# From method development to software integration: A comprehensive approach to geostatistical variogram uncertainty

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN (Dr. rer. nat.)

von der KIT-Fakultät für
Bauingenieur-, Geo- und Umweltwissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. **Mirko Mälicke**

aus Karlsruhe

**Tag der mündlichen Prüfung:** 29.09.2023

**Referenten:**
Prof. Dr.-Ing. Erwin Zehe
Prof. Dr.-Ing. Uwe Haberlandt
Prof. Dr. Achim Streit

Karlsruhe (2024)

# Abstract

This dissertation focuses on the analysis of spatial variability and uncertainty in geostatistics. The importance of well-estimated variograms for geostatistical analyses in highlighted and the need for tools to handle uncertainty and variability in geostatistics effectively is emphasized. This work combines method development in geostatistics with the development of new research software and dedicates a full chapter to the synthesis of both. It highlights how research depends on suitable tools on the one hand, but also how applicable research can frame software development, to be more effective and pointed on the other hand.

We present a new measure for capturing spatial dissimilarity and analyze the temporal evolution and emergence of soil moisture patterns using the Mean shift clustering algorithm. An included case-study uses a novel measure to capture the spatial dissimilarity of soil moisture and its change over time, showing that even a few soil moisture time series contain a considerable amount of information about dynamic changes in soil moisture. Spatial information contained in soil moisture observations is highly redundant, which suggests that soil moisture observations can be compressed without significant information loss.

A geostatistical software library was developed simultaneously with the preceding case study and over course of the following couple of years. Chapter A presents this open-source Python package for variogram estimation, SciKit-GStat. The package provides a flexible and interactive approach to variogram estimation that fits well into established frameworks for scientific computing. SciKit-GStat ships with a large number of predefined procedures, algorithms, and models, such as variogram estimators, theoretical spatial models, and binning algorithms.

With SciKit-GStat available, building on findings described in chapter 2, sophisticated, interactive tools for building an extension to the library were developed and are introduced in chapter 3. The approach replaces the empirical variogram with its uncertainty bound to acknowledge uncertainties characterizing the underlying geostatistical datasets and typical methodological approaches. This allows for a probabilistic description of the variogram and its parameters, enabling multiple interpretations of a sample and a multi-model context for geostatistical applications. The study shows how insights on uncertainty can be used to reject variogram mod-

els, thus constraining the space of formally equally probable models and addressing the issue of parameter equifinality.

Finally, B combines all three preceding chapters into a unified scientific workflow definition. A framework for combining reusable research software into replicable scientific analysis workflows, with a focus on implementing input and output interfaces for research software to increase transparency is discussed. The framework aims to make a clear distinction between a reusable workflow to reproduce existing results and a replicable analysis to apply previously obtained research findings to new data in a different context. A robust analysis using force-directed graphs to examine and visualize the covariance structure of representative variograms from the Attert catchment is presented. Here, the findings from 2 are replicated in a more systematic way using the software from chapter A and extend the analysis of the underlying covariance described by the associated empirical variograms, using force-directed for visualization in an original way. The chapter concludes with a discussion of promising approaches to interpreting the force-directed graph and its relationship to the covariance of the related variogram.

# Zusammenfassung

Diese Dissertation befasst sich mit der Analyse von räumlicher Variabilität und Unsicherheiten in der Geostatistik. Gut geschätzte Variogramme sind Voraussetzung für geostatistische Analysen und mit dieser Arbeit werden weitere Werkzeuge zum effektiven Umgang mit Unsicherheiten und Variabilität in der Geostatistik vorgestellt. Diese Arbeit verbindet die Methodenentwicklung in der Geostatistik mit der Entwicklung neuer Forschungssoftware und widmet ein ganzes Kapitel der Synthese beider Bereiche. Sie zeigt auf, wie die Forschung einerseits von geeigneten Werkzeugen abhängt, andererseits aber auch, wie angewandte Forschung einen kontextualisierten Rahmen für die Softwareentwicklung bietet. Nur so ist ein sinnvoller und effektiver Einsatz von Forschungssoftware möglich.

Durch die räumliche Beschreibung statistischer Unähnlichkeit können wir mithilfe des Mean-Shift-Clustering Algorithmus die zeitliche Entwicklung von Bodenfeuchtigkeitsmustern analysieren. In einer Fallstudie im Attert Einzugsgebiet in Luxembourg konnten wir mithilfe dieser Methodik zeigen, dass Zeitreihen der Bodenfeuchte zeitlich redundant sind und bereits kurze Abschnitte einen beträchtlichen Informationsgehalt über dynamische Veränderungen der Bodenfeuchte enthalten. Dies deutet darauf hin, dass Bodenfeutebeobachtungen ohne erheblichen Informationsverlust komrimiert werden können

Mit der oben genannten Studie starteten die ersten Arbeiten für eine geostatistische Softwarebibliothek, welche sich über viele Jahre erstrecken sollten. Im Kapitel A wird dieses vorgestellt: SciKit-GStat, ein open-source Paket für die Variogramschätzung, geschrieben in Python. Das Paket bietet einen flexiblen und interaktiven Ansatz zur Variogrammschätzung, der sich gut in etablierte Workflows für wissenschaftliches Rechnen einfügt. SciKit-GStat wird mit einer großen Anzahl von vordefinierten Funktionen, Algorithmen und Modellen ausgeliefert, wie z.B. Variogrammschätzer, theoretische räumliche Modelle und Binning-Algorithmen.

Mit der Veröffentlichung von SciKit-GStat wurden weitere interaktive Werkzeuge zur Erweiterung der Bibliothek entwickelt, die direkt auf Erkenntnisse aus Kapitel 2 aufbauen. Dies wird im Detail in Kapitel 3 beschrieben. Der Ansatz ersetzt das empirische Variogramm mit seinem Unsicherheitsband, um die Unsicherheiten zu berücksichtigen, die aus den zugrunde liegenden geostatistischen Datensätze methodischen Ansätzen stammen. Dies ermöglicht eine probabilistische

Beschreibung des Variogramms und seiner Parameter, die mehrere Interpretationen einer Stichprobe in einen Multimodell-Kontext ermöglicht. Die Studie zeigt, wie Erkenntnisse über die so beschriebene Unsicherheit genutzt werden können, um Variogramm-Modelle abzulehnen und so den Raum formal gleich wahrscheinlicher Modelle einzuschränken und Erkenntnisse zur Parameteräquifinalität der Modelle zu gewinnen.

Schließlich schlägt Kapitel B als Synthese der vorangegangenen Kapitel eine exemplarische Definition wissenschaftlicher Reproduzierbarkeit vor. Ziel ist eine verbesserte Transparenz in der Anwendung wissenschafter Analyse-Workflows, die durch die Implementierung einheitlicher Eingabe- und Ausgabeschnittstellen Forschungssoftware kontextualisiert, reproduzierbar und kombinierbar macht. Das vorgeschlagene Framework trifft eine klare Unterscheidung zwischen einer reproduzierbaren Prozedur, und replizierbaren wissenschaftlichen Methoden, deren Ziel die Anwendung bereits gewonnener Erkenntnisse in einen neuen Kontext hat. Zur Veranschaulichung wird eine robuste Analyse vorgestellt, bei der die Kovarianzstruktur von wiederkehrenden Bodenfeuchtemustern aus dem Einzugsgebiet von Attert mit Hilfe von kräftegeleiteten Graphen untersucht und visualisiert wird. Hier werden die Ergebnisse aus dem Kapitel 2 mit Hilfe von SciKit-GStat systematisch repliziert. Die Analyse der Kovarinaz durch repräsentative Variogramme, wird durch die Visualisierung mittels kräftegeleiteten Graphen erweitert. Das Kapitel schließt mit einer Diskussion über vielversprechende Ansätze zur Interpretation des kraftgerichteten Graphen und seiner Beziehung zur Kovarianz des zugehörigen Variogramms.

# Acknowledgments

I want to thank my supervisor Erwin Zehe, who poured countless hours of sweat and toil into guiding me through this scientific jungle. He kept my spirits up with enlightening conversations and broke up the monotony with welcome distractions.
I would also like to express my heartfelt thanks to Alberto Guadagnini for sharing his insights on geostatistical uncertainty with me.
My colleagues at the Hydrology working group at KIT deserve a shout-out, too. You guys made the journey more enjoyable with your camaraderie and support. I couldn't have made it this far without you.
I'd like to give a special shout-out to the co-authors of my publications - Erwin Zehe, Alberto Guadagnini, Sibylle Hassler, Theresa Blume, and Markus Weiler. Your insights and feedback were invaluable, and I'm proud to have worked alongside such talented individuals.
Last but not least, I'm indebted to my family for their unwavering support. My wife Sofie, in particular, deserves a medal for putting up with my endless blathering about research topics that probably put her to sleep. And my two boys, Jakob and Linus, have been the ultimate cheerleaders, even if they occasionally had to sacrifice their daddy time to science. Thank you all from the bottom of my heart.

# Contents

# 1 Introduction

"Good, reproducible science needs good, generalizable research software." This is one of the prime conclusions of this dissertation. Moreover, it has become the main credo of my work underpinning all following chapters. Generalizable software means software that can suit use cases beyond a specific implementation or configuration as used within the scope of a single research study. At first sight, it might seem to be hampering research, as time and/or resources are assigned to software development that has no immediate impact on the results of the respective research. During the course of this work, I will argue how this is not only short-minded, but actually, the opposite is true. A thorough description of research software and the context of its application lays the foundation to present an example of research software from the field of geostatistics: The variogram estimation toolbox SciKit-GStat (see chapter A).

## 1.1 Introduction to geostatistics

At its core, geostatistics is based on the basic assumption that for a given random field, proximity in space and proximity in quantity are correlated. More precisely, the closer two observations are in space, the more similar they are expected to be. The objective of geostatistics is to detect and mathematically formalize this correlation as a model function and either analyze its parameters or estimate (or simulate) the quantity at unobserved locations.

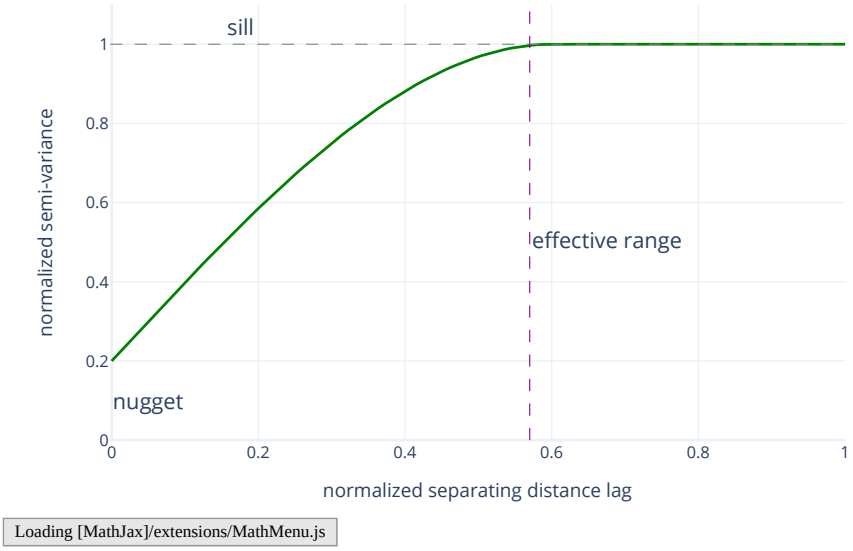
The physical environment is often represented as spatially distributed in geo-scientific models, which are usually only defined for a definable spatial context. A prime example is hydrological catchments, which are usually represented by their own parameters. Hence, universally applicable methods to analyze, harmonize, and re-scale observation data with respect to their spatial variability are needed. Geostatistical methods are well-established, widely used, and have proven to suit these needs for decades. Taking the interpolation of a variable as an example, the geostatistical approach is first fitting a mathematical model to the data, that describes spatial covariance. Secondly, weighted estimates at unobserved locations are based on this covariance to generate spatial coherent estimates, that respect

variability. In general, this approach is superior to purely mathematical interpolation methods, which impose a defined model, like spline interpolations. A spline interpolation will interpolate according to the polynomial function used, even if the represented model is not supported by the empirical data. In contrast, a geostatistical interpolation will fall back to mean values as the best guess for interpolation in the absence of a well-supported spatial model.

A core concept for geostatistical applications is therefore spatial variability. Statistical variability generally refers to the extent to which observations of the same population differ from each other. For the univariate case, it is usually measured as standard deviation or variance. For geostatistics in particular, the variability of a quantity across a spatial domain is considered. Here, these univariate measures are not suitable, as they i.e. assume independent observations. For natural phenomena, this is rarely the case as close observations are usually correlated to each other. As these observation are samples of the same population, this correlation is more precisely a spatial auto-correlation. Prime examples are meteorological parameters like air temperature, soil properties, hydraulic conductivity, or soil moisture. In these cases, the distance at which observations are not independent, but correlate, is often the focus of study as it can be related to the scale at which different natural processes are present. We refer to this distance as *correlation length*.

The aforementioned variations in a dataset are here interpreted as an inherent feature of observations. A natural process will manifest in variable quantities, often at different scales. On the contrary, the observation of this process can uncertain. While spatial variations and uncertain observations both result in increased variability of the sample, the distinction between both concepts is important. The variability of the process is usually part of the target variable. Geostatistics seeks to capture and model this spatial variability, while it is usually the goal to avoid uncertainties. In general terms, uncertainties can be described as variability due to a lack of complete knowledge about the real nature of a process. This can be related back to shortcomings or simplifications of a model concept or its structure itself, a lack of or imperfect data, or random variation. This work primarily focuses on uncertainties that are related back to the observation. An example is the precision and accuracy of the measurement, or the lack of knowledge of where, when, and how often to measure a quantity. These kinds of uncertainties are summarized as *observation uncertainty* here.

The spatial auto-correlation of a geostatistical dataset is described by a *semi-variogram* which is the foundation of almost any geostatistical application. The semi-variogram, also simply called variogram, is a spatial model that relates a measure for similarity (semi-variance) to the separating distance between two




Loading [MathJax]/extensions/MathMenu.js

**Figure 1.1:** Conceptual illustration of a spherical variogram model (green line) along with its three main parameters: nugget, sill and effective range. Separating distance and semi-variance are normalized.

observations (fig. 1.1). Empirical variograms are calculated for a sample and a model is fitted to the empirical variogram. Variogram models are positive definite and monotonously increasing up to a specific distance lag, which the model asymptotically approaches. The distance is called *effective range* and it corresponds to the correlation length of the model. The semi-variance of this length is called the *sill*. A variogram model may have a y-axis intersect, which is called *nugget*. In case of finite variance, nugget and sill (should) sum up to the sample variance together. With the help of the variogram it is possible to assign an expected semi-variance value to each separating distance, even if this distance is not included in the sample. Thus, the variogram model describes the expected covariance structure of the sample and can be used for simulating or interpolating while respecting the spatial auto-correlation. At the same time, the ratio between nugget and sill can be used to relate the overall variability of the sample to the share of variability, that is not captured by the model. In the shown conceptual example this is interpreted that 20% of the overall variability cannot be attributed to spatial auto-correlation.

This implies that a carefully estimated and in a geostatistical context meaningful

variogram is one of, if not the most crucial step of any geostatistical analysis or application workflow. In case the variogram does not expose a functional relationship between separating distance and observations of a dataset, that can explain most of the sample's variance, the application of any geostatistical method is not useful at all. With poorly fitted or unsupported variogram functions in use, any geostatistical interpolation will degenerate from a powerful tool to a resource-intensive visual gap-filler, that relies on spatial correlations, which are not supported by the sample. The variogram, as a statistical tool, is the main focus of this work.

The main consideration of variograms is to interpolate or simulate a quantity at unknown locations. The interpolation is referred to as *Kriging*, named after the inventor David Krige (Krige 1951). For kriging, the next (closest) observations are used to estimate a quantity at an unobserved location. The estimation is then the weighted sum and the weights are derived from a linear equation system that is built from the modeled semi-variance values for the corresponding separating distances. Hence, solely the theoretical model is used to impose the spatial correlation into the interpolation result, not the sample itself. Consequently, poorly supported or fitted models might still result in a spatial covariance present in the interpolation result. Kriging is referred to as BLUE - it is the best linear unbiased estimator. This is achieved by minimizing the prediction error and also requiring an expected value of zero for prediction errors. By utilizing the variogram, the spatial correlation is respected, and in case the neighboring observations are all outside the effective range, the kriging equation system will use the same weight for all neighbors, despite their actual distance. This is one of the key features of kriging, as this results in interpolated mean values in case the variogram does not exhibit any spatial structure (at that distance).

Beyond this interpolation, we use the variogram as a means to describe our dataset. By relaxing the concept of variograms to generic dispersion functions, we are able to detect recurrent dispersion function parameters. This is interpreted as a manifestation of the spatial correlation structure within the dataset. In chapter 2, we exemplify the use of dispersion functions beyond interpolation as we cluster time series data based on structural similarity. A thorough understanding of recurrent structures in the investigated system along with easily applicable software is the key to replicating the methodology to other datasets.

