun.","Cop.
     - Rome","Mun. - Rome"),col=c("blue","red","green","black","orangered1","
    chocolate4"),ncol=2,pch=19,bg="white")
    axis(1, at=c(1:3), labels=c("Original","7 days moving average","log"), las=0)
529 ## rh
    ### summer
531 pdf("img/rh_pre_su_filters.pdf",width=5,height=5)
    plot(r_rh_su_vie_cop,ylim=c(0,1),col="blue",xaxt="n",ylab="Hellinger Coefficient",
    xlab="",pch=19,main="Hellinger Coefficient, summer")
533 points(r_rh_su_vie_mun,col="red",pch=19)
    points(r_rh_su_vie_rom,col="green",pch=19)
535 points(r_rh_su_cop_mun,col="black",pch=19)
    points(r_rh_su_cop_rom,col="orangered1",pch=19)
537 points(r_rh_su_mun_rom,col="chocolate4",pch=19)
    legend("bottomleft",c("Vie. - Cop.","Vie. - Mun.","Vie. - Rome","Cop. - Mun.","Cop.
     - Rome","Mun. - Rome"),col=c("blue","red","green","black","orangered1","
    chocolate4"),ncol=2,pch=19,bg="white")
539 axis(1, at=c(1:3), labels=c("Original","7 days moving average","log"), las=0)
    ### winter
541 pdf("img/rh_pre_wi_filters.pdf",width=5,height=5)
    plot(r_rh_wi_vie_cop,ylim=c(0,1),col="blue",xaxt="n",ylab="Hellinger Coefficient",
    xlab="",pch=19,main="Hellinger Coefficient, winter")
543 points(r_rh_wi_vie_mun,col="red",pch=19)
    points(r_rh_wi_vie_rom,col="green",pch=19)
545 points(r_rh_wi_cop_mun,col="black",pch=19)
    points(r_rh_wi_cop_rom,col="orangered1",pch=19)
547 points(r_rh_wi_mun_rom,col="chocolate4",pch=19)
    legend("bottomleft",c("Vie. - Cop.","Vie. - Mun.","Vie. - Rome","Cop. - Mun.","Cop.
     - Rome","Mun. - Rome"),col=c("blue","red","green","black","orangered1","
    chocolate4"),ncol=2,pch=19,bg="white")
549 axis(1, at=c(1:3), labels=c("Original","7 days moving average","log"), las=0)

551
    # ============================================================================
553 # export
    # ============================================================================
555
    dev.off()
```

A.2 Java

A.2.1 Climate Twin Connector

This code was originally written by Jan Peters-Anders (AIT, jan.peters-anders@ait.ac.at) and just modified by the author, who implemented the similarity measures worked out in this thesis.

```
    package test;
2
    import java.sql.Connection;
4   import java.io.*;
    import java.sql.Array;
6   import java.sql.DriverManager;
    import java.sql.ResultSet;
8   import java.sql.SQLException;
    import java.sql.Statement;
10  import java.util.ArrayList;
```

```
import java.util.Arrays;
import java.util.List;
import java.util.Vector;
import java.util.Random;
import java.lang.String;
import java.math.*;
import java.lang.Number;

public class ClimateConnector {
  static int errorLevel = 1;

  final int ENTIRE_CLIMATE = 0;

  final int TEMP = 1;

  final int PREC = 2;

  final int ENTIRE_YEAR = 0;

  final int WINTER = 1;

  final int SPRING = 2;

  final int SUMMER = 3;

  final int AUTUMN = 4;

  final int f1961t1970 = 0;

  final int f1971t1980 = 1;

  final int f1981t1990 = 2;

  final int f1991t2000 = 3;

  final int f2001t2010 = 4;

  final int f2011t2020 = 5;

  final int f2021t2030 = 6;

  final int f2031t2040 = 7;

  final int f2041t2050 = 8;

  final int f2051t2060 = 9;

  final int f2061t2070 = 10;

  final int f2071t2080 = 11;

  final int f2081t2090 = 12;

  final int f2091t2100 = 13;

  final int PD = 0;

  final int RH = 1;

  public String executeDQuery(int id, int thtemp, int thprec, int indicator,
      double indicatorWeight, int sourcePeriod, int targetPeriod,
      int season, int simMeasure) {


    Connection theConnection = null;
    Connection theConnectionPostGIS = null;
    ResultSet theResult;
    String returnMessage = "EMPTY";
    boolean localhost = true;

```

```
    try {
82    Class.forName("org.postgresql.Driver");

84  } catch (ClassNotFoundException e) {
      // TODO Auto-generated catch block
86    e.printStackTrace();
    }
88  new ArrayList<int[][]>();
    try {
90  //    if (localhost == false) {
      //    // neu (ab Maerz 2010):
92        theConnection = DriverManager.getConnection(
              "jdbc:postgresql://localhost:5432/ct_dev", "jan",
94            "***");
          theConnectionPostGIS = DriverManager.getConnection(
96            "jdbc:postgresql://localhost:5432/ct_postgis_dev",
              "jan", "***");
98  //    }
    //    // neu (ab Maerz 2010):
100 //    if (localhost == true) {
    //      theConnection = DriverManager.getConnection(
102 //          "jdbc:postgresql://localhost:5452/ct_dev", "jan",
    //          "***");
104 //      theConnectionPostGIS = DriverManager.getConnection(
    //          "jdbc:postgresql://localhost:5452/ct_postgis_dev",
106 //          "jan", "***");
    //    }
108
      Statement theStatement = theConnection
110       .createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
112   theConnection.setAutoCommit(false);

114   Statement theStatementPostGIS = theConnectionPostGIS
          .createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
116           ResultSet.CONCUR_UPDATABLE);
      theConnectionPostGIS.setAutoCommit(false);
118
      // ================================================================
120   // start similarity function