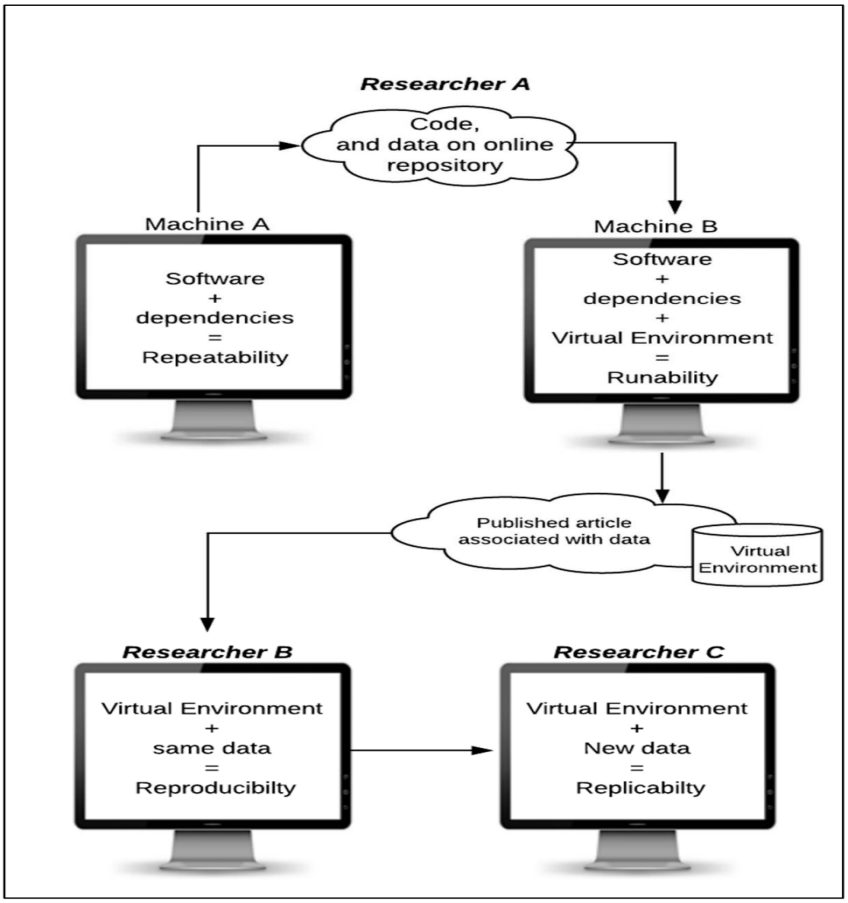



**Figure 1.2:** General steps necessary for ensuring technical reproducibility of research software. Figure taken from Essawy et al. (2020) (fig. 2)

## 1.2 Geostatistical research software

Scientific replicability and technical replicability are not the same. Synergies between both will be illustrated in this work, but scientific replicability can also limit the strictly technical aspects. Thus, concepts of replicability known from 'classic' software engineering need to be adapted before being applied to research software engineering. Unit testing is important to assure the technical correctness of different parts of software libraries, e.g. the variogram function value for any given variogram parameter set should always be the same, despite the programming language or environment used. A scientific replication of a method to i.e. a new dataset will imply, that variogram values are not exactly the same, but the

interpretation is. In chapter 2, we cluster a spatially distributed time series of soil moisture observations based on the spatial correlation structures emerging over time. Any cluster may be linked to a process altering the soil moisture, like evaporation, and characterized by its length and extent. Replicating the method to a new dataset might very well yield different lengths and extents, but indicate the same contextualized interpretation, pointing at the same process. Acknowledging this kind of scientific replicability cannot be covered by tests assuring (only) technical replicability.

Software development is becoming increasingly important in geoscience. Tools can be shared and used throughout the internet and an increasing number of scientific publications are accompanied by, or even about software. There are recognized journals in the environmental science domain available, that are dedicated to software and model development, namely *Geoscientific model development (GMD)* published by Copernicus or *Environmental Modelling & Software* published by Elsevier, to name just two.

Most (environmental) scientists do not have a formal software engineering background (Simm et al. 2018), hence tools, frameworks, concepts, and terminology develop in parallel and the field of research software engineering seems to be less structured than the 'classic' software engineering and computer science, which usually employ a more rigorous terminology and taxonomy. According to ISO/IEC 24765:2017 *software* is described as any program that is intended to be run on a computer system, including the program code, documentation, and data needed to run properly. This definition is too broad for research software engineering as it includes either all or hardly any program code written by (geo-)scientists, depending on how rigorously one demands comprehensive technical documentation and data distribution along with the software. From a software architecture point of view, this kind of software can be categorized into application software and end-user development, as further specified in ISO/IEC 2382-1:1993. This work will refer to application software if termed solely 'software', which is the collection of program code, documentation, and tests to run tasks reliably and reproducibly. This kind of software is usually realized in higher programming languages, in geoscience traditionally languages like FORTRAN, C, C++, Java or more recently also Python are used. These developments usually encapsulate models and data processing routines, are continuously developed, and form the basis for more contextualized and specific applications. In software engineering, this would be referred to as end-user developments. End-user development is usually realized through scripts written in so-called scripting languages, like Python, R, or Matlab, to name a few popular ones. These scripts are usually monolithic, system-dependent, and solve

specific problems, often coined by the scientists themselves (Simm et al. 2018). In the context of this work, this kind of software is referred to as *scripts*.

One of the main objectives to foster software engineering in geoscience is to enhance the replicability of current research (Baker 2016). Essawy et al. (2020) suggests a new taxonomy, that defines the often interchangeably used terms repeatable, runnable, reproducible, and replicable. I reference especially figure 1.2 for an overview, which was originally published in Essawy et al. (2020). In order to be repeatable, the researcher needs to define the program code, dependencies, and data that are necessary to obtain consistent results. Runability is achieved by containerizing the repeatable program, which will only become reproducible if another researcher can run the runnable version with the same data. In contrast, replicability covers the desired stage in which the containerized software can be fed with new data, in order to replicate the methodology in a new context. Hence, documentation and testing are core features of scientific software, that can help to move an end-user developed script on the replicability spectrum toward application software.

There are several approaches to increasing the replicability of scientific software. Simm et al. (2018) outlines some opportunities for software engineering, particularly in environmental modeling in this context. Namely, the demand for frameworks that abstract and formalize the underlying models. An exhaustive review of modeling frameworks is given in Chen et al. (2020), especially table 2 is helpful. Another approach, that is complementary to modeling frameworks is a higher degree of modularization and abstraction (Simm et al. 2018). Monolithic models need to be split into more generalizable model compartments, that at least separate the core algorithm from pre- and post-processing. This enables developers to combine and compare different compartments and domain-specific pre- and post-processing can be implemented in more domain-specific languages (ie. R), which in turn lets scientists work more effectively. Another approach is to enhance and formalize environmental models by using model application programming interfaces (APIs), which should roughly satisfy the same requirements (Y.-D. Choi et al. 2021).

## 1.3 Uncertainty analysis as common ground

It is quite evident, that this dissertation involves two main focus areas: geostatistics and research software engineering. From an abstract point of view, the link between both worlds is uncertainty. Geostatistics is fundamentally linked to spatial uncertainty, as the variogram is a spatially aggregated model used to statistically

obtain an estimate of a quantity. For the research software part, the overall aim is to improve reproducibility. The main means to achieve this are thorough and strict testing, to increase the technical correctness of applied software. Documentation aids researchers in understanding applied methods and comprehensive interfaces foster methodological standardization. Chapter A describes the development of the modular, comprehensive software library for variogram analysis, *SciKit-GStat*. Chapter 3 describes how this software library was extended to include sophisticated tools for uncertainty analysis in geostatistics.

With a good understanding of research software in the context of this work, I will present two studies, both exemplify the synergy between SciKit-GStat leveraging research and research extending the scope of the software. The first study (see chapter 2) investigates how recurrent patterns in multi-variate datasets emerge over time and exploits statistical dispersion to segment the data. SciKit-GStat could generalize the formal description of a semi-variogram model to the broader concept of statistical dispersion functions.

The second study presents a new approach representing empirical variograms as uncertainty bounds allowing for consistent propagation of observation uncertainties to interpolations and simulations (see chapter 3). This example is particularly interesting as it involves a high demand for various geostatistical models and algorithms, which were readily available through a number of research software solutions, including SciKit-GStat. At the same time, the study implemented new approaches for uncertainty propagation compatible with existing software. Being generalizable also includes the definition of generic interfaces for algorithms, which allowed for new usages beyond the original scope of the software. Both aforementioned studies are an example of the synergy between good science and good software, which not only leverages both but allowed the development of new methods to investigate a unique, multi-dimensional dataset from the Attert catchment in Luxembourg.

Chapter B describes the development of *Geostat API*, which makes SciKit-GStat, the uncertainty extension *SKGstat-Uncertainty* (chapter 3) and uncertainty-based dispersion function clustering (chapter 2)available, as reproducible and interoperable as possible. It is demonstrated how well-designed software fosters relevant scientific method development in the example of clustering spatial soil moisture patterns in the context of observation uncertainty. The modular nature of the exemplified reproducible geostatistical workflows allowed for rapid prototyping of an original method to visualize the covariance structure of a geostatistical sample beyond the scope of the original research as presented in chapter 2. Force-directed network graphs, as a method from other areas of research, are used as a means to

investigate the covariance structure of a sample beyond the aggregated information presented by an empirical variogram and associated models and their ensuing parameters.

# Part I

# Understanding dynamic pattern

# 2 Dynamic soil moisture patterns

# Soil Moisture - variable in space, redundant in time

The following chapter 2 is already published as a research article in *Hydrology and Earth System Science* as:

*Mälicke, M., Hassler, S. K., Blume, T., Weiler, M., and Zehe, E.: Soil moisture: variable in space but redundant in time, Hydrol. Earth Syst. Sci., 24, 2633–2653, https://doi.org/10.5194/hess-24-2633-2020, 2020.*

Author contributions:

The methodology was developed by me, supervised by EZ and discussed with SKH. The data was provided by TB and MW. All code was developed by me. The manuscript was written by myself, with contributions by EZ in the introduction and discussion. SKH supplied the field and data descriptions. The structure, narrative and language of the manuscript were revised and significantly improved by TB.

Data and Code availability:

Major parts of the analysis are based on the scipy (Virtanen et al. 2020), scikit-learn (Pedregosa et al. 2011a) and scikit-gstat package (Mirko Mälicke, Möller, et al. 2021a). All plots were generated using the matplotlib package (Hunter 2007b). The full analysis Python scripts are published on Github (https://github.com/mmaelicke/soil-moisture-dynamics-companion-code) (Mirko Mälicke 2019).

The data is available upon request.

## 2.1 Abstract

Soil moisture at the catchment scale exhibits a huge spatial variability. This suggests that even a large amount of observation points would not be able to capture soil moisture variability.
We present a measure to capture the spatial dissimilarity and its change over time. Statistical dispersion among observation points is related to their distance to describe spatial patterns. We analyzed the temporal evolution and emergence of these patterns and use the Mean shift clustering algorithm to identify and analyze clusters. We found that soil moisture observations from the 19.4 $km^2$ Colpach catchment in Luxembourg cluster in two fundamentally different states. On the one hand, we found rainfall-driven data clusters, usually characterized by strong relationships between dispersion and distance. Their spatial extent roughly matches the average hillslope length in the study area of about 500 m. On the other hand, we found clusters covering the vegetation period. In drying and then dry soil conditions there is no particular spatial dependence in soil moisture patterns and the values are highly similar beyond hillslope scale.

By combining uncertainty propagation with information theory, we were able to calculate the information content of spatial similarity with respect to measurement uncertainty (when are patterns different outside of uncertainty margins?). We were able to prove that the spatial information contained in soil moisture observations is highly redundant (differences in spatial patterns over time are within the error margins). Thus, they can be compressed (all cluster members can be substituted by one representative member) to only a fragment of the original data volume without significant information loss.
Our most interesting finding is that even a few soil moisture time series bear a considerable amount of information about dynamic changes of soil moisture. We argue that distributed soil moisture sampling reflects an organized catchment state, where soil moisture variability is not random. Thus, only a small amount of observation points is necessary to capture soil moisture dynamics.

## 2.2 Introduction

Although soil water is by far the smallest fresh water stock on earth, it plays a key role in the functioning of terrestrial ecosystems. Soil moisture controls (preferential) infiltration and runoff generation and is a limiting factor for vegetation growth. Plant-available soil water affects the Bowen ratio i.e. the partitioning of net radiation energy in latent and sensible heat, and last but not least it is an important control

for soil respiration and related trace gas emissions. Technologies and experimental strategies to observe soil water dynamics across scales have been at the core of the hydrological research agenda for more than 20 years (Topp, Davis, et al. 1982; Topp, Zebchuk, et al. 1984). Since these early studies published by Topp, spatially and temporally distributed Time Domain Reflectometry (TDR) and Frequency Domain Reflectometry (FDR) measurements have been widely used to characterize soil moisture dynamics at the transect (eg. T. Blume et al. 2009) , hillslope (eg. L. Brocca, Morbidelli, et al. 2007; Starr et al. 2002) and catchment scale (eg. Bronstert et al. 2012; Andrew W. Western, Zhou, et al. 2004). A common conclusion for the catchment scale is that soil moisture exhibits pronounced spatial variability and distributed point sampling often don't yield representative data for the catchment (see eg. L. Brocca, Tullo, et al. (2012) and E. Zehe, Graeff, et al. (2010) or numerous studies given in 2.2 of Vereecken et al., 2008).

Although large spatial variability seems to be a generic feature of soil moisture, there is also evidence that ranks of distributed soil moisture observations are largely stable in time as observed at the plot (Rolston et al. 1991; E. Zehe, Graeff, et al. 2010), hillslope (T. Blume et al. 2009; Brocca et al. 2009; L. Brocca, Morbidelli, et al. 2007), and even catchment scale (Rodger B. Grayson et al. 1997; Martínez-Fernández and Ceballos 2003). This rank stability, which is also often referred to as temporal stability (Vanderlinden et al. 2012), can i.e. be used to improve sensor networks (eg. Heathman et al. 2009) or select the most representative observation site in terms of soil moisture dynamics (eg. A. J. Teuling et al. 2006). In both cases rank stability assumes some kind of organization in the catchment, otherwise this representativity would not be observed.

Soil moisture dynamics have been subject to numerous review works (eg. Daly and Porporato 2005; Vereecken et al. 2008). More specifically, the temporal stability of soil moisture was reviewed by Vanderlinden et al. (2012). The authors analyzed a large number of studies with respect to the controls on time stability of soil water content (TS SWC), but yet "the basic question about TS SWC and its controls remain unanswered. Moreover, the evidence found in literature with respect to TS SWC controls remains contradictory" (Vanderlinden et al. 2012, p.2 l.2ff). We want to contribute by proposing a method that helps to understand how and when spatial soil moisture patterns are persistent.

Soil moisture responds to two main forcing regimes, namely rainfall driven wetting or radiation driven drying. The related controlling factors and processes differ strongly and operate at different spatial and temporal scales and the soil moisture pattern reflects thus the multitude of these influences (Bárdossy and Lehmann 1998). Hence, we hypothesise that periods in which different controlling factors were dominant are reflected in fundamentally different soil moisture patterns. This can

manifest itself in changes in the spatial covariance structure (Lark 2012; Schume et al. 2003), either in form of changing nugget to sill ratios (spatially explained variance) (E. Zehe, Graeff, et al. 2010) or state dependent variogram ranges (spatial extent of correlation) (Andrew W. Western, Zhou, et al. 2004). In a homogeneous, flat and non-vegetated landscape the soil moisture pattern shortly after a rainfall event would be the imprint of the precipitation pattern and provide predictive information about its spatial covariance. In contrast, in a heterogeneous landscape driven by spatially uniform block rain events, the spatial pattern of soil moisture would be a largely stable imprint of different landscape properties controlling through-fall, infiltration as well as vertical and lateral soil water redistribution. Without further forcing, the spatial pattern will gradually dissipate due to soil water potential depletion and by lateral soil water flows. We therefore hypothesize that differences in soil moisture (across space) are higher shortly after a rainfall event and are dissipated afterwards.

Landscape heterogeneity is thus a perquisite for temporarily persistent spatial patterns found in a set of soil moisture time series. While most catchments are strongly heterogeneous, it is striking how spatially organized they are (Bras 2015; Dooge 1986; McDonnell et al. 2007; Sivapalan 2003; Erwin Zehe, Ehret, et al. 2014). Spatial organization manifests for instance through systematic and structured patterns of catchment properties, such as a catena. This might naturally lead to a systematic variability of those processes controlling wetting and drying of the soil. One approach to diagnose and model systematic variability is based on the covariance between observations in relation to their separating distance (Burgess and R. Webster 1980) and geo-statistical interpolation or simulations methods (Kitanidis and Vomvoris 1983; S. Ly et al. 2011; Pool et al. 2015).

A spatial covariance function describes how linear statistical dependence of observations declines with increasing separating distance up to the distance of statistical independence. This is often expressed as experimental variogram. Geo-statistics relies on several assumptions such as second order stationarity (see e.g. Lark (2012) or Burgess and R. Webster (1980)), which are ultimately important for interpolation. Due to the above-mentioned dynamic nature of soil moisture observations, the most promising avenue for interpolation would be a spatio-temporal geostatistical modeling of our data (De Cesare et al. 2002; Jost et al. 2005; Ma 2002; Ma 2003; Snepvangers et al. 2003).