122   // float[] sourceArray = new float[28];
      // float[][] scenarioArrays = new float[28][61456];
124
      // indicator selector
126   String StrIndicator = "herbert";
      int ArrayLength = 0;
128   String[] indicatorArray = { "climate", "tm", "rr" };
      switch (indicator) {
130   case ENTIRE_CLIMATE:
        StrIndicator = indicatorArray[ENTIRE_CLIMATE];
132     break;
      case TEMP:
134     StrIndicator = indicatorArray[TEMP];

136     break;
      case PREC:
138     StrIndicator = indicatorArray[PREC];

140     break;
      }
142
      // season selector
144   String StrSeason = "herbert";

146   String[] seasonArray = { "all", "wi", "sp", "su", "au" };

148   switch (season) {
      case ENTIRE_YEAR:
150     StrSeason = seasonArray[ENTIRE_YEAR];
```

```
        break;
152     case WINTER:
          StrSeason = seasonArray[WINTER];
154       break;
        case SPRING:
156       StrSeason = seasonArray[SPRING];
          break;
158     case SUMMER:
          StrSeason = seasonArray[SUMMER];
160       break;
        case AUTUMN:
162       StrSeason = seasonArray[AUTUMN];
          break;
164     }

166     // period selector
        String StrSourcePeriod = "herbert";
168
        String[] periodArray = { "1961_1970", "1971_1980", "1981_1990",
170         "1991_2000", "2001_2010", "2011_2020", "2021_2030",
            "2031_2040", "2041_2050", "2051_2060", "2061_2070",
172         "2071_2080", "2081_2090", "2091_2100" };

174     switch (sourcePeriod) {
        case f1961t1970:
176       StrSourcePeriod = periodArray[f1961t1970];
          break;
178     case f1971t1980:
          StrSourcePeriod = periodArray[f1971t1980];
180       break;
        case f1981t1990:
182       StrSourcePeriod = periodArray[f1981t1990];
          break;
184     case f1991t2000:
          StrSourcePeriod = periodArray[f1991t2000];
186       break;
        case f2001t2010:
188       StrSourcePeriod = periodArray[f2001t2010];
          break;
190     case f2011t2020:
          StrSourcePeriod = periodArray[f2011t2020];
192       break;
        case f2021t2030:
194       StrSourcePeriod = periodArray[f2021t2030];
          break;
196     case f2031t2040:
          StrSourcePeriod = periodArray[f2031t2040];
198       break;
        case f2041t2050:
200       StrSourcePeriod = periodArray[f2041t2050];
          break;
202     case f2051t2060:
          StrSourcePeriod = periodArray[f2051t2060];
204       break;
        case f2061t2070:
206       StrSourcePeriod = periodArray[f2061t2070];
          break;
208     case f2071t2080:
          StrSourcePeriod = periodArray[f2071t2080];
210       break;
        case f2081t2090:
212       StrSourcePeriod = periodArray[f2081t2090];
          break;
214     case f2091t2100:
          StrSourcePeriod = periodArray[f2091t2100];
216       break;
        }
218
        String StrTargetPeriod = "herbert";
220
```

```
        switch (targetPeriod) {
222     case f1961t1970:
          StrTargetPeriod = periodArray[f1961t1970];
224       break;
        case f1971t1980:
226       StrTargetPeriod = periodArray[f1971t1980];
          break;
228     case f1981t1990:
          StrTargetPeriod = periodArray[f1981t1990];
230       break;
        case f1991t2000:
232       StrTargetPeriod = periodArray[f1991t2000];
          break;
234     case f2001t2010:
          StrTargetPeriod = periodArray[f2001t2010];
236       break;
        case f2011t2020:
238       StrTargetPeriod = periodArray[f2011t2020];
          break;
240     case f2021t2030:
          StrTargetPeriod = periodArray[f2021t2030];
242       break;
        case f2031t2040:
244       StrTargetPeriod = periodArray[f2031t2040];
          break;
246     case f2041t2050:
          StrTargetPeriod = periodArray[f2041t2050];
248       break;
        case f2051t2060:
250       StrTargetPeriod = periodArray[f2051t2060];
          break;
252     case f2061t2070:
          StrTargetPeriod = periodArray[f2061t2070];
254       break;
        case f2071t2080:
256       StrTargetPeriod = periodArray[f2071t2080];
          break;
258     case f2081t2090:
          StrTargetPeriod = periodArray[f2081t2090];
260       break;
        case f2091t2100:
262       StrTargetPeriod = periodArray[f2091t2100];
          break;
264     }

266     // Variables for temporary and result values.
        // ========================================
268
        // int[][][] seasonResult
270     /**
         * is used to store temporary r-values. Values below threshold get
272      * "-999". columns: [0] FID [1] r-value
         */
274     float[][][] seasonResult = new float[2][33080];

276     // ArrayList<int[][][]> seasonResultList
        /**
278      * temporary store of saisonal r-values. contains seasonResult
         * arrays per column. columns: [0] winter [1] spring [2] summer [3]
280      * autumn
         */
282     ArrayList<float[][][]> seasonResultList = new ArrayList<float[][][]>();
        for (int i = 0; i < 4; i++) {
284       seasonResultList.add(i, seasonResult);
        }
286
        // int[][][] indicatorResult
288     /**
         * temporary store of whole year r-values. columns: [0] FID [1]
290      * r-value
```

```
         */
292      float[][][] indicatorResult = new float[3][seasonResult[0].length + 1];