However, here we take a different avenue, as we do not intend to interpolate. One of our goals is to detect dynamic changes in the spatial soil moisture pattern. Following Sampson and Guttorp (1992) we relate the statistical dispersion of soil moisture observations to their separating distance to characterize how their similarity and predictive information declines with this distance (see section 2.3). More

specifically, we analyze temporal changes in the spatial dispersion of distributed soil moisture data and hypothesize that a grouping of the data is possible solely based on the changes in spatial dispersion. We want to find out whether typical patterns emerge in time, how those relate to the different forcing regimes and whether those patterns are recurrent in time. The latter is an indicator for predictability and (self) - organization in dynamic systems (Wendi and Marwan 2018; Wendi, Marwan, and Merz 2018).

Erwin Zehe, Ehret, et al. (2014) argued that spatial organization manifests through a similar hydrological functioning. This is in line with the idea of Wagener et al. (2007) on catchment classification, or the early idea of a geomorphological unit hydrograph (Patil and Stieglitz 2012; Rodríguez-Iturbe et al. 1979; Sivapalan et al. 2011). Recently, R. Loritz et al. (2018) corroborated the idea of Erwin Zehe, Ehret, et al. (2014) and showed that hydrological similarity of discharge time series implies that they are redundant. Redundancy in our context means that new observations (over time) do not add significant new information to the data set of spatial dispersion. Thus, they can be compressed without information loss (Weijs et al. 2013). This combination of compression rate and information loss is understood to be a measure of spatial organization in our work. More specifically, R. Loritz et al. (2018) showed that a set of 105 hillslope models yielded, despite their strong differences in topography, a strongly redundant runoff response. Using Shannon entropy (Shannon 1948) R. Loritz et al. (2018) showed that the ensemble could be compressed to a set of 6 to 8 typical hillslopes without performance loss. Here we adopt this idea and investigate the redundancy of patterns in spatially distributed soil moisture data along with their compressibility.

The core objective of this study is to provide evidence that distributed soil moisture time series provide, despite their strong spatial variability, representative information on soil moisture dynamics. More specifically, we test the following hypotheses:

- **H1:** Radiation-driven drying and rainfall-driven wetting leave different fingerprints in the soil moisture pattern.
- **H2:** Both forcing regimes and their seasonal variability may be identified through temporal clustering of dispersion functions.
- **H3:** Spatial dispersion is more pronounced during and shortly after rainfall driven wetting conditions.
- **H4:** Soil moisture time series are redundant, which implies they are compressible without information loss. However, the degree of compressibility is changing over time.

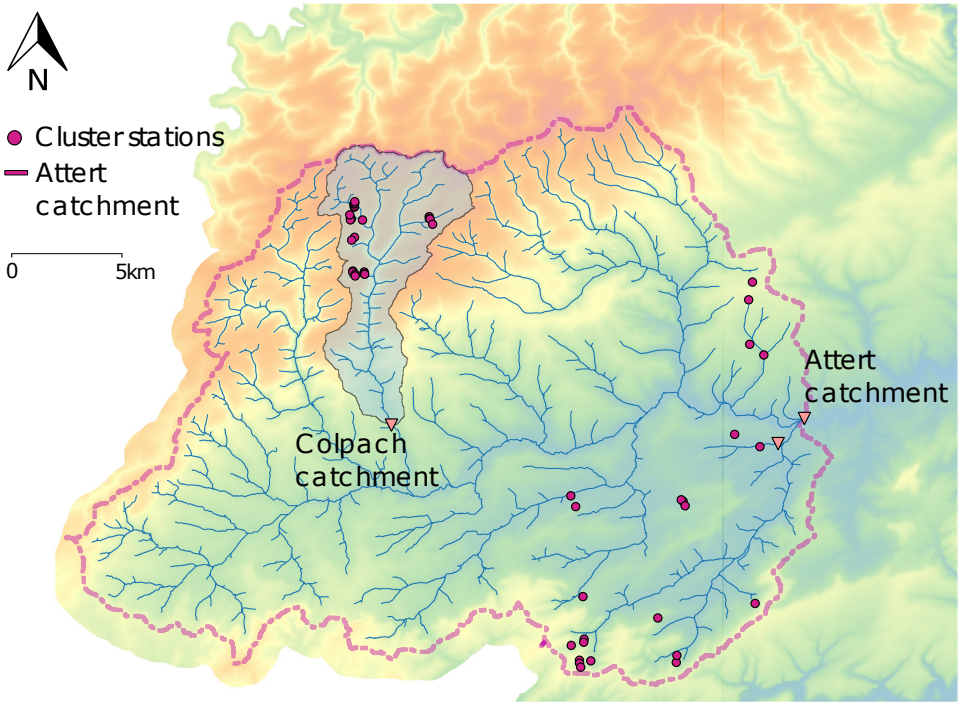



**Figure 2.1:** Attert experimental catchment in Luxembourg and Belgium. The purple dots show the sensor cluster stations installed during the CAOS project. Here we focus on those cluster stations within the Colpach catchment. Figure adapted after Ralf Loritz et al. (2017).

We test these hypotheses using a distributed soil moisture data set collected in the Colpach catchment in Luxembourg. In section 2.3 we give an overview of the study site and our method. The results section consists of three parts: spatial dispersion functions, temporal patterns in their emergence and some insights on generalization (or compressibility) of these functions, followed by a discussion and summary.

## 2.3 Methods

### 2.3.1 Study area and soil moisture data set

We base our analyses on the CAOS data set, which was collected in the Attert experimental watershed between 2012 and 2017 and is explained in Erwin Zehe, Ehret, et al. (2014). The Attert catchment is situated in western Luxembourg and Belgium (Figure 2.1). Mean monthly temperatures range from 18 °C in July to 0 °C

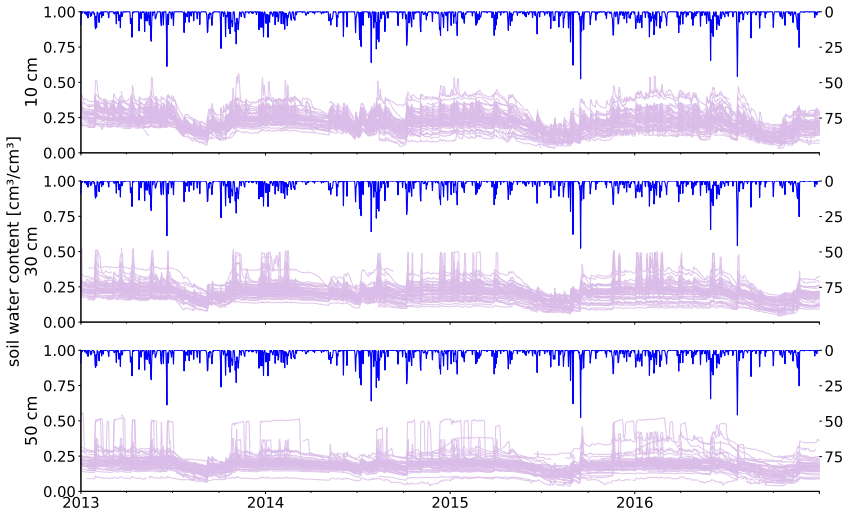



**Figure 2.2:** Soil moisture data overview. Soil moisture observations in 10 cm (top), 30 cm (middle) and 50 cm (bottom).

in January. Mean annual precipitation is approximately 850 mm (Laurent Pfister et al. 2000). The catchment covers three geological formations, Devonian schists of the Ardennes massif in the northwest, a mixture of Triassic sandy marls in the center and a small area on Luxembourg Sandstone on the southern catchment border (Martinez-Carreras et al. 2012). The respective soils in the three areas are haplic Cambisols in the schist, different types of Stagnosols in the marls area and Arenosols in the sandstone (IUSS Working Group 2006; Sprenger et al. 2016). The distinct differences in geology are also reflected in topography and land use. In the schist area, land use is mainly forest on steep slopes of the valleys, which intersect plateaus that are used for agriculture and pastures. The marls area has very gentle slopes and is mainly used for pastures and agriculture, while the sandstone area is forested on steep topography.

The experimental design is based on spatially distributed, clustered point measurements within replicated hillslopes. Typical hillslope lengths vary between 400 and 600 m showing maximum elevations of 50 to 100 m above stream level. For further details on the hillslopes we refer to Figure 6a in Ralf Loritz et al. (2017) and a detailed description in section 3.1.1 of the same publication. Sensor clusters were installed on hillslopes at the top, midslope and hill foot sector along the

anticipated flow paths. Within each of those clusters, soil moisture was recorded in three profiles in 10, 30 and 50 cm depth using Decagon 5TE sensors. While the entire design was stratified to sample different geological settings (schist, marls, sandstone), different aspects and land use (deciduous forest and pasture), we focus here on those sensors installed in the Colpach catchment. In total we used 19 sensor cluster locations and thus 57 soil moisture profiles consisting of 171 time series.

Soil moisture in the 19.4 $km^2$ Colpach catchment exhibits high but temporally persistent spatial variability (Fig. 2.2). For each point in time a wide range of water content values can be observed across the catchment. The range of soil moisture observations is generally wider in winter than in summer. From visual inspection it seems that the heterogeneity in observations is not purely random but systematic as the measurements are rank stable over long periods. One has to note that the different cluster locations differ in is aspect, slope and landuse. From the data shown in figure 2.2, two sensors have been removed. Both measured in 50 cm and can be seen in the figure at the very bottom. Both recorded values close to or even below $0.1 cm^3 cm^{-}3$ for the whole period of four years. Additionally, the plateaus lasting for a couple of days at constant $0.5 cm^3 cm^{-}3$ in 50 cm and 30 cm were removed.

### 2.3.2 Dispersion of soil moisture observations as function of their distance

We focus on spatial patterns of soil moisture and how they change over time. For our analysis the data set was aggregated to mean daily soil moisture values $\theta$. Each time series is further aggregated using a moving window of one month as described by equation (2.1).

$$z_x(t) = \frac{\sum_t^{t+b} \theta_x}{b} \tag{2.1}$$

This is calculated for each observation location $x$ and time step $t = 1, 2, \ldots, (L-b)$, with a time series length of $L$ in days and a window size of $b = 30$ [1] .

To estimate the spatial dependence structure between observations, we relate their pairwise separation distance to a measure of pairwise similarity. Here, we further define the statistical spatial dispersion to be a measure of spatial similarity. We compare the empirical distribution of pairwise value differences at different

1 We tested different window sizes, as we expect that different processes control the emergence of spatial dependence at different temporal scales. The chosen window size was most suitable to detect seasonal effects.

distances. Statistically, a more dispersed empirical distribution is less well described by its mean value. Thus, observations taken at a specific distance are more similar in value, if they are less dispersed.

To estimate the dispersion, we use the Cressie-Hawkins estimator (Cressie and Hawkins 1980). This estimator is more robust to extreme values and the contained power transformation handles skewed data better than estimators based on the arithmetic mean (Bárdossy and Kundzewicz 1990; Cressie and Hawkins 1980). The estimator is given by equation (2.2):

$$a_t(h) = \frac{1}{2}\left(\frac{1}{N(h)}\sum_{i,j}\sqrt{|z_t(x_i) - z_t(x_j)|}\right)^4 \left(0.457 + \frac{0.494}{N(h)} + \frac{0.045}{N^2(h)}\right)^{-1} \tag{2.2}$$

for each moving window position $t$ with $z_t(x_{i,j})$ given by equation (2.1) for each pair of observation locations $x_i, x_j$. $h$ is the separating distance lag between these points pairs and $N(h)$ the number of points pairs formed at the given lag $h$. 10 classes were formed with a maximum separation distance of 1200 $m^2$. The lag classes are not equidistant, but with a fixed $N(h)$ for all classes. This is further discussed in section 2.3.3.

### 2.3.3 Clustering of dispersion functions

We analyzed how and if meaningful spatial dispersion functions emerge and whether those converge into stable configurations. To tackle the hypotheses formulated in the introduction a clustering is applied to the dispersion functions derived for each window. The clustering algorithm should form groups of functions that are more similar to each other than to members of other clusters. The similarity between two dispersion functions is calculated by the Euclidean vector distance between the dispersion values forming the function. This distance is defined by equation (2.3):

$$d(\vec{u}, \vec{v}) = \sqrt{(\vec{u} - \vec{v})^2} \tag{2.3}$$

with $\vec{u}, \vec{v}$ being two dispersion function vectors. This is the Euclidean distance of two points in the (higher dimensional) value space of the dispersion function's distance lags. Two identical dispersion functions are represented by the same point in this value space and hence their distance is zero. Thus, distance lags are not equidistant, as this could lead to empty lag classes. Empty lag classes result in an undefined

2 Observation point pairs further apart than 1200 m are most likely located on different hillslopes. These points might share similar soil, topographic and terrain aspect characteristics. Soil misture dynamics might thus be similar, although they are located at rather lage separating distances

position in the value space, which has to be avoided. The clustering algorithm cannot use the number of clusters as a parameter, as this can hardly be determined a priori. One clustering algorithm meeting these requirements is the *Mean shift* algorithm (Fukunaga and Hostetler 1975). The actual code implementation is taken from Pedregosa et al. (2011a), which follows the Comaniciu and Meer (2002) variant of Mean shift. A detailed description of the Mean shift algorithm can be found in the appendix (see 2.7.1).

### 2.3.4 Cluster compression based on the cluster centroids

The next step is to generate a representative dispersion function for each cluster. The straightforward representative function is the cluster centroid (the dispersion function closest to the point of highest cluster member density, see appendix 2.7.1 for a detailed explanation). All dispersion functions are calculated with the same parameters, including the maximum separating distance of 1200 m. At larger lags we found instances of declining dispersion values, because we then paired points located on different hillslopes, but otherwise in similar landscape units (i.e. same hillslope position or land use). To facilitate the comparison of the dispersion functions we decided to monotonize them. In geostatistics this is usually done through fitting of a theoretical variogram model to the experimental variogram, which assures monotony and positive definiteness. Here we do not force a specific shape by a fitting a model function. Instead we use the technique of monotonizing the cluster centroid as suggested by Hinterding (2003) using the PAVA-Algorithm (Barlow et al. 1972). The implementation is from Pedregosa et al. (2011a). This way, the final compressed dispersion functions are monotonically increasing, while still reflecting the shape properties of the cluster members. If dispersion functions are monotonically increasing, they also provide information about the characteristic length of the soil moisture pattern. Similar as for the semi-variogram in geostatistics this characteristic length corresponds the the lag distance where the dispersion function reaches its first local maximum.

We suggest that the number of clusters needed to represent all observed spatial dispersion functions over a calendar year can be used as a measure of spatial organization (fewer clusters needed means higher degree of organization, because dispersion functions are redundant in time). Additionally, it is insightful to judge the information loss that goes along with this compression, as a high compression with little information loss is understood as a manifestation of spatial and temporal organization of soil moisture dynamics.

In line with R. Loritz et al. (2018) we use the Shannon Entropy as measure for the compression without information loss. It requires treatment of the clusters

as discrete probability density functions, which in turn implies a careful selection of an appropriate classification of the data. Motivated by R. Loritz et al. (2018), we use the uncertainty in the dispersion function as a minimal class size for this classification, as described in section 2.3.5.

### 2.3.5 Uncertainty propagation and compression quality

#### Uncertainty propagation

Soil moisture measurements have considerable measurement uncertainty of 1 - 3 $cm^3cm^{-3}$ as reported by manufacturers. For our uncertainty propagation we assume an absolute uncertainty/measurement error $\Delta\theta$ of 0.02 $cm^3cm^{-3}$.