294      // ArrayList<int[][]> indicatorResultList
         /**
296       * permanent store of whole year r-values per indicator. contains
          * indicatorResult arrays per column. columns: [0] temperature [1]
298       * precipitation
          */
300      ArrayList<float[][]> indicatorResultList = new ArrayList<float[][]>();
         for (int i = 0; i < 3; i++) {
302        indicatorResultList.add(i, seasonResult);
         }
304
         // int[][] climateResult
306      /**
          * array to store result data late to be written to database.
308       * columns: [0] FID -> cell ID [1] sim -> overall similarity [2]
          * tm_total -> whole year temperature similarity [3] tm_wi -> winter
310       * temperature similarity [4] tm_sp -> spring temperature similarity
          * [5] tm_su -> summer temperature similarity [6] tm_au -> autumn
312       * temperature similarity [7] rr_total -> whole year precipitation
          * similarity [8] rr_wi -> winter precipitation similarity [9] rr_sp ->
314       * spring precipitation similarity [10] rr_su -> summer
          * precipitation similarity [11] rr_au -> autumn precipitation
316       * similarity
          */
318      float[][] climateResult = new float[12][seasonResult[0].length + 1];

320      // int[][] climateTempList
         /**
322       * temporary array for calculations. columns: [0] FID [1] r-value
          */
324      int[][] climateTempList = new int[4][seasonResult[0].length + 1];

326      int indicatorCountFrom = 0;
         int indicatorCountTo = 0;
328      if (StrIndicator == "climate") {
           sOP("ENTIRE CLIMATE");
330        indicatorCountFrom = 0;
           indicatorCountTo = 2;
332      } else if (StrIndicator == "tm") {
           sOP("TEMP");
334        indicatorCountFrom = 0;
           indicatorCountTo = 1;
336      } else if (StrIndicator == "rr") {
           sOP("PREC");
338        indicatorCountFrom = 1;
           indicatorCountTo = 2;
340      } else {
           sOP("indicator select error!");
342      }

344      int seasonCountFrom = 0;
         int seasonCountTo = 0;
346      if (StrSeason == "all") {
           sOP("ENTIRE YEAR");
348        seasonCountFrom = 0;
           seasonCountTo = 4;
350      } else if (StrSeason == "wi") {
           sOP("WINTER");
352        seasonCountFrom = 0;
           seasonCountTo = 1;
354      } else if (StrSeason == "sp") {
           sOP("SPRING");
356        seasonCountFrom = 1;
           seasonCountTo = 2;
358      } else if (StrSeason == "su") {
           sOP("SUMMER");
360        seasonCountFrom = 2;
```

```
            seasonCountTo = 3;
362         } else if (StrSeason == "au") {
              sOP("AUTUMN");
364           seasonCountFrom = 3;
              seasonCountTo = 4;
366         } else {
              sOP("season select error!");
368         }

370         sOP("temperature threshold: " + thtemp);
            sOP("precipitation threshold: " + thprec);
372         sOP("source region: " + id);
            sOP("source period: " + StrSourcePeriod);
374         sOP("target period: " + StrTargetPeriod);

376         // BEGIN INDICATOR LOOP
            for (int indicatorCount = indicatorCountFrom; indicatorCount <
                indicatorCountTo; indicatorCount++) {
378           StrIndicator = indicatorArray[indicatorCount + 1];
              int tolerance = 0;
380           if (indicatorCount == 0) {
                tolerance = thtemp;
382           } else if (indicatorCount == 1) {
                tolerance = thprec;
384           } else {
                sOP("indicatorCount error!");
386           }
              // BEGIN SEASON LOOP
388           for (int seasonCount = seasonCountFrom; seasonCount < seasonCountTo;
                  seasonCount++) {
                sOP("seasonCount: " + seasonCount);
390             StrSeason = seasonArray[seasonCount + 1];
                sOP("get data " + StrIndicator + " for " + StrSeason);
392             // Build the SQL query code for the source region:
                String query = "SELECT id,dist_" + StrIndicator + "_"
394                 + StrSourcePeriod + "_"
                    + seasonArray[seasonCount + 1]
396                 + " FROM dist_arrays_europe WHERE  id = " + id
                    + ";";
398             sOP(query);
                if (indicatorCount == 0) {
400               ArrayLength = 38;
                } else if (indicatorCount == 1) {
402               ArrayLength = 28;
                }
404             float[] sourceArray = new float[ArrayLength];
                float[][] scenarioArrays = new float[ArrayLength][33080];
406
                for (theResult = theStatement.executeQuery(query); theResult
408                 .next();) {
                  float idControl = (Float.valueOf(theResult.getFloat(1)))
410                     .floatValue();
                  Array array = theResult.getArray(2);
412               sOP(array.getArray().getClass().toString());
                  sOP("idControl: " + idControl);
414               sourceArray[0] = idControl;
                  java.lang.Float[] tempArray = (Float[]) array
416                     .getArray(); // fuer Servlet
                  sOP("tempArray.length: " + tempArray.length);
418               for (int g = 0; g < tempArray.length; g++) {
                    // sOP("tempArray:" + tempArray[g]);
420                 sourceArray[g + 1] = tempArray[g];
                  }
422             }
                // Build the SQL query code for the target regions:
424             query = "SELECT id,dist_" + StrIndicator + "_"
                    + StrTargetPeriod + "_"
426                 + seasonArray[seasonCount + 1]
                    + " FROM dist_arrays_europe;";
428             sOP(query);
```