Next we propagate these uncertainties into the dispersion functions and the distances among those. As we assume the measurement uncertainties to be statistically independent, we use the Gaussian uncertainty propagation to calculate error bands / margins . In a general form, for any function $f(z)$ and an absolute error $\Delta z$ the propagated error $\Delta f$ can be calculated. In our case $z$ is itself a function of $x$, the observation location, and the general form is given by equation (2.4).

$$\Delta f = \sqrt{\sum_{i=1}^{N} \left( \frac{\partial f}{\partial z(x_i)} \Delta z(x_i) \right)^2} \tag{2.4}$$

To apply equation (2.4) for our method, the measurement uncertainty $\Delta\theta$ is propagated into the dispersion estimator given by equation (2.2). The dispersion estimator is derived with respect to $z(x)$ and following equation (2.1), the uncertainty in $z(x)$, $\Delta z$, is denoted as $\Delta z = \Delta\theta = 0.02 cm^3 cm^{-}3$. Then, with given $\Delta z$, we can propagate the uncertainty into the dispersion function. As the dispersion function is a function of the spatial lag $h$, we need to propagate the uncertainty $\Delta a$ (uncertainty of dispersion estimator) for each value of $h$. At the same time, following equation (2.2), for each $h$, $z(x_i) - z(x_i + h)$ is a fixed set of point pairs. Instead of propagating uncertainty through equation (2.2), we can substitute $z(x_i) - z(x_i + h)$ by $\delta$, the pairwise differences, for each value of $h$. The uncertainty $\Delta\delta$ is given by equation (2.5)

$$\Delta\delta = \sqrt{\Delta z_i^2 + \Delta z_{i+h}^2} = \sqrt{2}\Delta z \tag{2.5}$$

The uncertainty of dispersion $\Delta a$ is then defined by equation (2.6):

$$\delta a = \frac{\partial a}{\partial \delta} \Delta\delta \tag{2.6}$$

$$= 2c\left(\frac{1}{N}\sum_{i=1}^{N}(|\delta_i|)^{\frac{1}{2}}\right)^3 \cdot \frac{1}{N}\left(\sum_{i=1}^{N}|\delta_i|^{-}1\right)^{\frac{1}{2}} \cdot \Delta\delta \tag{2.7}$$

where the factors from equation (2.2) that stay constant in the derivative are denoted as $c$ and defined in equation (2.8). In line with equation (2.2) $N$ is the number of observation pairs available for a given lag class $h$ and therefore constant for a single calculation. $\Delta\delta$ and $\delta$ are the substitutes for $z$, as described above (see equation (2.5)).

$$c = \frac{1}{2} * (0.457 + \frac{1}{N} + \frac{0.045}{N^2})^{-1} \tag{2.8}$$

The last step is to propagate the uncertainty into the distance function as defined in equation (2.3). The Euclidean distance is used as a measure for proximity by Mean shift, as it groups dispersion functions at short distances together (for more details see appendix section 2.7.1). At the same time, we use the uncertainty propagated into the Euclidean distance between two dispersion functions to assess compression quality (as further described in section 2.3.5). Following equation (2.4) the propagated uncertainty $\Delta D$ can be calculated by the derivative of equation 2.3 with respect to each of the vectors multiplied by the corresponding value of $\Delta a$, which results in equation (2.9):

$$\Delta d_{\vec{u},\vec{v}} = \sqrt{\left(\frac{\delta d}{\delta \vec{u}}\vec{\Delta u}\right)^2 + \left(\frac{\delta d}{\delta \vec{u}}\vec{\Delta u}\right)^2} \tag{2.9}$$

$$= \sqrt{\frac{1}{2}\sum_{i=1}^{n}(|\vec{u}-\vec{v}|^{\frac{1}{2}})\sum_{i=1}^{n}((2(|\vec{u}_i-\vec{v}_i|)\Delta\vec{u}_i)^2 + (2(|\vec{u}_i-\vec{v}_i|)\Delta\vec{v}_i)^2)} \tag{2.10}$$

Where $\vec{u},\vec{v}$ are two spatial dispersion function vectors as defined and used in equation (2.3). $\vec{\Delta u},\vec{\Delta v}$ are the vectors of uncertainties for $\vec{u},\vec{v}$, where $\Delta v_i$ is the uncertainty propagated into the $i^{th}$ lag class as shown in equation (2.6). $n$ is the number of lag classes and thus the length of each vector $\vec{u},\vec{v},\vec{\Delta u},\vec{\Delta v}$.

Equation (2.9) is applied to all possible combinations of dispersion functions $\vec{u},\vec{v}$ to get all possible uncertainties in dispersion function distances.

### Compression quality

The Shannon entropy (Shannon 1948) of all pairwise dispersion function distances is used as measure for information content. The Shannon entropy of a discrete

probability density function of states (patterns in this case) is maximized for the uniform distribution. It corresponds to the number of yes/no questions one has to ask to determine the state of a system. The minimum entropy is zero, which corresponds to the deterministic case that the system state is always known. A common way to define spatial organisation of a physical system is through its distance from the maximum entropy state (Kleidon 2012; Kondepudi and Prigogine 1998). The deviation of the entropy of the dispersion functions in a cluster from its maximum value is thus a measure for their redundancy and thus similarity.

For a discrete frequency distribution of $n$ bins, the information entropy $H$ is defined as:

$$H = -\sum_{n} p_n log_2(p_n) \tag{2.11}$$

Where $p_n$ is the relative probability of the $n$th bin. $H$ is calculated for each depth in each year individually to compare the information content across years and depths. Note that the term *bin* is also used in literature to refer to the *binning* of pairwise data, e.g. in geo-statistics. For this kind of binning, although technically the same thing, we used the term *lag classes* here, to distinguish from the binning as shown in equation (2.11). Thus, when we write *bin* or *binning* we refer to the classification of distances between dispersion functions, not observation points.

To assure comparability we use one binning for all calculations of $H$ (across years and depths). To achieve this, all pairwise distances between all spatial dispersion functions of all four years in all three depths are calculated. The discrete frequency distribution is formed from 0 up to the global maximum distance (between two dispersion functions) calculated using equation (2.3). The bins are formed equidistant using a width of the maximum function distance that still lies within the error margins calculated using equation (2.9). Thus, the information content of the spatial heterogeneity is calculated with respect to the expected uncertainties. This way we can be sure to distinguish exclusively those spatial dispersion functions that lie outside of the error margins.

The Kullback-Leibler divergence (Kullback and Leibler 1951) is a measure for the difference between two empirical, discrete probability distributions. Usually, one distribution is considered to be the population and the other one a sample from it. The Kullback-Leibler divergence $D_{KL}$ then quantifies the uncertainty introduced (e.g. in an statistical model) using a sample as a substitute for the population.

We use the Kullback-Leibler divergence to measure and quantify the information loss due to compression. To compress the series of dispersion functions, each cluster member is expressed by its centroid function. Now, we need to calculate the amount of information lost in this process. To calculate the mean informa-

tion content of the compressed series each cluster member is substituted by the respective cluster centroid. This substitution is obviously not a compression in a technical sense, but necessary to calculate the Kullback-Leibler divergence. Then a frequency distribution for compressed series $X$ and the uncompressed series $Y$ can be calculated. The Kullback-Leibler divergence $D_{KL}$ of $X, Y$ is given in equation (2.12):

$$D_{KL}(X, Y) = H(X||Y) - H(Y) \tag{2.12}$$

Where $H(X||Y)$ is the cross entropy of $X$ and $Y$ and defined by equation (2.13):

$$H(X||Y) = \sum_{x \in X} p(x) * log_2 p(y) \tag{2.13}$$

Where $p(x)$ is the empirical non-exceedance probability of the frequency distribution $X$ and $p(y)$ of $Y$, respectively.

## 2.4 Results

### 2.4.1 Dispersion functions over time

Figure 2.3 a) shows the spatial dispersion functions for all moving window positions in 2016 for the 30 cm sensors. The position of the moving window in time can be retraced by the line color, darker red means a later Julian day. Each of the spatial dispersion functions relates the dispersion for all pairwise observations to their separating distance in the corresponding lag class. Dispersion increases with separating distance, as small values correspond to observations which have similar values while large values suggest the opposite. As expected, the dispersion is a suitable metric for similarity/dependency of observations.

The spatial dispersion functions take several distinct shapes with each of these shapes occurring during a certain period in time. More specifically from Figure 2.3 a) one can identify groups of functions of similar reds plotting close to each other. Dispersion functions of similar red saturation, which reflects proximity in time, are also similar in shape, and this in turn reflects similar spatial patterns. Similar dispersion functions were grouped using the Mean shift clustering algorithm (Fig. 2.3 b); here, the color indicates the cluster membership.

To provide further insight on the temporal occurrence of cluster members, we colored the soil moisture time series according to the color codes of the identified clusters (Fig. 2.3 d). The blue parts of the soil moisture time series were classified into Cluster #1, while the the orange was classified into Cluster #2. Note that cluster

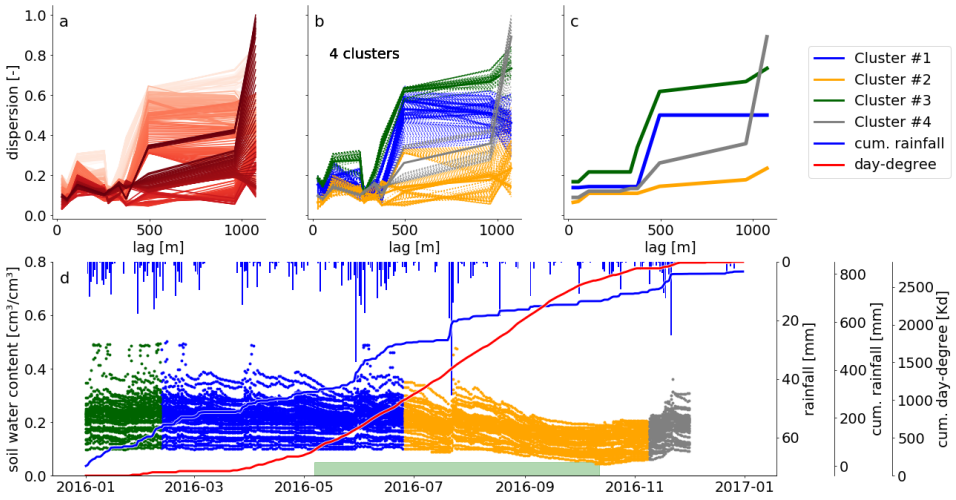



**Figure 2.3:** Spatial dispersion functions in 30 cm for 2016 based on on a window size of 30 days.
**a):** Spatial dispersion function for each position of the moving window. The red color saturation is indicating the window position. The darker the red the later in the year.
**b):** The same dispersion functions as presented in a). Here the color indicates cluster membership as identified by the Mean shift algorithm.
**c):** Compressed spatial dispersion information represented by corrected cluster centroids. The colors match the clusters as presented in b).
**d):** Soil moisture time series of 2016 in 30 cm depth. The colors identify the cluster membership of the spatial dispersion function of the current window location and is matching the colors in b) and c). The bars on the top show the daily precipitation sums. The solid blue line is the cumulative daily precipitation sum and the red line the cumulative sum of all mean daily temperatures > 5 °C. The green bar marks the assumed vegetation period. It covers the dates where the cumulative day-degree sum is > 15% & < 90% of the maximum.

memberships are constant for long periods of time, which means that also the soil moisture patterns are persistent over these periods. Exact cluster lifespans can be found in table 2.3. We could identify four clusters in 30 cm, with the orange cluster roughly occurring during the vegetation period and the other three the remaining time of the year. As new observations did not change the patterns during these periods, they were redundant in time.

As the spatial dispersion functions in the presented example are redundant in time, we compressed the information by replacing the dispersion function within one cluster by the cluster centroid. All four representative functions shown in Figure 2.3 c) exhibit increasing dispersion with separating distance. For the blue and green cluster this happens step-wise at a characteristic distance of 500 m. That reminds us of a Gaussian variogram, which can also show a step-wise characteristic. The small grey cluster shows an increase at 500 and another one at 1000 m separating distance. In contrast the orange cluster, however, shows a only a gentle increase with distance.

In the vegetation period observations are similar even at large separating distances. Interestingly, dispersion functions in the orange cluster start with small values that only gently increase with separating distance. That means soil moisture becomes more homogeneous. Outside of the vegetation period, different spatial patterns can be observed, with increasing dissimilarity with separating distances. The part of the blue cluster overlapping with vegetation period shows still higher soil moisture values. The transition to the orange cluster sets in as the soil moisture drops (Fig. 2.3 d). This suggests that vegetation influences, such as root water uptake, smooth out variability in soil water content, leading to a more homogeneous pattern in space, as further discussed in section 2.5.3.

### 2.4.2 Dispersion time series as a function of depth

Figure 2.4 shows the time series of the dispersion functions for all depths. Note that the coloring between the sub-figure is arbitrary, due to Mean shift, that means there is no connection between the orange cluster between the three figures.

In comparison to the dispersion functions in 30 cm (Fig. 2.4 b) the soil moisture signal in 10 cm (Fig. 2.4 a) is more variable in time. A look at the centroid of the orange cluster (Fig. 2.4 d) reveals a higher spatial heterogeneity in winter and spring at large separating distances. At the same time the observations get spatially more homogeneous in summer, particularly when the blue cluster emerges, i.e. the dispersion at large lags decreases significantly. We can still find a summer-recession cluster in 10 cm, but compared to the depth of 30 cm we also find this spatial footprint of continuous drying earlier in the year around May. This is likely due to

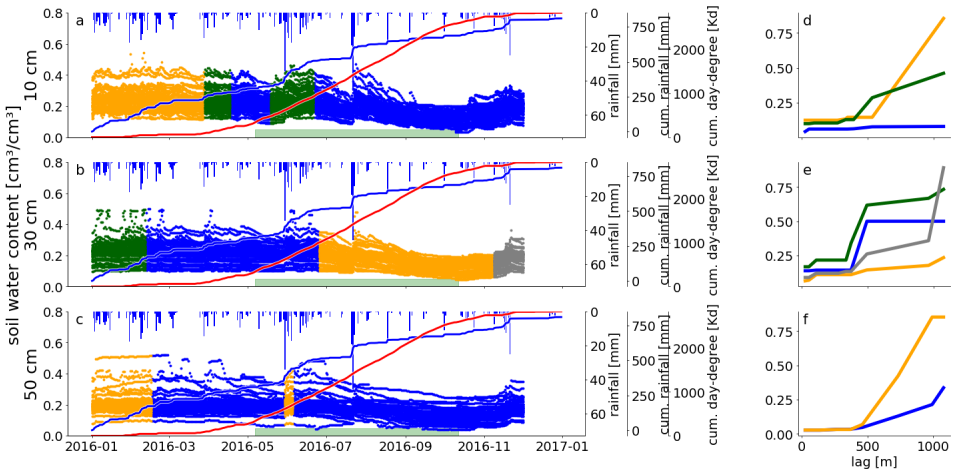



**Figure 2.4:** Soil moisture time-series of 2016 in all three depths with respective cluster centroids. The three rows show the data from 10 cm (a), 30 cm (b) and 50 cm (c). The colors indicate the cluster membership of the corresponding dispersion function of the respective window position. The green bar marks the assumed vegetation period. It covers the dates where the cumulative day-degree sum is > 15% & < 90% of the maximum. The cluster centroids for each depth is shown in d, e and f.

a higher sensitivity to rising temperatures. Note that during May there was only little rainfall and the soil moisture is already declining. This blue cluster shows very small dispersion values for all separating distance classes (Fig. 2.4 d), just as the orange cluster in 30 cm depth.

The green clusters emerge with strong rainfall events after longer previous dry spells (Fig. 2.4, a,d). We would have expected a third occurrence at the beginning of August, but the soil may already be too dry to bear a detectable dependency on separating distance (Remember that the blue cluster does not show increasing dispersion with distance).

Observations at 50 cm depth show a clear spatial dependency throughout the whole year. We cannot identify a summer cluster, Mean shift yielded two clusters and rainfall forcing does not have a clear influence on their occurrence or transition. The two 50 cm dispersion functions (Fig. 2.4 f) show a clear dependence on distance, but they differ in their dispersion value at large lags. In 10 cm and 30 cm we found dispersion functions of fundamentally different shape, like the flat, blue function (Fig. 2.4, d) or the step-wise blue and green functions (Fig. 2.4 e). In 50 cm depth the characteristic length is 500 m and the blue cluster persists throughout most of the year (282 days, see table 2.3). The orange cluster occurs during the cool and wet start of the year, showing a larger dispersion and thus stronger dissimilarity

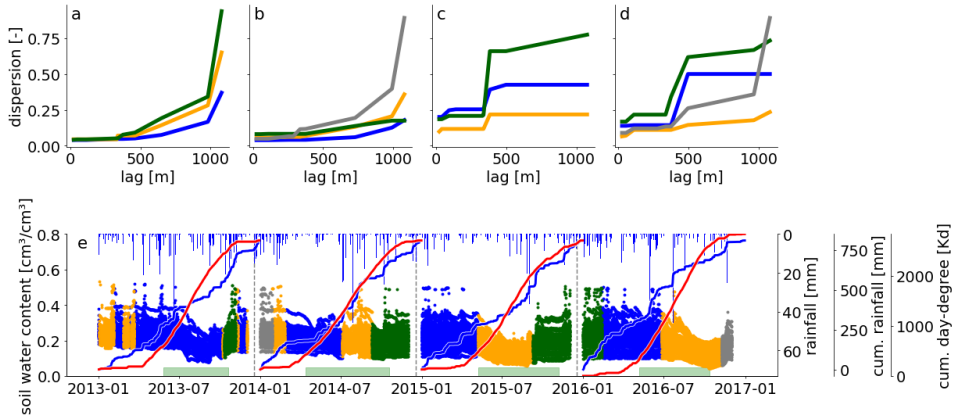



**Figure 2.5:** Soil moisture timeseries of all years in 30 cm depth (e) and the respective cluster centroids (a-d). The colors of the soil moisture data indicate the cluster membership of the corresponding dispersion function of the respective window position and correspond with the color of the cluster centroid (in a-d). The cumulative rainfalls (blue) and cumulative temperature sums (red) are shown for each year individually. The green bar marks the assumed vegetation period. It covers the dates where the cumulative day-degree sum is > 15% & < 90% of the maximum.

at larger lags (Fig. 2.4 f). Interestingly this cluster occurs again in early June after an intense rainfall period. However, a similar rainfall period in August does not trigger the emergence of this orange cluster as the top soil above 50 cm is so dry, that even this strong wetting signal does not reach the depth of 50 cm (Fig. 2.4 c). This behavior reveals the low pass behavior of the top soil, which causes a strong decoupling of the soil moisture pattern in 50 cm depth from event scale changes.

### 2.4.3 Recurring spatial dispersion over the years

Table 2.1 summarizes the most important features of the clustering for all observation depths. Soil moisture patterns and their clustering appear generally to be clearer for 2015 and 2016. The vegetation period is more often characterized by a typical cluster and dispersion functions more often reveal a clear spatial dependency. In some cases (10 cm, 2013 and 2014) no spatial dependency of dispersion functions could be observed throughout the whole year. Less clusters were formed in 2015 and 2016. Note that annual rainfall sums were higher in 2013 and 2014, while 2015 and 2016 had significantly more precipitation in the first half of the year followed by a dry summer (compared to 2013 and 2014).

To further illuminate inter annual changes in soil moisture patterns we present

**Table 2.1:** Qualitative description of method success in all years and depths. The results from years other than 2016 and all depths were inspected visually and are summarized here for sake of completeness. The first three columns identify the year, sensor depth and the number of clusters found by Mean shift. The remaining three columns state if specific features existed in the given result. *Vegetation period* marks whether or not the vegetation period was characterized by a single, or two, clusters. *Spatial structure*: Does a dependency of dispersion on distance exist **outside** the vegetation period? *rainfall transition*: Were cluster transitions accompanied by a rainfall event in close (temporal) proximity? This feature is marked 'yes' if it was more often the case than it was not.

| Year | depth | # of clusters | vegetation period | spatial structure | rainfall transition |
|---|---|---|---|---|---|
| 2013 | 10 | 4 | yes | no | no |
| 2013 | 30 | 3 | no | yes | yes |
| 2013 | 50 | 6 | no | yes | yes |
| 2014 | 10 | 3 | yes | no | yes |
| 2014 | 30 | 4 | no | yes | no |
| 2014 | 50 | 5 | no | yes | yes |
| 2015 | 10 | 3 | yes | yes | no |
| 2015 | 30 | 3 | yes | yes | yes |
| 2015 | 50 | 3 | no | yes | no |
| 2016 | 10 | 3 | no | yes | yes |
| 2016 | 30 | 4 | yes | yes | yes |
| 2016 | 50 | 2 | no | yes | no |

the time series of cluster memberships for the sensors in 30 cm for the entire monitoring period in Figure 2.5. From this example it becomes obvious that patterns are recurring. 2013 and 2014 cluster centroids look different from the following two years. Dispersion values increase with distance in all centroids in 2013 and 2014, while 2015 and 2016 show a sudden increase at 400 - 500 m (Fig. 2.5 a-d). 2015 and 2016 are segmented by Mean shift in a similar way, and cluster centroids reveal that the green clusters in both years are actually the same. This green cluster emerges with the occurrence of the largest rainfall event in the observation period and lasts for around 5 months. All dispersion functions within this cluster look nearly identical (see Appendix, Fig. 2.9, b). Similar observations can be made between 2014 and 2015. Here, the green and blue cluster seem to be an inter-annual cluster. However, in contrast to to 2015/2016 the dispersion functions here are of different shape (see Appendix, Fig. 2.8, b). Hence, the cluster transition indicated between 2014 and 2015 is indeed a real transition. When looking at cluster memberships throughout the whole period, the division into calendar years is rather meaningless, while the division in hydrological years is much more appropriate, as it is reflected by the cluster membership and its changes.

Distinct summer recessions in soil moisture are only identified in 2015 and 2016. Evapotranspiration (indicated by the cumulative temperature curves in Fig. 2.5 e) is dominating over rainfall input (blue sum curve) in the soil moisture signal. Mean shift could identify a significantly distinct spatial dependency in dispersion, as shown by the two orange centroids in figure 2.5 c and d). They are both distinct from the other centroids in the same period by showing only a gentle increase in dispersion. A likely reason for the absence of a distinct summer recession in 2014 is the rather wet and cold spring and summer, as can be seen from the steep cumulative rainfall curve during that period (Fig. 2.5 e). In 2013 this identification did not work. Possible reasons are provided in the discussion section 2.5.5.

### 2.4.4 Redundant spatial dispersion functions

We calculated the Shannon entropy for all soil moisture time series for all years and depths (Table 2.2). As explained in section 2.3.5 this reflects the intrinsic uncertainty of the clusters. Most entropy values are within a range of $1 < H < 2.5$. The maximum possible entropy for a uniform distribution of the used binning is 3.55. The Kullback-Leibler divergence $D_{KL}$ is a measure for the information loss due to the compression of the cluster onto the centroid dispersion function. In the overwhelming majority of the cases, the information loss is one magnitude smaller than the intrinsic uncertainty and ranges between $0.01 < D_{KL} < 0.4$. Hence, the

**Table 2.2:** Information content and information loss due to compression. The information content is given as Shannon entropy $H$, which is the expectation value of information in information theory. $2^H$ gives the number of distinct states the underlying distribution can resolve. The information loss after compression is given by the Kullback-Leibler divergence $D_{KL}$ between the compressed and uncompressed series of dispersion functions. The last column relates $D_{KL}$ to $H$

| Year | Depth | # of clusters | $H$ [bit] | $2^H$ | $D_{KL}$ [bit] | $D_{KL} * (H + D_{KL})^{-1}$ |
|---|---|---|---|---|---|---|
| 2013 | 10 cm | 4 | 0.97 | 1.95 | 0.44 | 0.31 |
| 2013 | 30 cm | 3 | 1.49 | 2.81 | 0.06 | 0.04 |
| 2013 | 50 cm | 6 | 2.0 | 3.99 | 0.13 | 0.06 |
| 2014 | 10 cm | 3 | 1.35 | 2.55 | 0.22 | 0.14 |
| 2014 | 30 cm | 4 | 1.57 | 2.97 | 0.3 | 0.16 |
| 2014 | 50 cm | 5 | 2.44 | 5.42 | 0.28 | 0.1 |
| 2015 | 10 cm | 3 | 1.87 | 3.67 | 0.18 | 0.09 |
| 2015 | 30 cm | 3 | 1.18 | 2.26 | 0.09 | 0.07 |
| 2015 | 50 cm | 3 | 2.39 | 5.24 | 0.9 | 0.27 |
| 2016 | 10 cm | 3 | 2.49 | 5.62 | 0.76 | 0.23 |
| 2016 | 30 cm | 4 | 1.44 | 2.71 | 0.02 | 0.02 |
| 2016 | 50 cm | 2 | 3.21 | 9.27 | 2.5 | 0.44 |

information loss due to compression is negligible. There is one exception in 2016 (50 cm).

The clusters obtained in 30 cm for the year 2016 (compare 2.4.1) showed an entropy of 1.44. Compared to this value, the Kullback-Leibler divergence caused by compression of only 0.02 is small, if not negligible. The last column of table 2.2 relates $D_{KL}$ to the overall uncertainty. It contributes less than one third in almost all cases (2016, 50 cm is the only exception). In the majority of the cases it does not contribute more than 20%.

According to equation (2.11) the Shannon entropy is derived from an discrete, empirical probability distribution. As it is calculated using the binary logarithm, $2^H$ gives the amount of discriminable states in this discrete distribution. This number of states is deemed to be a reasonable upper limit for the number of clusters for Mean shift. A higher number of clusters than $2^H$ appears meaningless, and this assures that only those clusters are separated, which are separated by a distance large than the margin of uncertainty.

## 2.5 Discussion

In line with our central hypothesis **H1** - that radiation-driven drying and rainfall-driven wetting leave different fingerprints in the soil moisture pattern which manifests in temporal changes in the dispersion functions - we found strong evidence that soil water dynamics is organized in space and time. Our findings reveal that this organization is not static but exhibits dynamic changes which are closely related to seasonal changes in forcing regimes. A direct consequence is that soil moisture observations are quite predictable in time despite their strong spatial heterogeneity. This is in line with conclusions of e.g. Mittelbach and Seneviratne (2012) or A. J. Teuling et al. (2006), who also found characteristic spatial patterns to persist in time. We used the statistical dispersion of soil moisture observations in dependence of their separating distance to describe spatial patterns. The vector distance of these dispersion functions was used to cluster them. As measure for the degree of organization we used the information loss that goes along with the compression of the entire cluster, ie the replacement of the cluster by the most representative cluster member. Here we found that this compression adds negligible uncertainty compared to the intrinsic uncertainty, caused by propagation of measurements uncertainties. We thus conclude that soil moisture is heterogeneous, but temporally persistent over several months.

In the following we will discuss our main findings that similarity in space leads to dynamic similarity in time, the way we utilized the measurement uncertainty to determine the information content and how two different processes forcing soil moisture dynamics induce two fundamentally different spatial pattern.

### 2.5.1 Spatial similarity persist in time

We related the dispersion of pairwise point observations to their separating distance. For brevity and due to their shape we called these relationships *dispersion functions*. We emphasize that this term is not meant in a strict sense and no mathematical functional relationship, analogous to a theoretical model, has been fitted to the experimental dispersion functions. Despite the fact that the presented functions are empirical, they show clear, recurrent shapes on many occasions.

We found spatial similarity to persist in time. This is reflected in the temporal stability in cluster membership. In line with **H2** - that both forcing regimes and their seasonal variability may be identified through temporal clustering of dispersion functions - the results (Fig. 2.3, 2.5, 2.4) provided evidence that similar dispersion functions emerge in fact very closely in time. Generally they appeared in continuous periods or blocks in time and their changes coincided with changes or a switch in

the forcing regimes. In case we can relate the emergence of such a cluster more quantitatively to the nature and strength of a specific forcing event/process, we can analyze for how long this event/process imprints the spatial pattern of soil moisture observations. Or in other words: we can analyze how long a catchment state *remembers* a disturbance. However, an attempt to relate cluster transitions to rainfall sums and frequencies within the respective moving windows (see Fig. 2.7) did not yield clear dependencies.

Although cluster memberships occur in temporally continuous blocks in all depths throughout all years, for a few cases we could not relate their emergence to distinct changes in forcing. This implies that **H2** needs to be partly revised.

Dispersion functions in 50 cm show a clear spatial dependency throughout the year, with distinct differences within and outside the vegetation period. In 50 cm of 2016 this is different. We find essentially two clusters that do not separate the data series by vegetation period. The shape of the two centroids (Fig. 2.4, f) is similar, only at large distances they differ in value. That means, from orange to blue cluster observations became more similar at large separating distances. Heavy rainfall disturbs this pattern leading to stronger dissimilarity at larger distances and that pattern lasted for a couple of weeks. Then, evapotranspiration driven drying smooths out soil moisture variability and during a similarly strong rainfall event in summer, the cluster can not emerge again as the soil is already too dry. The soil acts as a low pass filter here, which filters out any change in state above a specific frequency. This happens mainly due to dispersion of the infiltrating and percolating water through the soil, or due to storage in the soil matrix. By the time it reaches the deep layers, spatial differences are eliminated. This kind of behaviour is well known and was already reported in the early 90s (Entekhabi et al. 1992; Wu et al. 2002). More recently (Rosenbaum et al. 2012) "found large variations in spatial soil moisture patterns in the topsoil, mostly related to meteorological forcing. In the subsoil, temporal dynamics were diminished due to soil water redistribution processes and root water uptake". In the same year, Takagi and Lin (2012) analyzed a dataset of 106 locations in a forested catchment in the US for spatial organization in soil moisture patterns. They found a seasonal change in more shallow depths (30 cm), controlled by rainfall and evapotranspiration. In deeper depths patterns became more temporally persistent. All these findings are in line with our results and conclusions.

Mittelbach and Seneviratne (2012) decomposed a long term (15 months) soil moisture time series into time-invariant and dynamic contributions to the spatial variance. Their dataset spanned 14 sites from Switzerland at a clearly different scale (150 x 210 km). The study quantified the time-invariant contribution on average to 94%, which leads to "a smaller spatial variability of the temporal dynamics than

possibly inferred from the spatial variability of the mean soil moisture" (Mittelbach and Seneviratne 2012, p.2177 L.14ff). This is comparable to the instances, where we find long lasting clusters while the absolute soil moisture changes considerably (e.g. Fig 2.3 d), early April or mid of July).

### 2.5.2 Uncertainty analysis

We related the evaluation of compression quality directly to the measurement uncertainty. This was achieved by Gaussian error propagation of measurement uncertainty into the dispersion functions and their distances. The latter allowed definition of a minimum separable vector distance between two dispersion functions that are different with respect to the error margin. We based the bin width for calculating the Shannon entropy on this minimum distance, because this assured that the Shannon entropy gives the information content of each cluster *with respect to the uncertainty*. On this basis it was possible to assess compression quality not only by the number of meaningful clusters found, but also based on the information lost due to compression with respect to uncertainty.

In line with **H4** spatial patterns of soil moisture were found to be persistent over weeks, if not months. In many instances we found only two to four clusters within one year and compression was possible with small if not negligible information loss. That means, during one cluster period an entire set of dispersion functions does not contain substantially more information than the centroid function. Hence, the whole cluster can be represented by only the centroid function. We conclude that this is a manifestation of a strongly organized state which persist for a considerable time, as most observations were redundant during these periods.

A. J. Teuling et al. (2006) concluded that picking a random soil moisture observation location and deriving the temporal dynamics from this single sensor is more accurate than using the spatial mean of many soil moisture time series. This conclusion was true for all three datasets they tested (A. J. Teuling et al. 2006). This representativity of a single sensor to our understanding a manifestation of a persistent spatial pattern in soil moisture dynamics, which also enables us to compress clusters without information loss.

From equation (2.11) it can be seen, that the Shannon entropy changes substantially with the binning. Therefore, it is of crucial importance to define a meaningful binning based on objective criteria. We suggest that only a discrimination into bins larger than the error margins makes sense, because smaller differences cannot be resolved based on the precision of the sensors. For the application presented in this work, this is important because otherwise one could not compare the compression quality between depths or years, as different binnings lead to different Shannon

entropy values, even for the same data. Hence, it would be difficult to analyze effects or differences of spatial dispersion in depth or over the years. We thus conclude that the Shannon entropy should only be used if the measurement uncertainties of the data are properly propagated.

We provided an example of how the quality of a compression can be assessed. Instead of considering the number of clusters (compression rate) only, we linked the compression rate to the resulting information loss. We could show that in the majority of the cases substantial compression rates could be achieved, which are accompanied by negligible information losses. We thus suggest that the trade off between compression rate and information loss should be used as compression quality measure.

### 2.5.3 Different dominant processes lead to different patterns

Outside of the vegetation period, we found a recurring picture of spatial dispersion functions with characteristic lengths clearly smaller than the typical extent of hillslopes. Dispersion functions were calculated in three depths for every day throughout four years. In most cases there is an characteristic length at which the dispersion function shows a sudden rise in dispersion. For spatial lags smaller than this distance the dispersion is usually very small. Higher lags show much higher and more variable dispersion values. This characteristic length is approx. 500 m. This corresponds to a common hillslope length for the Colpach catchment. During the vegetation period variability at large separating distance was smoothed out. Dispersion was low also at large distances suggesting similarity even at distances larger than the typical slope length. We thus conclude that there is dependence of the dispersion on the rainfall pattern, which is reflected in the dispersion function's shape and characteristic length. This confirms **H2** and suggest that vegetation is a possible dominant factor in smoothing out soil moisture variability. A similar conclusion is drawn by Meyles et al. (2003), who identified 'preferred states in soil moisture' (Rodger B. Grayson et al. 1997; Andrew W Western and Rodger B Grayson 1998; Andrew W. Western, Rodger B. Grayson, et al. 1999) and could relate the state transition to a significant change in the characteristic length of their geostatistical analysis. We generally found more than two clusters, but we still consider these results to be comparable. Most of the clusters identified during vegetation period are more similar to each other than to the clusters outside of the vegetation period (and vice versa). This can be related to the 'wet' and 'dry' state in Meyles et al. (2003). Although conducted in a very different climate McNamara et al. (2005) also widened the separation of two preferred states into five which they found to be explanatory for runoff generation. Interestingly they found the seasonal interplay

of precipitation and evapotranspiration responsible for transitions between states. Vanderlinden et al. (2012) further references Gómez-Plaza et al. (2001) as an example study, which identified vegetation as the dominant factor. Plant root activity is changing the temporal stability of soil moisture in the upper 20 cm of the soil considerably.

Outside the vegetation period we observed multiple cluster transitions. Although more than one cluster was identified, the clusters were more similar in shape to each other, than to the clusters in the 'dry' summer period. In many cases these cluster transitions coincided with a shift in rainfall regimes. Either the first stronger rainfall event after a longer period without rainfall sets in, or one of the heaviest rainfall events of that year occurs. There are also instances with recurring clusters that develop more than once (eg. Fig. 2.3, 2.4 a, 2.4 c, 2.5 e). As these periods are controlled by rainfall either different rainfall patterns or different hydrological processes are dominating. Depending on antecedent wetness, rainfall amounts and rainfall intensity, infiltration and subsurface flow processes can change and thus also alter the soil moisture pattern. Although this may only be a coincidence, we found the green cluster in 2016 (Fig. 2.3) to form with strong rainfall input setting in after a period of little rainfall. Similar observations can be made for other years, unfortunately not in all cases. Consequently, we can neither confirm nor reject **H3** - that spatial dispersion is more pronounced during and shortly after rainfall driven wetting conditions.