```
            int rowCount = 0;
430         for (theResult = theStatement.executeQuery(query); theResult
                .next();) {
432           float idScenario = (Float.valueOf(theResult.getInt(1)))
                  .floatValue();
434           Array array = theResult.getArray(2);
              scenarioArrays[0][rowCount] = idScenario;
436           java.lang.Float[] dArray2 = (Float[]) array.getArray(); // fuer
              // Servlet
438           for (int g = 0; g < dArray2.length; g++) {
                // sOP("dArray2("+g+"):" + dArray2[g]+",");
440             scenarioArrays[g + 1][rowCount] = dArray2[g];
              }
442           rowCount++;
            }
444         sOP("begin similarity measurement ...");
            sOP("seasonCount: " + seasonCount);
446         // BEGIN similarity measurement:
            // Computes the r-value between the source region and the
448         // current
            // target region.
450         seasonResult = sim(sourceArray, scenarioArrays, simMeasure,
                ArrayLength, tolerance);
452         sOP("sim() done: " + seasonResult[1][12000]);
            // write the results in the ArrayList seasonResultList.
454         seasonResultList.add(seasonCount, seasonResult);
            sOP("writing done.");
456         theResult.close();
            sOP("end similarity measurement ...");
458
            // BEGIN DEBUGGING INFORMATION
460         // sOP("seasonResultList.get(seasonCount)[1].length: " +
            // seasonResultList.get(seasonCount)[1].length);
462         if (indicatorCount == 0) {
              int temp = seasonCount + 3;
464           climateResult[temp] = seasonResultList.get(seasonCount)[1];
              sOP("climateResult slot " + temp);
466         } else if (indicatorCount == 1) {
              int temp = seasonCount + 8;
468           climateResult[temp] = seasonResultList.get(seasonCount)[1];
              sOP("climateResult slot " + temp);
470         } else {
              sOP("indicatorCount error!");
472         }
            // END DEBUGGING INFORMATION
474       }
          // END SEASON LOOP
476       // BEGIN merge seasons to year
          for (int rowCount = 0; rowCount < seasonResultList
478           .get(seasonCountFrom)[0].length; rowCount++) {
            // sOP("seasonCountFrom: " + seasonCountFrom);
480         indicatorResult[0][rowCount] = seasonResultList
                .get(seasonCountFrom)[0][rowCount]; // get FID
482         indicatorResult[1][rowCount] = -999; // reset variable
            float seasonSimilarity = 0;
484         for (int seasonCount = seasonCountFrom; seasonCount < seasonCountTo;
                seasonCount++) {
              // sOP("checking similarity value for seasonCount: " +
486           // seasonCount + " @ rowCount " + rowCount);
              if (indicatorResult[1][rowCount] != 0) {
488             if (seasonResultList.get(seasonCount)[1][rowCount] < tolerance) {
                  indicatorResult[1][rowCount] = 0;
490             } else {
                  seasonSimilarity = seasonSimilarity
492                   + seasonResultList.get(seasonCount)[1][rowCount];
                  indicatorResult[1][rowCount] = seasonSimilarity
494                   / (seasonCountTo - seasonCountFrom);
                }
496           }
            }
```

```
498             // sOP("similarity value = " +
                // indicatorResult[1][rowCount]);
500             // sOP("indicatorCount: " + indicatorCount);
                // BEGIN DEBUGGING INFORMATION
502             if (indicatorCount == 0) {
                  // sOP("writing similarity value for indicator " +
504               // indicatorCount);
                  climateResult[2][rowCount] = indicatorResult[1][rowCount];
506               indicatorResultList
                      .add(indicatorCount, indicatorResult);
508               // sOP("OK");
                } else if (indicatorCount == 1) {
510               // sOP("writing similarity value for indicator " +
                  // indicatorCount);
512               climateResult[7][rowCount] = indicatorResult[1][rowCount];
                  indicatorResultList
514                   .add(indicatorCount, indicatorResult);
                  // sOP("OK");
516             } else {
                  sOP("indicatorCount error!");
518             }
                // sOP("climateResult[2][rowCount]: " +
520             // climateResult[2][rowCount]);
                // END DEBUGGING INFORMATION
522             // indicatorResultList.add(indicatorCount, indicatorResult);
              }
524           // END merge seasons to year.

526       }

528       // END INDICATOR LOOP

530       // BEGIN combine indicator similarities to climate similarity
          sOP("END indicator loop");
532       sOP("seasonResultList.get(seasonCountFrom)[0].length: "
              + seasonResultList.get(seasonCountFrom)[0].length);
534       for (int rowCount = 0; rowCount < seasonResultList
              .get(seasonCountFrom)[0].length; rowCount++) {
536         climateResult[0][rowCount] = indicatorResultList
                .get(indicatorCountFrom)[0][rowCount]; // get FID
538 //        sOP("StrIndicator: " + StrIndicator);
    //        sOP("indicator: " + indicator);
540           switch (indicator) {
              case ENTIRE_CLIMATE:
542             if (climateResult[2][rowCount] < thtemp
                    || climateResult[7][rowCount] < thprec) {
544               climateResult[1][rowCount] = 0;
                } else {
546               if (climateResult[2][rowCount] < climateResult[7][rowCount]) {
                    climateResult[1][rowCount] = climateResult[2][rowCount]
548                     + (climateResult[7][rowCount] - climateResult[2][rowCount])
                        * (float) indicatorWeight;
550               } else if (climateResult[2][rowCount] > climateResult[7][rowCount]) {
                    climateResult[1][rowCount] = climateResult[7][rowCount]
552                     + (climateResult[2][rowCount] - climateResult[7][rowCount])
                        * (float) (1 - indicatorWeight);
554               } else if (climateResult[2][rowCount] == climateResult[7][rowCount]) {
                    climateResult[1][rowCount] = climateResult[2][rowCount];
556               } else {
                    climateResult[1][rowCount] = -999;
558               }
                }
560             break;
              case TEMP:
562             if (climateResult[2][rowCount] < thtemp) {
                  climateResult[1][rowCount] = 0;
564             } else {
                  climateResult[1][rowCount] = climateResult[2][rowCount];
566             }
                break;
```

```
568           case PREC:
                if (climateResult[7][rowCount] < thprec) {
570               climateResult[1][rowCount] = 0;
                } else {
572               climateResult[1][rowCount] = climateResult[7][rowCount];
                }
574             break;
              }
576         }
            // END combine indicator similarities to climate similarity
578
            sOP("END construction site");
580
            ArrayList selectedItems = new ArrayList();
582         int regionCount = 0;
            for (int i = 0; i < climateResult[0].length; i++) {
584           if (climateResult[1][i] > 0) {