Many other works also tried to link soil moisture pattern to forcing. Adriaan J Teuling and Troch (2005) report for soil moisture measurements taken on an agricultural field in Belgium, that the first rainfall events in the late growing season even out the variability, which arose due to heterogeneous transpiration. Although soil moisture pattern became more homogeneous in summer in our case, we also suspect rainfall events after the vegetation period to be responsible for cluster transitions. Similarly, Albertson and Montaldo (2003) present a set of examples of modelled experiments, in which precipitation is consistently 'producing' variability in soil moisture dynamics and transpiration is reducing variability. The question of how spatial patterns or their variability change is also contradictory in the literature. Vanderlinden et al. (2012) present two studies in their review. Both investigated the variability of time persistent soil moisture patterns over depth. While Pachepsky et al. (2005) found no difference in depth, W. Choi and Jacobs (2011) reported a decrease of variability with depth. During the vegetation period no spatial dependence is detectable. For the vegetation period, we found usually only one or at maximum two clusters (Table 2.2). These clusters are characterized by showing no dependence of dispersion on separating distance. That means, evapotranspiration forcing the system to drier states is doing this in a (spatially)

homogeneous manner. Dispersion is not only low when the catchment is dry, it is also low while the system is drying. Similar observations have been reported for the Tarrawarra catchment in Australia (Rodger B. Grayson et al. 1997; Andrew W Western and Rodger B Grayson 1998; Andrew W. Western, Rodger B. Grayson, et al. 1999). Although these works focused on the relation of spatial organization to topographic indices, no spatial correlation of soil moisture observations could be found for the dry period. This is comparable to our to our findings about dispersion functions during the vegetation period. It has to be noted that the lowest soil moisture values, i.e. residual moisture, are only observed for very short periods in time. At residual soil moisture all sensors show essentially the same absolute value (which leads to small dispersion as well).

We conclude that cluster transitions were often triggered by rainfall events. Not each of the strongest rainfall events caused a cluster transition and not each cluster transition could be related to a rainfall sums or frequency within the window of the transition. The characteristics provided in appendix 2.7.2 provide a good starting point, but further investigations on the rainfall events, their spatial characteristics and relation to the moisture state are needed.

### 2.5.4 Mean shift as a diagnostic tool

We used Mean shift mainly as a diagnostic tool to cluster dispersion functions based on their similarity. Similarity is measured by the Euclidean distance between two dispersion function vectors. This Euclidean distance does, however, not provide information on the underlying cause of dissimilarity and thus a minor difference in the values of the dispersion functions, even though characterized by a very similar shape, could result in the same level of dissimilarity as a change in the shape of the dispersion function. We observed some cluster separations that were caused by minor differences in mean dispersion, while essentially describing the same spatial dependency.

It is possible to train better Mean shift algorithm instances. As described in the methods, we selected the bandwidth parameter for Mean shift to yield meaningful results for the entire data set. The same parameter was used for all subsets to cluster dispersion functions on the same basis. This makes the clustering procedure itself comparable and thus, the number of identified clusters can support result interpretation. Nevertheless, it is likely that better bandwidth parameters can be found for each data subset individually and overcome misclassifications as described above. Our objective, however, was to find clustering results that can directly be compared to each other (instead of comparing hyper-parameters).

Dispersion functions operate in a higher dimensional space and might be affected

by the curse of dimensionality. Mean shift clusters data points based on their distance to each other. Following the theory of the curse of dimensionality, with each added dimension (of these points), the difference of maximum and minimum distance between points become less significant (Beyer et al. 1999). On the one hand, we wish to resolve dispersion functions on as many distance lag classes as possible to gain more insight on spatial dependencies. On the other hand, each additional lag class possibly decreases the performance of Mean shift (or any other clustering algorithm) and turns the results less meaningful. We calculated dispersion functions using a 30 days aggregation window and therefore end up with 335 points for Mean shift. However, despite the limited number of points and the resulting uncertainty of cluster identification the clusters identified here seem plausible.

Mean shift is sensitive for the bandwidth parameter. As described in the methods (section 2.3.3), the bandwidth parameter has to be specified and has direct influence on the amount of clusters formed by the algorithm. We found a suitable parameter through trial-and-error. It would be more satisfactory to infer this crucial parameter from the data or supplementary information gathered at field campaigns. However, to our knowledge there is no such method or procedure to infer bandwidth parameters for Mean shift from a data sample.

### 2.5.5 Limitations of the proposed method

Successful clustering does not point out spatial dependency. Mean shift can cluster functions without spatial dependency, as it uses their distance and no actual covariance between the functions. In this case the clustering is based on differences in mean, which may not even be statistically significant. The Mean shift algorithm is not meant to test clusters for statistical independence. If two groups of points are separated or not depends only on the bandwidth parameter. Therefore the centroid functions of each cluster have to be checked for their shape and the information on spatial dependency that follows from that shape.

Our approach to find suitable bins to calculate the Shannon entropy is sensitive to outliers. We decided to rather define the width of a bin instead of their number. The reasons and necessity to do so were discussed in detail in section 2.5.2. As a width we used the uncertainty propagated into dispersion function distances. From all distances within uncertainty margins, we used the maximum value. In cases where this maximum distance is an outlier, it will influence the whole entropy calculations. This is a limitation to our method, but an acceptable one as it is still superior to other approaches from our point of view. Choosing the maximum distance within each year or depth (or both) will yield more bins for entropy calculations and therefore a wider range of values, but it would be very hard to compare these values.

From the point of view of the monitoring network, it has to be mentioned that the analysis of the 2013 data is likely to be less reliable, as during this period of installation the number of sensors was still lower than in the following years.

Due to the sampling design and the amount of observation points, we did not systematically test for differences of forest vs pasture plots, but ran our analyses across the two land covers. The fundamentally different shapes of cluster centroids in the summer clusters and, thus, the strong effect of vegetation altering soil moisture patterns might be partly more pronounced due to the sampling design and not easy transferable to other sites. In our opinion, we would have made the same observations with a more stratified sampling design, as this is systematic catchment behavior, but we can neither confirm nor reject this.

## 2.6 Conclusions

We presented a new method to identify periods of similar spatial dispersion present in a data set. While soil moisture observations might be spatially heterogeneous, spatial patterns are much more persistent in time. We found two fundamentally different states: On the one hand rainfall-driven cluster formations, usually characterized by strong relationships between dispersion and separating distance and a characteristic length roughly matching the hillslope scale. On the other hand we found clusters forming during the vegetation period. A drying and then dry soil exhibits dispersion functions which are much flatter, indicating homogeneity across space. Interestingly, these functions flatten out by minimizing the dispersion on large distance lags, which implies that dissimilarities do not increase with separating distance. We can thus see how the soil acts as a low pass filter.

While these long lasting periods of similar spatial patterns help us to understand how and when the soil is wetting or dying in an organized manner, there are possible applications beyond this. One could use the identification of clusters to stratify data based on spatial dispersion for combined modeling. Then, for example, a set of spatio-temporal geostatistical models or hydrological models applied on each period separately, might in combination return reasonable catchment responses.

Our most interesting finding is that even a few soil moisture time series bear a considerable amount of predictive information about dynamic changes of soil moisture. We argue that distributed soil moisture reflects an organized catchment state, where soil moisture variability is not random and only a small amount of observation points is necessary to capture soil moisture dynamics.

# 2.7 Appendix

## 2.7.1 Mean shift algorithm

Mean shift starts by forming a cluster for each sample on its own. Here, a *sample* corresponds to one dispersion function. We will illustrate the fundamental mechanism of the algorithm for the two-dimensional case, as the samples can easily be plotted in $\mathbb{R}^2$ (see figure 2.6 a, b). Mean shift works iteratively. In each iteration, a window is shifted over all samples, which can be thought of as coordinate points in the two-dimensional case (see fig. 2.6 a). This window is called a *kernel* that is controlled by a size parameter called *bandwidth*, which is the Euclidean distance between two samples. In the two-dimensional case, this can be thought of a circle with a radius set to the given bandwidth as shown in figure 2.6 a. In each kernel position, the center of sample density is calculated and the current sample is *shifted* onto this point, which is the new cluster mean, called cluster centroid. On the next iteration, the newly created cluster centroids are used as the new (input) samples, as shown in figure 2.6 b. Hence, with the bandwidth, we define a maximum Euclidean distance at which two samples are still considered to belong to the same group. The iterations stop when the shifting means converge (centroids do not change their position anymore). We substitute the centroids calculated on the last iteration by the original sample closest to this point. Thus, we choose the most representative dispersion function for the cluster.

Mean shift is sensitive to the selected bandwidth. Two clusters whose centroids are within one bandwidth length will be shifted into a combined cluster before convergence is met. As a result a bandwidth parameter chosen too big might classify all samples as a single cluster as indicated in figure 2.6 c. In case the bandwidth is chosen too small many tiny clusters with just a few members will be the result. Figure 2.6 d shows an extreme example, where no sample will shift anywhere. We tested different bandwidth parameters at a few examples and set the bandwidth to the 30% percentile of all pairwise Euclidean vector distances between the dispersion functions of one year and depth. We chose the so-called flat kernel as a kernel, which would result in a circle in the two-dimensional case and a N-dimensional sphere in the $\mathbb{R}^N$, where $N$ is the number of lag classes used for the dispersion function.

## 2.7.2 Auxiliary quantitative results

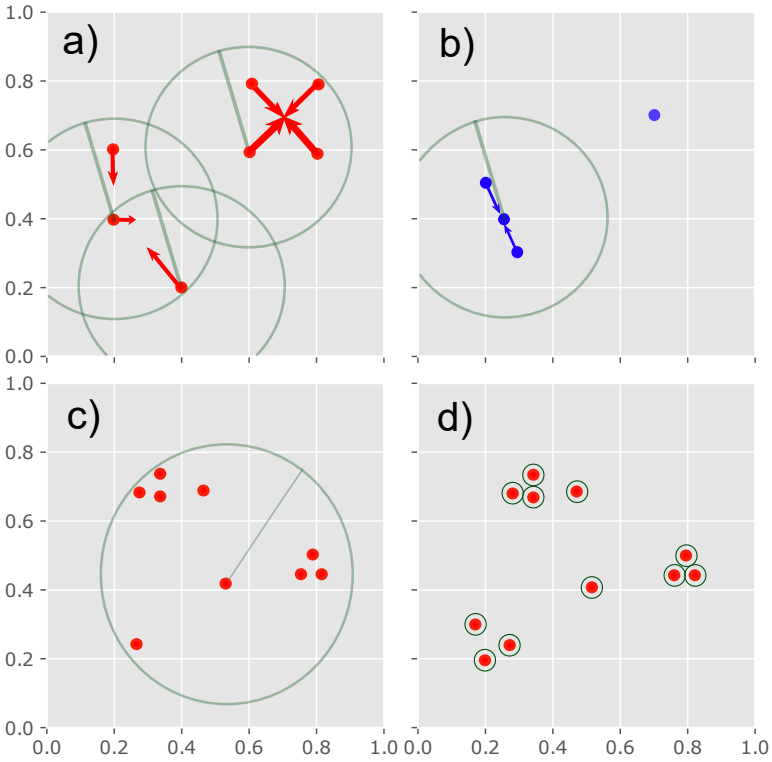



**Figure 2.6:** Schematic procedure of the Mean shift algorithm in $\mathbb{R}^2$. **a):** Red dots indicate hypothetical samples to be clustered. The circles are illustrating the flat kernel of the centered sample at first iteration. The bandwidth parameter is illustrated by the radius. The red arrows indicate the shift of the respective sample onto the geometric mean of all samples inside the current kernel. Note that three points on the left side are shifted differently, as the upper and lower point do not lie in each others kernel. **b):** Second iteration step after a). The blue dots are the shifted means from a) and will be used as the input sample for the next iteration. The procedure finishes when no points can be 'shifted' anymore. **c):** Example of a large bandwidth (radius), which will result in only one cluster at convergence. **d):** Example of a too small bandwidth, where no point will be shifted at all.

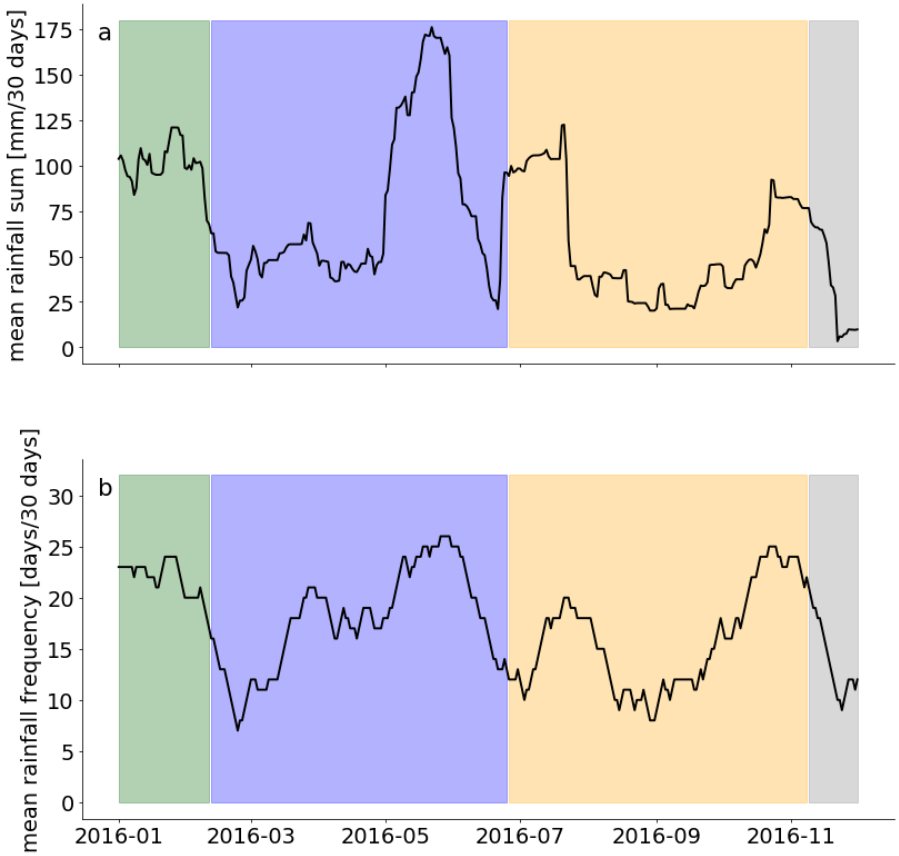



**Figure 2.7:** Mean rolling rainfall sum (a) and rainfall frequency (b) in for 2016. The colored boxes indicate the current cluster as shown in figure 2.3 d. Both values were calculated for the same windows as the dispersion functions by using equation (2.1) for the daily rainfall sums, with the total rainfall sum in the window in (a) and the number of days rainfall occured in (b).

**Table 2.3:** Quantitative results summary. For each depth and cluster of 2016 different cluster characteristics were calculated. The duration of each cluster is given in the third column. To compare rainfall forcing with the emergence of clusters, the rainfall characteristics were based on the same moving window as the clusters. The mean rainfall frequency $f_{30}$ within each window is given in the fourth column. The mean 30 day-sum over the whole cluster $\sum_{i=0}^{30} R$ in the fifths column. To assess the variability of dispersion functions within each cluster, different measures are given. $\gamma$ is the dispersion, as calculated in equation (2.2). This describes the dispersion of dispersion functions within one cluster. $H$ is the entropy of the distribution of all cluster members within each cluster. Both measures are calculated for the distribution of each distance lag class individually.

| depth | cluster | duration [days] | $f_{30}$ | $\sum_{i=0}^{30} R$ | $\gamma$ | $H$ |
|---|---|---|---|---|---|---|
| 10cm | blue | 167 | 16.22 | 59.39 | 5.5e-6 | 1.34 |
| 10cm | orange | 88 | 17.48 | 73.73 | 1.12e-5 | 0.69 |
| 10cm | green | 113 | 20.54 | 81.31 | 4.36e-5 | 1.25 |
| 30cm | blue | 135 | 17.84 | 73.32 | 2.16e-5 | 1.83 |
| 30cm | orange | 136 | 15.78 | 54.87 | 4.21e-6 | 1.63 |
| 30cm | green | 42 | 21.86 | 100.62 | 4.4e-5 | 1.31 |
| 50cm | blue | 282 | 16.41 | 60.66 | 3.51e-5 | 0.99 |
| 50cm | orange | 54 | 21.59 | 98.02 | 3.64e-5 | 0.9 |

# 2.8 Detailed result plots of 30 cm in 2014 and 2015

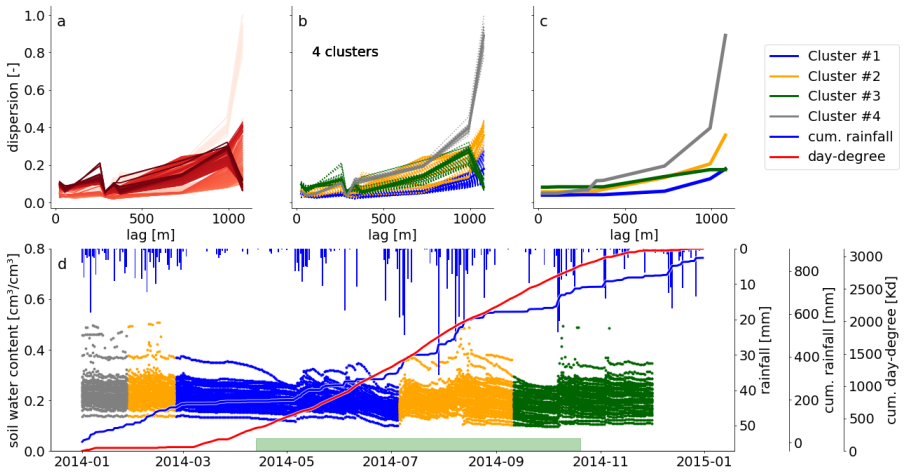



**Figure 2.8:** Spatial dispersion functions in 30 cm for 2014 based on on a window size of 30 days.
**a):** Spatial dispersion function for each position of the moving window. The red color saturation is indicating the window position. The darker the red the higher in the year.
**b):** The same dispersion functions as presented in a). Here the color indicates cluster membership as identified by the Mean shift algorithm.
**c):** Compressed spatial dispersion information represented by corrected cluster centroids. The colors match the clusters as presented in b).
**d):** Soil moisture time series of 2014 in 30 cm depth. The colors identify the cluster membership of the spatial dispersion function of the current window location and is matching the colors in b) and c). The bars on the top show the daily precipitation sums. The solid blue line is the cumulative daily precipitation sum and the red line the cumulative sum of all mean daily temperatures > 5 °C. The green bar marks the assumed vegetation period. It covers the dates where the cumulative day-degree sum is > 15% & < 90% of the maximum.

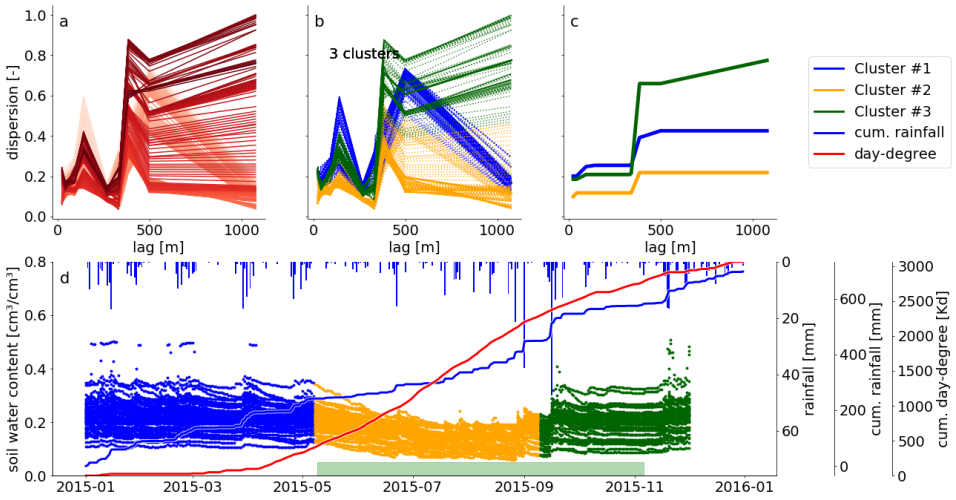



**Figure 2.9:** Spatial dispersion functions in 30 cm for 2015 based on on a window size of 30 days.
**a):** Spatial dispersion function for each position of the moving window. The red color saturation is indicating the window position. The darker the red the later in the year.
**b):** The same dispersion functions as presented in a). Here the color indicates cluster membership as identified by the Mean shift algorithm.
**c):** Compressed spatial dispersion information represented by corrected cluster centroids. The colors match the clusters as presented in b).
**d):** Soil moisture time series of 2015 in 30 cm depth. The colors identify the cluster membership of the spatial dispersion function of the current window location and is matching the colors in b) and c). The bars on the top show the daily precipitation sums. The solid blue line is the cumulative daily precipitation sum and the red line the cumulative sum of all mean daily temperatures > 5 °C. The green bar marks the assumed vegetation period. It covers the dates where the cumulative day-degree sum is > 15% & < 90% of the maximum.

# 3 Uncertain observations in geostatistics

# SciKit-GStat Uncertainty: A software extension to cope with uncertain geostatistical estimates

The following chapter 3 is already published as a research article in *Spatial Statistics* as:

*Mälicke, M., Guadagnini, A., & Zehe, E.: SciKit-GStat Uncertainty: A software extension to cope with uncertain geostatistical estimates. Spatial Statistics, https://doi.org/10.1016/j.spasta.2023.100737, 2023.*

Author contributions:
Mirko Mälicke: Conceptualization, Use-Case, Data, Methodology, Software. Alberto Guadagnini: Use-Case, Data, Methodology. Erwin Zehe: Use-Case, Methodology.

Data and Code availability:
The pancake dataset is available with the SciKit-GStat package (Mirko Mälicke, Möller, et al. 2021a). The source code, including the pancake data sample, is available on Github[3]. The Berea sandstone data sample can be obtained from the original publication (Vincent C. Tidwell and John L. Wilson 1997).

The source code for the *SciKit-GStat Uncertainty* extension is available on Github[4]. This repository includes a copy of the used data samples. The primary distribution of the software package is a docker image[5].

The demo application is not open source. It can be reached at https://geostat.hydrocode.de

**3** https://github.com/mmaelicke/scikit-gstat; last accessed: 25.10.2022
**4** https://github.com/hydrocode-de/skgstat_uncertainty; last accessed 25.10.2022
**5** : https://github.com/hydrocode-de/skgstat_uncertainty/pkgs/container/skgstat_uncertainty; last accessed: 25.10.2022

## 3.1 Abstract

This study is focused on an extension of a well established geostatistical software to enable one to effectively and interactively cope with uncertainty in geostatistical applications. The extension includes a rich component library, pre-built interfaces and an online application. We discuss the concept of replacing the empirical variogram with its uncertainty bound. This enables one to acknowledge uncertainties characterizing the underlying geostatistical datasets and typical methodological approaches. This allows for a probabilistic description of the variogram and its parameters at the same time. Our approach enables 1) multiple interpretations of a sample and 2) a multi-model context for geostatistical applications. We focus the sample application on propagating observation uncertainties into manual variogram parametrization and analyze its effects. Using two different datasets, we show how insights on uncertainty can be used to reject variogram models, thus constraining the space of formally equally probable models to tackle the issue of parameter equifinality.

## 3.2 Introduction

Geostatistical analyses are key in several research and industrial areas, including environmental and Earth sciences and engineering application. In this broad context, geostatistics typically considers (statistical) dependences of spatial or spatio-temporal datasets. In viewing a given quantity as a correlated random field, it has been shown to provide critical insights on ways to interpolate, assess, re-scale, and model scenarios of interest in the presence of scarce information. A broad variety of studies is geared towards assessing uncertainty through geostatistical estimation or simulation frameworks (Delbari et al. 2009; Emery and Peláez 2011; Handcock and Stein 1993; André G Journel 1994; Lloyd and Peter M Atkinson 2001; Mowrer 1997; Nowak and Verly 2005; Todini 2001; Erwin Zehe, Becker, et al. 2005), including some recent hydrological applications focused on preferential pathway analysis (Schiavo et al. 2022; E. Zehe, R. Loritz, et al. 2021). Otherwise, only a limited number of studies focus on a rigorous framework of analysis to explicitly include uncertainties associated with the empirical variogram and the way these can impact the estimation of an appropriate interpretive model. In this context, our study aims at providing enhanced insights on this, as the reliability of a geostatistical analysis hinges on an appropriate estimation of the empirical variogram. Thus, our distinctive objective relates to the way one can incorporate uncertainties into the variogram estimation. We then assess the way uncertainty associated

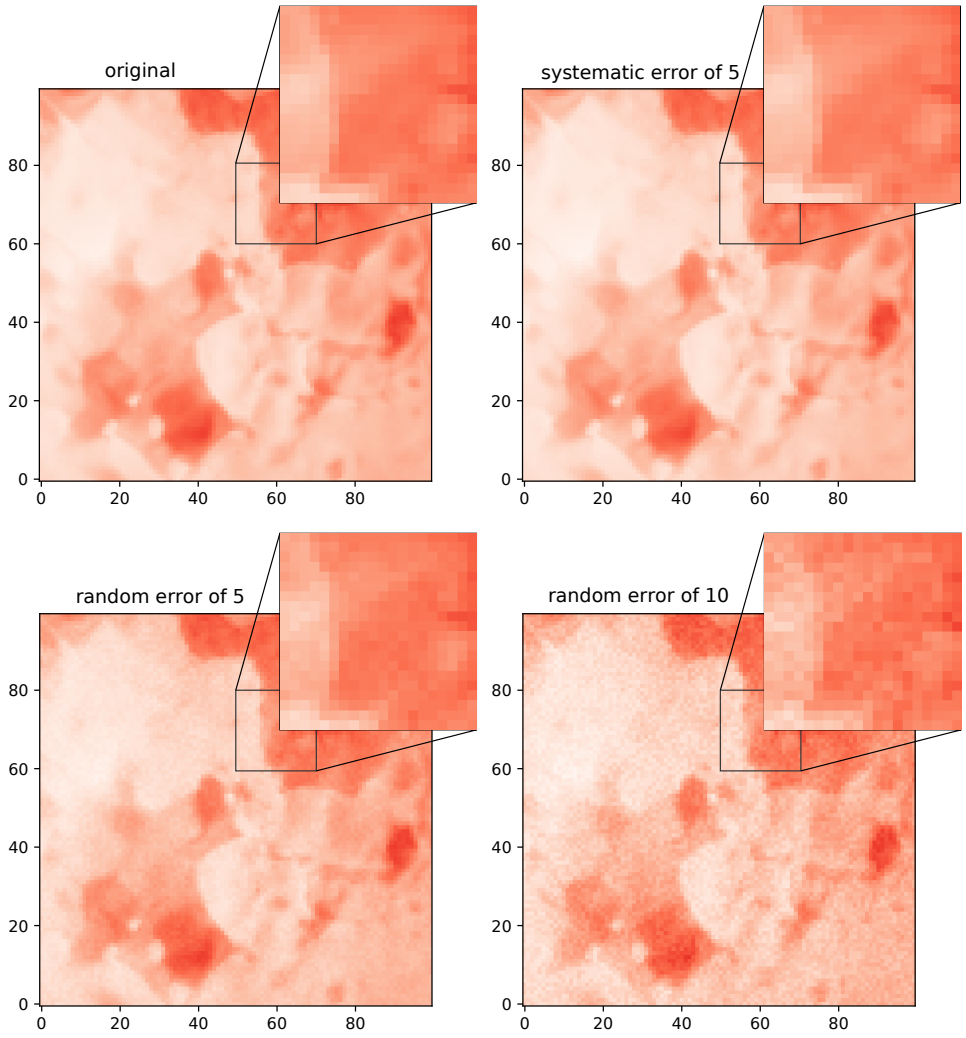



**Figure 3.1:** Image of the pancake, which motivated this work. It shows the image of a pancake with several conceptualized errors applied. **a)** Red channel of the original image with a 20x20 zoom of an area with apparent gradient on short distance. **b)** Image from a) with a systematic shift of 5 in the red channel value. **c)** Image from a) with a random error of 5 applied to each cell in the red channel. **d)** Image from a) with a random error of 15 applied to each cell in the red channel.

with the assessment of the empirical variogram can propagate onto subsequent analysis steps. This allows seamless inclusion of uncertainty into geostatistical interpolations.

To the best of our knowledge, only a limited series of studies address uncertainty in the empirical variogram. Webster and Oliver (1993) define confidence limits for individual spatial models and their parametrizations. Their study considers sub-sampling of a dense datasets and focuses solely on the impact of sample size and the way a threshold can be defined for it through numerical Monte Carlo simulations. Pardo-Igúzquiza and Peter Dowd (2001) describe various approaches to yield approximations of the standard error associated with the variance evaluated across a sample. These authors point out that exact confidence intervals for the empirical variogram are difficult to construct in practice and only a number of approximations can be employed. Some of the methods discussed therein are detailed in Section 3.4. Their studies relate the uncertainty of empirical variograms to the nature of the semi-variance estimator. Metrics of statistical robustness are then proposed on the basis of the size of the underlying finite sample.

While building on these approaches, here we address the joint effect of several sources of uncertainty on the empirical variogram. We highlight ways these can be tackled and ultimately be included into a variogram modeling context. Some of these sources of uncertainty are aleatory. These include e.g., the inherently limited precision of data in terms of accuracy of an observed quantity as well as of the spatial locations at which observations are taken. Other sources of uncertainty are epistemic and stem from incomplete knowledge about a system functioning and/or processes taking place therein (Der Kiureghian and Ditlevsen 2009; Hora 1996; Hüllermeier and Waegeman 2021). Observations are never perfect in terms of precision and accuracy associated with a given measurement. Furthermore, in some cases one cannot observe directly a target quantity, while only data (corrupted by uncertainty) associated with other related quantities can be monitored. As a common example, one can refer to a rainfall radar, which is not rendering rainfall observation, but reflectivity of hydro-meteors. The latter depends on size and shape of the meteors, their chemical phase and a variety of additional factors (Neuper and U. Ehret 2019). All of these sources of uncertainty jointly contribute to what we term *observation uncertainty* in this study.

An exemplary scenario underpinning of our study is associated with the geostatistical Python package SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a) and corresponds to an image of a pancake taken at a given time during browning. Figure 3.1 a) illustrates the actual image and an inset of a target area. Color gradation corresponds to the red channel pixel value, which has a resolution of 8-bit, as common

for images. We rely on this pixel value as observation here. This is also consistent with common remote sensing observation techniques, the pancake surface and the image corresponding to the random field under study and to the measurement, respectively. Note that this representation (along with the 8-bit resolution) already implies observation uncertainty. Any given RGB value in the photograph does not reflect the real color of the actual pancake. There are systematic and random errors influencing the measurement. These include e.g., the moisture of the air between the camera and the pancake or oscillations of the light bulb brightness slightly affecting room illumination. To assist evaluation of observation uncertainties in the context of pancakes and as an example to provide a visual depiction of the effect of uncertain observation, we apply a systematic shift in value (fig.3.1 b) and a random variation in value (fig. 3.1 c, d) of a different magnitude in each sub-panel to the original image. Differences in color from figure 3.1a (original) to figure 3.1b and c are visually very hard to detect. This illustrates that even a considerable variation in value might manifest in a subtle way from a visual standpoint. Figure 3.1c depicts the magnitude of measurement error, which forms the basis for some of the analyses detailed in Section 3.4 and 3.5.

These kinds of observation uncertainties are somehow less subtle in remote sensing, groundwater hydrology or soil science. Sensor sensitivity studies have shown that observation values are typically subject to much larger ranges of uncertainty (ie. fig. 4 Arthur and Robinson 2015; C. Jackisch et al. 2020; Erwin Zehe and Blöschl 2004). In addition to the above mentioned elements, one should note that some research studies can also be affected by un-calibrated sensors and/or, in some instances, on community-sourced sensors (Chapman et al. 2017), which do not comply with the same measurement standards and might also provide only indirect information about the quantity of interest.

Prompted by these elements, we illustrate here the software library SKGstat-Uncertainty that has been developed to specifically address these outstanding issues. The latter is built on existing and established packages for geostatistics in Python. It implements existing and original methods to analyze, assess, quantify, visualize, and propagate uncertainties in variogram estimation. Existing software solutions in Python include SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a). The latter is a variogram estimation toolbox that is currently characterized by only limited capabilities to handle observation uncertainties. For example, in the current implementation one could add error bars to semi-variance values on the basis of a manual input. Additionally, GSTools (Müller et al. 2021a), an advanced geostatistical toolbox in Python, implements uncertainty elements for Kriging only if the user can supply the measurement error as a parameter. In this context, SKGstat-Uncertainty

can be identified primarily as an extension to SciKit-GStat and is also compatible with GSTools.

SKGstat-Uncertainty is designed as a general toolbox, that is aimed at performing uncertainty analyses associated with variogram estimation in a way that is accessible to a broad audience. As such, end-users are envisioned to be associated with education, research, and industry sectors. In addition to providing a thorough introduction to the various functions of the toolbox, we exemplify the importance of variogram uncertainty upon considering two exemplary datasets.

Note that our study does not involve automatic fitting of a variogram model, even as the toolbox includes these features (namely the method-of-moments and the Maximum Likelihood approach). For the purpose of our exemplary study, we favor manual fitting of variogram functions to the uncertainty bounds. Doing so enables users to readily inspect various dimensions of uncertainty arising in the context of variogram analysis. By replacing the empirical semi-variance with its confidence limits (see Section 3.4.3), we explore the uncertainty in the parametrization of a given variogram model. Importantly, we also show that the choice of the theoretical model itself becomes uncertain. In this sense, the heart of the toolbox is a processing module that implements a suite of methods for the quantification of uncertainty associated with empirical semi-variance. Each of these is conducive to an uncertainty bound against which a collection of variogram models and ensuing parametrizations can be assessed. A rich selection of visualization routines enables the user to inspect various aspects of uncertainty. This offers a considerable added value with respect to parameterizing a black-box workflow to obtain a result, which might possibly be considered as the *right* or *most probable* one.

We perform the uncertainty analysis for **a)** the pancake data set depicted in figure 3.1 and **b)** a hydrogeological dataset. The latter comprises a set of well-established and broadly used air permeability data collected across a Berea sandtone rock on a regular, dense grid (Vincent C Tidwell and John L Wilson 1999; Vincent C. Tidwell and John L. Wilson 1997; Vincent C. Tidwell and John L. Wilson 2002) and is detailed in Section 3.4.1.