586             selectedItems.add(String.valueOf(climateResult[0][i]));
                regionCount++;
588             DriverManager.println(climateResult[0][i] + "("
                    + climateResult[1][i] + ")");
590           }
            }
592
            // ================================================================
594         // end similarity function

596         // sOP(String.valueOf(regionCount));

598         StringBuffer queryString = new StringBuffer("");

600         theStatementPostGIS
                .addBatch("DROP TABLE result_map_europe_yogi CASCADE;");
602
            Random rand = new Random();
604
            int rand_int = rand.nextInt();
606
            if (rand_int < 0) {
608           rand_int = rand_int * -1;
            }
610
            sOP("rand_int: " + rand_int);
612
            String createTempTable = ("CREATE TABLE test_" + rand_int
614             + " (FID int4," + " sim float(24),"
                + " tm_total float(24)," + " tm_wi float(24),"
616             + " tm_sp float(24)," + " tm_su float(24),"
                + " tm_au float(24)," + " rr_total float(24),"
618             + " rr_wi float(24)," + " rr_sp float(24),"
                + " rr_su float(24)," + " rr_au float(24));");
620
            sOP(createTempTable);
622
            theStatementPostGIS.addBatch(createTempTable);
624
            sOP("climateResult[0].length: " + climateResult[0].length);
626
            for (int t = 0; t < climateResult[0].length - 1; t++) {
628
              theStatementPostGIS
630               .addBatch("INSERT INTO public.test_"
                      + rand_int
632                   + " (FID, sim, tm_total, tm_wi, tm_sp, tm_su, tm_au, rr_total, rr_
                          wi, rr_sp, rr_su, rr_au) VALUES ("
                      + climateResult[0][t] + ", "
634                   + climateResult[1][t] + ", "
                      + climateResult[2][t] + ", "
636                   + climateResult[3][t] + ", "
```

```
                + climateResult[4][t] + ", "
                + climateResult[5][t] + ", "
                + climateResult[6][t] + ", "
                + climateResult[7][t] + ", "
                + climateResult[8][t] + ", "
                + climateResult[9][t] + ", "
                + climateResult[10][t] + ", "
                + climateResult[11][t] + ");");
        // sOP("Batch No. "+t+" added!");
        // sOP("similarity = " + climateResult[1][t]);
        // sOP("temperature similarity = " + climateResult[2][t]);
        // sOP("precipitation similarity = " + climateResult[7][t]);
      }
      sOP("insert table done!");

      if (selectedItems.size() == 0) {
        sOP("No Match! --> No Result Selection.");
        queryString.append(" \"input_fid\"< 0'");
      }

      if (selectedItems.size() == 1) {
        sOP("selectedItems(0): " + (String) selectedItems.get(0));
        queryString.append(" \"input_fid\"='"
            + (String) selectedItems.get(0) + "'");
      } else {
        for (int i = 0; i < selectedItems.size(); i++) {
          // sOP("selectedItems("+i+"): " +
          // (String)selectedItems.get(i));
          queryString.append(" \"input_fid\"='"
              + (String) selectedItems.get(i) + "'");

          if (i < selectedItems.size() - 1) {
            queryString.append(" OR ");
          }

        }

        ArrayList record = new ArrayList();

        record = executeQuery(theStatementPostGIS, theConnectionPostGIS);

        theStatementPostGIS.clearBatch();

        String createTable = ("CREATE TABLE result_map_europe_yogi (FID)  AS SELECT
              a.FID,"
            + " a.sim, "
            + " a.tm_total,"
            + " a.tm_wi,"
            + " a.tm_sp,"
            + " a.tm_su,"
            + " a.tm_au,"
            + " a.rr_total,"
            + " a.rr_wi,"
            + " a.rr_sp,"
            + " a.rr_su,"
            + " a.rr_au,"
            + " b.the_geom FROM test_"
            + rand_int + " a INNER JOIN g_rot b ON a.FID = b.g_rot_id;");//
        String addPrimaryKey = "ALTER TABLE result_map_europe_yogi ADD PRIMARY KEY
            (FID);";

        sOP(createTable);
        theStatementPostGIS.addBatch(createTable);

        sOP(addPrimaryKey);
        theStatementPostGIS.addBatch(addPrimaryKey);

                theStatementPostGIS.addBatch("DROP TABLE point_of_interest_yogi;")
                    ;
                String poiCreateString = "CREATE TABLE point_of_interest_yogi (id)
```

```
                    AS SELECT  g_rot.g_rot_id,   g_rot.the_geom FROM g_rot WHERE g
                    _rot.g_rot_id = "+id+"; ";
704                 sOP("poi: " + poiCreateString);
                    theStatementPostGIS.addBatch(poiCreateString);
706

708         record = executeQuery(theStatementPostGIS, theConnectionPostGIS);

710         // theResult.close();
            theStatement.close();
712         theStatementPostGIS.close();
            theConnection.close();
714         sOP("DATABASE UPDATE DONE (PREC)!");
            returnMessage = ("#DATABASE_UPDATE_DONE_PREC#");
716
            XML_Writer xw = new XML_Writer();
718           xw.writeXMLValues(String.valueOf(id),String.valueOf(thtemp),String.
                  valueOf(thprec),String.valueOf(indicator),
                  String.valueOf(indicatorWeight),String.valueOf(sourcePeriod),String.
                      valueOf(targetPeriod),String.valueOf(season),String.valueOf(
                      simMeasure));
720       }
        } catch (Exception e) {
722       sOP(e.toString());
          returnMessage = e.getLocalizedMessage();
724     }

726     return returnMessage;

728   }

730   // begin sim()

732   /**
       * calculates r-value. returns int[][][] columns: [0] FID [1] r-value
734    */
      public float[][][] sim(float[] xArray, float[][][] yArray, int simMeasure,
736       int ArrayLength, int threshold) { // output: array including id
        // and r-value; input: x and y
738     // array (frequencies),
        // similarity measure (pd/rh))
740     float regionID = 0.0f;
        float[] sourceArray = new float[ArrayLength];
742     float[][] scenarioArrays = new float[ArrayLength][33080];
        float[][][] climateHitList = new float[2][scenarioArrays[0].length];
744     sourceArray = xArray;
        scenarioArrays = yArray;
746     for (int rows = 0; rows < scenarioArrays[0].length; rows++) {
          double sim = 0;
748       regionID = scenarioArrays[0][rows];
          for (int i = 1; i < ArrayLength; i++) {
750
            float val1 = scenarioArrays[i][rows];
752         float val2 = sourceArray[i];

754         // set variables
            float targetArraysSum = 0;
756         float sourceArraySum = 0;
            float[] scenarioArrayRel = new float[ArrayLength];
758         float[] sourceArrayRel = new float[ArrayLength];
            sim = 0;
760         // calculate sum
            for (int k = 1; k < ArrayLength; k++) {
762           targetArraysSum = targetArraysSum + scenarioArrays[k][rows];
              sourceArraySum = sourceArraySum + sourceArray[k];
764         }
            // sOP("control sum: " + controlArraySum + " scenario sum: "
766         // + scenarioArraysSum);
            // calculate relative frequencies
768         for (int r = 1; r < ArrayLength; r++) {
```

```
            scenarioArrayRel[r] = 100 * scenarioArrays[r][rows]
770             / targetArraysSum;
            sourceArrayRel[r] = 100 * sourceArray[r] / sourceArraySum;
772         // sOP(scenarioArrayRel[r] + "," + controlArrayRel[r]);

774         switch (simMeasure) {
            case PD:
776           // Proportional Similarity
              if (scenarioArrayRel[r] < sourceArrayRel[r]) {
778             sim = sim + scenarioArrayRel[r];
              } else {
780             sim = sim + sourceArrayRel[r];
              }
782           break;
            case RH: {
784           // Hellinger Coefficient
              sim = Math
786               .sqrt(scenarioArrayRel[r] * sourceArrayRel[r])
                  + sim;
788         }
            }
790
          }
792       // sOP("PD: " + sim);
          // climateHitList[1][rows] = sim;
794       // sOP("relative frequency: " + scenarioArrayRel);
        }
796     climateHitList[0][rows] = (Integer.valueOf((int) regionID))
            .intValue();
798     if (sim >= threshold) {
          climateHitList[1][rows] = (float) sim;
800     } else {
          climateHitList[1][rows] = 0;
802     }

804   }

806   float result[][][] = climateHitList;
      return result;
808 }

810 // end sim()

812 public static ArrayList executeQuery(Statement stmt, Connection conn) {
      // Connection theConnection;
814   ArrayList result = new ArrayList();
      try {
816
        stmt.executeBatch();
818     int[] updateCounts = stmt.executeBatch();
        sOP("updateCounts: " + updateCounts.length);
820
        // sOP(stmt.executeBatch());
822     conn.commit();

824     result.add(0, "OK!");

826   } catch (SQLException e) {
        // procees to the next exception
828     e = e.getNextException();
        e.printStackTrace();
830     result.add(0, "Failed!");
        return result;
832   }
      return result;
834 }

836 private static void sOP(String text) {
      if (errorLevel == 1) {
838     System.out.println(text);
```

```
    }
840
  }
842


844
  /**
846  * @param args
   */
848 public static void main(String[] args) {

850   // TODO Auto-generated method stub

852   final int ENTIRE_CLIMATE = 0;
      final int TEMP = 1;
854   final int PREC = 2;

856   final int f1961t1970 = 0;
      final int f1971t1980 = 1;
858   final int f1981t1990 = 2;
      final int f1991t2000 = 3;
860   final int f2001t2010 = 4;
      final int f2011t2020 = 5;
862   final int f2021t2030 = 6;
      final int f2031t2040 = 7;
864   final int f2041t2050 = 8;
      final int f2051t2060 = 9;
866   final int f2061t2070 = 10;
      final int f2071t2080 = 11;
868   final int f2081t2090 = 12;
      final int f2091t2100 = 13;
870
      final int ENTIRE_YEAR = 0;
872   final int WINTER = 1;
      final int SPRING = 2;
874   final int SUMMER = 3;
      final int AUTUMN = 4;
876
      final int PD = 0;
878
      final int RH = 1;
880
      ClimateConnector cp = new ClimateConnector();
882   cp.executeDQuery(181119, 80, 90, TEMP, 0.5, f2001t2010, f2001t2010,
          SUMMER, PD);
884 }

886 }
```

Bibliography

Andronova, N. G. and Schlesinger, M. E. (2001). Objective estimation of the probability density function for climate sensitivity. *Journal of Geophysical Research*, 106:22605–22612.

Barry, R. G. and Chorley, R. J. (1992). *Atmosphere, weather, and climate*. Routledge.

Bauer, A. and Günzel, H. (2004). *Data-Warehouse-Systeme. Architektur, Entwicklung, Anwendung*. Dpunkt.Verlag GmbH, 2., überarb. und aktualis. a. edition.

Beven, K. and Binley, A. (1992). The future of distributed models: Model calibration and uncertainty prediction. *Hydrological Processes*, 6(3):279–298.

Biehl, M., Hammer, B., Verleysen, M., and Villmann, T. (2009). *Similarity-Based Clustering*.

Bock, H. H. and Diday, E. (2000). *Analysis of symbolic data*. Springer.

Bowman, A. W. and Azzalini, A. (1997). *Applied smoothing techniques for data analysis*. Oxford University Press.

Cartwright, W. (1997). New media and their application to the production of map products. *Computers & Geosciences*, 23(4):447–456.

Cartwright, W. (2009). Moving from map and geospatial information provision with the web to collaborative publishing using web 2.0. In *Geokommunikation im Umfeld der Geographie. Tagungsband zum Deutschen Geographentag 2009 in Wien*, volume 19 of *Wiener Schriften zur Geographie und Kartographie*, pages 9 – 22. Karel Kriz, Wolfgang Kainz und Andreas Riedl, Wien.

Cartwright, W., Peterson, M. P., and Gartner, G. F. (2007). *Multimedia cartography*. Springer.

Conrad, V. (2007). *Methods in Climatology*. Read Books.

Dettinger, M. D., Cayan, D. R., Diaz, H. F., and Meko, D. M. (1998). North–South precipitation patterns in western north america on Interannual-to-Decadal timescales. *Journal of Climate*, 11(12):3095–3111.

Drosg, M. (2009). *Dealing with Uncertainties*. Springer.

Déqué, M., Rowell, D., Lüthi, D., Giorgi, F., Christensen, J., Rockel, B., Jacob, D., Kjellström, E., de Castro, M., and van den Hurk, B. (2007). An intercomparison of regional climate simulations for europe: assessing uncertainties in model projections. *Climatic Change*, 81(0):53–70.

Eckey, H., Kosfeld, R., and Rengers, M. (2002). *Multivariate Statistik*. Gabler Verlag.

Ferber, R. (2003). *Information Retrieval. Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. Dpunkt Verlag, 1 edition.

Feynman, R. P., Leighton, R., and Hutchings, E. (1997). *"Surely you're joking, Mr. Feynman!"*. W.W. Norton.

Forest, C. E., Stone, P. H., Sokolov, A. P., Allen, M. R., and Webster, M. D. (2002). Quantifying uncertainties in climate system properties with the use of recent climate observations. *Science*, 295(5552):113–117.

Gaile, G. L. and Willmott, C. J. (1984). *Spatial statistics and models*. Springer.

Gerstengarbe, F., Werner, P. C., and Fraedrich, K. (1999). Applying Non-Hierarchical cluster analysis algorithms to climate classification: Some problems and their solution. *Theoretical and Applied Climatology*, 64(3):143–150.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. (1995). Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia*, pages 231–236, San Francisco, California, United States. ACM.

Gower, J. (1978). Some remarks on proportional similarity. *The Journal of general microbiology*, 107:387–389.

Grabisch, M. and Nguyen, H. T. (1994). *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*. Kluwer Academic Publishers.

Güßefeldt, J. (1999). *Regionalanalyse, m. CD-ROM*. Oldenbourg, 2 edition.

Heisenberg, W. (1969). *Der Teil und das Ganze*. R. Piper.

Henderson-Sellers, A. and McGuffie, K. (1988). *A Climate Modelling Primer*. John Wiley & Sons Ltd.

Higgins, A., Bahler, L., and Porter, J. (1993). Voice identification using nearest-neighbor distance measure. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 375–378 vol.2.

Houghton, J. T., Yihui, D., and Griggs, D. J. (2001). *Climate Change 2001: The Scientific Basis: Contribution of Working Group I to the Third Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press.

Hubbard, D. W. (2007). *How to measure anything*. John Wiley and Sons.

Hutcheson, G. and Sofroniou, N. (1999). *The multivariate social scientist: introductory statistics using generalized linear models*. SAGE.

India, M. B. and Bonillo, D. L. (2001). *Detecting and modelling regional climate change*. Springer.

Kainz, W. and Mayer, F. (1993). *GIS Und Kartographie: theoretische Grundlagen und Zukunftsaspekte*, volume 6 of *Wiener Schriften zur Geographie und Kartographie*. Institute for Geography, University of Vienna, Wien.

Kline, P. (2000). *The handbook of psychological testing*. Routledge.

Klir, G. J. and Folger, T. A. (1988). *Fuzzy Sets, Uncertainty and Information*. Prentice Hall.

Kottek, M., Grieser, J., Beck, C., Rudolf, B., and Rubel, F. (2006). World map of the Köppen-Geiger climate classification updated. *Meteorologische Zeitschrift*, 15(3):259–263.

Lautenschlager, M., Keuler, K., Wunram, C., Keup-Thiel, E., Schubert, M., Will, A., Rockel, B., and Boehm, U. (2009). Climate simulation with CLM, scenario A1B run no.1, data stream 3: European region MPI-M/MaD. doi:10.1594/WDCC/CLM_A1B_1_D3.

Lin, D. (1998). An Information-Theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann Publishers Inc.

Loibl, W., Beck, A., Dorninger, M., Formayer, H., Gobiet, A., and Schöner, W. (2007). reclip:more - research for climate protection: model run, evaluation, executive summary. Technical report, ARC-sys., Wien.

Loibl, W., Peters-Anders, J., and Züger, J. (2010). Climate twins - a tool to explore future climate impacts by assessing real world conditions: Exploration principles, underlying data, similarity conditions and uncertainty ranges. *Geophysical Research Abstracts*, Vol. 12(EGU2010-12149).

Loibl, W., Züger, J., and Köstl, M. (2009). Reclip:more. *Standort - Zeitschrift für angewandte Geographie*, 33(3):94–100.

MacEachren, A. M. (2004). *How maps work*. Guilford Press.

Malberg, H. (2002). *Meteorologie und Klimatologie. Eine Einführung*. Springer-Verlag GmbH, 3., aktualis. u. erw. a. edition.

Malerba, D., Esposito, F., and Monopoli, M. (2002). Comparing dissimilarity measures for probabilistic symbolic objects. *Data mining III*, page 31.

Minasny, B., McBratney, A. B., and Bristow, K. L. (1999). Comparison of different approaches to the development of pedotransfer functions for water-retention curves. *Geoderma*, 93(3-4):225–253.

Nielsen, J. (1993). *Usability engineering*. Morgan Kaufmann.

Oliver, J. E. (2005). *Encyclopedia of world climatology*. Springer.

Peterson, M. P. (2003). *Maps and the internet*. Elsevier.

Plaut, G. and Simonnet, E. (2001). Large-scale circulation classification, weather regimes, and local climate over france, the alps and western europe. *Clim Res*, 17(3):303–324.

Power, C., Simms, A., and White, R. (2001). Hierarchical fuzzy pattern matching for the regional comparison of land use maps. *International Journal of Geographical Information Science*, 15:77–100.

Riedl, A. (2000). *Virtuelle Globen in der Geovisualisierung*. Dissertation, University of Vienna.

Roeckner, E., Bäuml, G., Bonaventura, L., Brokopf, R., Esch, M., Giorgetta, M., Hagemann, S., Kirchner, I., Kornblueh, L., Manzini, E., Rhodin, A., Schlese, U., Schulzweida, U., and Tompkins, A. (2003). The atmospheric general circulation model ECHAM 5. PART i: Model description. Technical report, Max-Planck-Institute for Meteorology.

Rohli, R. V. and Vega, A. J. (2007). *Climatology*. Jones & Bartlett Publishers.

Rubel, F. and Kottek, M. (2010). Observed and projected climate shifts 1901–2100 depicted by world maps of the Köppen-Geiger climate classification. *Meteorologische Zeitschrift*, 19(2):135–141.

Santini, S. and Jain, R. (1999). Similarity measures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):871–883.

Schobesberger, D. and Nausner, B. (2009). User interface design und usability von kartographischen Online-Informationssystemen. In *Geokommunikation im Umfeld der Geographie. Tagungsband zum Deutschen Geographentag 2009 in Wien*, volume 19 of *Wiener Schriften zur Geographie und Kartographie*, pages 76 – 81. Karel Kriz, Wolfgang Kainz und Andreas Riedl, Wien.

Schönwiese, C. (2008). *Klimatologie*. UTB, Stuttgart, 3., verbesserte und aktualisierte aufl. edition.

Semenov, M. A., Brooks, R. J., Barrow, E. M., and Richardson, C. W. (1998). Comparison of the WGEN and LARS-WG stochastic weather generators for diverse climates. http://eprints.lancs.ac.uk/6088/.

Shneiderman, B. and Plaisant, C. (2009). *Designing the User Interface*. Addison-Wesley.

Sint, P. P. (1975). *Ähnlichkeitsstrukturen und Ähnlichkeitsmaße*. Number 1/1975 in Schriftenreihe des Instituts für Sozioökonomische Entwicklungsforschung.

Solomon, S., Qin, D., Manning, M., Chen, Z., Marquis, M., Averyt, K., Tignor, M., and Miller, H. (2007). *Climate Change 2007 - The Physical Science Basis: I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Spertus, E., Sahami, M., and Buyukkokten, O. (2005). Evaluating similarity measures: a large-scale study in the orkut social network. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 678–684, Chicago, Illinois, USA. ACM.

Steadman, R. G. (1979). The assessment of sultriness. part II: effects of wind, extra radiation and barometric pressure on apparent temperature. *Journal of Applied Meteorology*, 18(7):874–885.

Strahler, A. (1951). *Physical Geography*. J. Wiley and Sons, New York.

Stricker, M. A., Orengo, M., Niblack, W., and Jain, R. C. (1995). Similarity of color images. In *Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 381–392, San Jose, CA, USA. SPIE.

Thornthwaite, C. W. (1948). An approach toward a rational classification of climate. *Geographical Review*, 38(1):55–94.

Vegelius, J., Janson, S., and Johansson, F. (1986). Measures of similarity between distributions. *Quality and Quantity*, 20(4):437–441.

von der Lippe, P. M. (1993). *Deskriptive Statistik*. Utb.

Vosniadou, S. and Ortony, A. (1989). *Similarity and analogical reasoning*. Cambridge University Press.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3):338–353.

Curriculum Vitae

Joachim Ungar
curriculum vitae

General Information
born 16.04.1984 in Innsbruck
Austrian

Contact
Graf Starhemberggasse 6/18
1040 Vienna, Austria
Tel.: +43 650 49 79 691
joachim.ungar@gmail.com

Education

- 1994-2002: secondary education: Akademisches Gymnasium Salzburg (AHS)
- 21.6.2002: Final examination
- 2004-2011: Study of Cartography and Geoinformation, Department of Geography and Regional Research, University of Vienna

Professional Experience

- 10/2002 – 09/2003: community service at Lebenshilfe Salzburg, carer for the handicapped
- 02/2008: Web Developer, HTML implementation of an Annual Report for Ge-Ber
- 05/2008 – 02/2009 (minimally employed): Web Developer at Ge-Ber Geschäftsberichte (now Nexxar – www.nexxar.com)
- 2008/2009: Tutor of „Multimedia and Geocommunication“ Part I and II, University of Vienna
- 04/2010 – 05/2010: Voluntary GIS work for the Instituto Geografico Militar (www.igm.cl) after the Chilean earthquake in February 2010
- 11/2009 – 10/2010: Diploma thesis within the Climate Twins project at the Austrian Institute of Technology (AIT)
- since 10/2010: AIT independent contractor, Department of Foresight and Policy Development Department

Abilities

- Foreign languages: English, basics in Spanish and French
- Computing:
  - various operating systems (Ubuntu Linux, Windows, Mac OS)
  - GIS (ArcGIS, ERDAS, Quantum GIS, GDAL/OGR, ...), WebGIS (MapServer, Mapnik, OpenGeo Suite, Gaia, ...)

- basic programming skills (R, Python, PHP, Java)
- image processing and publishing (GIMP, InkSkape, ImageMagick, Illustrator, Photoshop)
- word processing / office (OpenOffice, MS Office, LaTeX)
- web developing (HTML, CSS, JavaScript)