After an introducing the software package and the sample application for manual variogram parametrization, we explore the following two research hypotheses:

- hypothesis H1: Empirical variograms (or semi-variances) are uncertain due to inherent observation and estimation uncertainty.
- hypothesis H2: Uncertain empirical semi-variances imply that an interpreting variogram model and the embedded parameters are uncertain; this, in turn, yields uncertain geostatistical interpolation results.

Testing both hypotheses relies on the presented toolbox.

Our study is structured as follows: Section 3.3 describes the toolbox from a technical perspective; Section 3.4 includes all methodologies used for the presented analysis; Section 3.5 illustrates the results and our findings, which are then discussed in Section 3.6. Conclusions are presented in 3.7.

## 3.3 Software implementation

Our software is a toolbox that is designed to extend the functionality of two well-established geostatistical Python packages, i.e., SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a) and GSTools (Müller et al. 2021a). Key extensions include the implementation of geostatistical analysis tools and functions with options for uncertainty analysis and propagation. The toolbox implements building blocks to form applications governed through a dedicated graphical user interface. While the main focus is set on variography and Kriging, the toolbox is general and can be readily extended to include additional features.

The toolbox SKGstat-Uncertainty is written in Python and is published as open source (Mirko Mälicke 2022a). It is a collection of functions, which can be run through the Python framework `streamlit`. This opens a web-browser based interface to operate the underlying Python code and its settings. As such, advanced programming skills are not required to load data, set up geostatistical libraries, pre- and post-process data, set model parameters, run analyses, and visualize the ensuing results.

Applications built with our toolbox can be scaled. With minimal overhead, it can be run locally on any client computer, a feature that enables one to readily interact with locally hosted data. Alternatively, public streamlit applications can be hosted on a cloud infrastructure of streamlit with limited resources, freely available. It is further noted that deploying a streamlit application on custom infrastructure is straightforward and in line with common web-based deployment strategies. This enables one to use the software at any scale in educational and professional scenarios, in a freely accessible mode, or as the foundation of a paid model. Finally, the toolbox is distributed as a Docker image with fixed software versions and architecture. Docker is a common solution to ensure reproducible software deployment independent of the host architecture and operating system. This enables one to repeat analyses ensuring consistent results.

The software toolbox is structured across several units. First, the *Data Models* describes the structure of the data used by the application. Exemplary, one model

describes the attributes and structure of uploaded samples, while another one describes the attributes, which represent an empirical variogram. Data models also include relations between data model instances (usually called entities). Each model is implemented as a Python class and can easily be exported to the open standard format JSON[6]. Thus, students, scientists, or engineers and practitioners can easily export data and results from the application and use these for further analyses in any other framework of their choice. SKGstat-Uncertainty uses an SQLite database to save application data and intermediate results, as a default option. Connecting the toolbox to other database systems is also possible, as it uses the widely spread Python module `SQLAlchemy` (Bayer 2012), which can connect to (almost) any relational database management system. The demo application stores the data in a remote PostgreSQL database.
Another unit termed *processor* implements algorithms for model evaluation, sampling, uncertainty propagation, and analysis. These algorithms are detailed in Section 3.4. An *Application Programming Interface (API)* unit collects functions for all common data management tasks, including filtering, creating, editing, or deleting information. While the API is used by the application, it is also usable as a standalone Python module and can be run as a command line interface directly from the operating system. The core unit is termed *components*. It includes the main functions, which are used by the streamlit framework to build the application. These functions run and operate the analysis as specified by the developer.
The *chapters* unit is a collection of standalone streamlit applications. These can be composed together into a final application, or can be run individually. Each of the chapters covers a given topic. Most chapters build on others, e.g., the chapter about Kriging algorithms can only be used after variograms are estimated for a target dataset. The software toolbox currently implements the following chapters:

- **Data management** - This chapter can upload, list and edit existing datasets. New samples can be created by re-sampling exiting datasets.

- **Learn geostatistics** - This chapter provides an interactive and guided step-by-step introduction to geostatistics, which might be appropriate for an undergraduate or early stage graduate student. The details are not covered in this work, given their introductory nature and target audience.

- **Variogram estimation** - The chapter implements an interactive interface to estimate sample variograms and propagate various kinds of uncertainty into

6 Human readable JSON format specification. URL: https://www.json.org/json-en.html, last accessed: 25.10.2022.

the empirical variogram. This yields a uncertainty bound-based empirical variogram.

- **Model parametrization** - The chapter implements an interactive interface to identify an arbitrary amount of models and associated model parametrization within the uncertainty bounds of each variogram.
- **Kriging** - The chapter implements four different Kriging algorithms (simple, ordinary, universal, and external drift Kriging) leveraging on the identified variogram model functions to project data onto unobserved locations.
- **Geostatistical simulations** - The chapter implements an interface to perform geostatistical simulations for each theoretical variogram model function. For simplicity, the simulation feature of the tool is not included in this study.
- **Analysis tool** - The chapter enables one to visualize estimation (i.e., Kriging) or simulation results with a variety of pre-defined visualization options (see, e.g., Section 3.5).

A scientific demo application (termed *uncertain geostatistics*) is implemented to assist the user and can be reached publicly at https://geostat.hydrocode.de[7]. It does not add any significant functionality in terms of geostatistics or uncertainty analysis. The demo application runs an additional PostgreSQL database instead of the default sqlite database. Besides the chapters of SKGstat-Uncertainty described above, three more chapters were added to the application. The *help page* chapter loads documentation from the underlying Python packages SciKit-GStat and GSTools for reference. A *tutorials page* lists a number of short video tutorials about the other chapters. Additionally, a landing page including a login was added. Authenticated users are granted full access to additional data samples, which are not available under an open data license. Without authentication, data are still available when using the application. Otherwise, re-sampling and downloading non-open data (e.g., the Berea sandstone dataset illustrated in Section 3.4.1) are disabled. Authenticated access to the scientific sample application is managed by a third party[8], access to the Berea sandstone dataset can be obtained from the original publication (Vincent C. Tidwell and John L. Wilson 1997).

7 The whole geostatisitcal ecosystem around SciKit-GStat, SKGstat-Uncertainty and demo applications can be reached at https://geostat.hydrocode.de. The standalone demo application is deployed at https://uncertain.geostat.hydrocode.de

8 As of this writing, the demo application and the Python package are properties of hydrocode GmbH (https://hydrocode.de). The Python package is open source, while the demo application is free of charge.

# 3.4 Data and Methods

## 3.4.1 Data

### Pancake dataset

A detailed description of the pancake dataset is offered by (Mirko Mälicke, Möller, et al. 2021a). In line with this study, we consider the red channel of the RGB image in our analyses. For the purpose of our analysis, we re-scale the original red channel image described by (Mirko Mälicke, Möller, et al. 2021a) (and associated with a $500 \times 500$ resolution) to a $100 \times 100$ resolution using a mean filter. Note that this step corresponds to smoothing the original image, hence decreasing the sample spatial variance. Otherwise, **a)** it does not affect the workflow underpinning the application of our approach to tackling sample variogram uncertainty and **b)** it enables us to obtain a sample that is approximately the same size as the one associated with the air permeability information evaluated across the block of Berea sandstone described in Section 3.4.1. We then apply our workflow considering a reduced size data sample constructed upon re-sampling the $100 \times 100$ resolution image according to a uniform 10 $times$10 grid without any offset from the border, to avoid extrapolations in Kriging analyses.

### Berea sandstone

The second dataset we consider is well established and representative of a Darcy-scale collection of air-permeability data (Vincent C Tidwell and John L Wilson 1999; Vincent C. Tidwell and John L. Wilson 1997; Vincent C. Tidwell and John L. Wilson 2002). The latter are sampled on the six faces of a $81 \times 74 \times 63 cm^3$ block of Berea sandstone, across an area of $30 \times 30 cm^2$. The sampling grid comprises $36 \times 36$ regularly spaced nodes (horizontal resolution $\Delta = 0.85 cm$). Data collection relies on four air minipermeameters, each with a given tip-seal (inner and out radius of the minipermeameter are $r\ i = \{0.15, 0.31, 0.63, 1.27\}$ and $r_2\ i = \{1, 2, 3, 4\}$, respectively). For the purpose of our analyses, we focus on the set of data associated with the smallest tip-seal radius.
Recent geostatistical analyses of these data include the works of (Dell'Oca et al. 2020; Riva, Neuman, et al. 2013).

Given the size of the minipermeameter tip, the original Berea sandstone dataset can be considered exhaustive and is used as the (hydrogeological) field equivalent of the pancake image. A sub-sample of the air permeability data to be used in our uncertainty analyses is then obtained upon considering the information available on a uniform $8 \times 8$ grid, approximately corresponding to 10% of the field. This

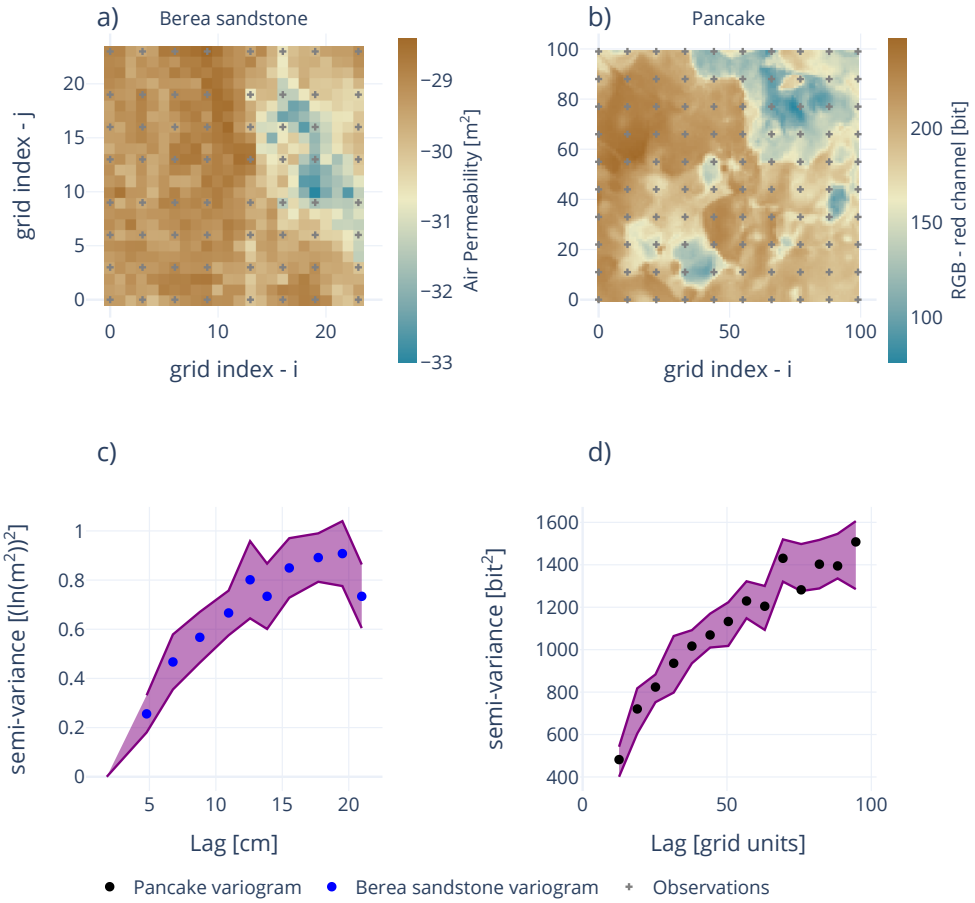



**Figure 3.2:** Data Overview: **a)** Permeability data associated with one of the faces (denoted as face 1) of the Berea sandstone sample obtained through the minipermeameter characterized by a 0.15 cm inner radius of the tip. Data are originally published and described in Vincent C Tidwell and John L Wilson (1999) and Vincent C. Tidwell and John L. Wilson (1997). **b)** Spatial distribution of the data associated with the pancake setting (see also figure 3.1 a)), color gradation being adjusted to match the corresponding visualization related to the Berea sandstone sample. Symbols in (a) and (b) correspond to the data employed in our exemplary analyses. **c, d)** Empirical variogram obtained considering the sampled data depicted in (a) and (b) for the Berea sandstone grid sample (**c**, blue circles) and the pancake dataset (**d**, black circles). The purple area corresponds to the uncertainty bounds estimated for the variograms.

enables us to perform the same types of analyses for the two selected datasets and consistently compare results across these.

### 3.4.2 Empirical variogram estimation

Empirical variograms are estimated using the Python package SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a). The package offers various options to this end. The scientific demo application integrates nine out of the ten binning algorithms implemented in SciKit-GStat (table 3.1). Depending on the binning algorithm, the user may select the number of lag classes for the evaluation of the variogram and the associated confidence limits. The largest separating distance at which point pairs are formed can be set directly or selected from predefined values such as, e.g., the median separating distance. The semi-variance of the resulting population of increments corresponds to the sample variogram for a given separation distance (or lag) and can be estimated through one of the five implemented estimators (table A.1). In case of a positively skewed data set and in the presence of outliers, we recommend the use of robust semi-variance estimators (see, e.g., Table A.1).

The empirical variogram for the pancake dataset (fig. 3.2 d) is estimated upon relying on Matheron semi-variance (Matheron 1963) according to 14 evenly spaced bins. The largest separating distance between a point pair was set to 100 grid units, thus coinciding with the length of the side of the domain across which data are sampled. Visual inspection of the results shows that the empirical variogram is characterized by a nugget/sill ratio of about 0.25. This is deemed as a remarkable amount of the total observed variability that could not be explained by the observed degree of spatial dependence (or correlation) of the target quantity.

The empirical variogram for the data associated with the Berea sandstone sample is depicted in figure 3.2 c). The KMeans based binning algorithm (see table 3.1) is employed to form 10 lag classes up to the largest considered lag of 24 cm. Similar to the pancake dataset, this corresponds to the length of the side of the domain across which data are sampled. The semi-variance is evaluated using the Matheron estimator, consistent with the pancake dataset. These results (see fig.3.2) suggest that the nugget/sill ratio of the empirical variogram might be smaller for the Berea than for the pancake dataset.

### 3.4.3 Uncertainty bounds of the empirical variogram

The key element of the application is the possibility to propagate observation uncertainties onto the estimation of the empirical variogram. These are then ultimately

**Table 3.1:** Overview of all lag class binning methods implemented in SciKit-GStat (from Mirko Mälicke, Möller, et al. (2021a)).

| Function | Identifier | Description | Implementation |
|---|---|---|---|
| Equidistant lags | `'even'` | *N* lags of same width; Almost always used. | Mirko Mälicke, Möller, et al. (2021b) |
| Uniform lags | `'uniform'` | *N* lags of same sample size; Estimates are based on the same sample size & no empty bins | Mirko Mälicke, Möller, et al. (2021b) |
| Sturge's rule | `'sturges'` | Equidistant lags derived from Sturge's rule; use for small normal distributed distance matrices | Virtanen et al. (2020) |
| Scott's rule | `'scott` | Equidistant lags derived from Scott's rule; use for large datasets | Virtanen et al. (2020) |
| Freedman-Diaconis estimator | `'fd'` | Equidistant lags; use for small datasets with outliers in the distance matrix | Virtanen et al. (2020) |
| Square-root | `'sqrt'` | Equidistant lags; Very fast function, but usually not recommended | Virtanen et al. (2020) |
| Doane's rule | `'doane'` | Equidistant lags; based on data skewness, use for small non-normal distance matrices | Virtanen et al. (2020) |
| K-Means | `'kmeans'` | Non-equidistant lags; clustered distance matrix is used as binning; slow but statistically robust | Pedregosa et al. (2011a) |
| Hierarchical Clusters | `'ward'` | Non-equidistant lags; clustered distance matrix is used as binning; Based on Ward's criterion for minimizing cluster variance. Computational intensive | Pedregosa et al. (2011a) |

**Table 3.2:** Overview of all semi-variance estimator functions implemented in SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a) (modified after Mirko Mälicke, Möller, et al. (2021a)).

| Estimator | Identifier | Description | Reference |
|---|---|---|---|
| Mathéron | `'matheron'` | Default, most popular estimator | Matheron (1963) |
| Cressie-Hawkins | `'cressie'` | Power transformation based - robust to outliers | Cressie and Hawkins (1980) |
| Dowd | `'dowd'` | Median based, fast estimator for non-normal distributed residuals | Dowd (1984) |
| Genton | `'genton'` | Percentile-based estimator - powerful for skewed residuals, but very computationally intensive | Genton (1998) |
| Shannon Entropy | `'entropy'` | Information theory metric focusing on information content of residuals | Shannon (1948) |

employed to characterize the empirical variogram through bounds of uncertainty. We note that we specifically tailor our approach to empirical variograms and consider the underlying random field to be either second-order stationary or to satisfy the intrinsic hypothesis.

The first option available relies on the quantification of a confidence interval through the standard deviation of the empirical density of the (zero-mean) residuals of the squared increments of the target quantity corresponding to a given lag. The approach is straightforward and can be used, e.g., when no other information on observation uncertainty is available. The characteristic width, $\delta$, associated with the confidence interval is evaluated as:

$$\delta = z\frac{\sigma}{\sqrt{N}} \tag{3.1}$$

where $\sigma$ is the standard deviation of sample squared increments, $N$ is sample size, and $z$ is the $z$-score of the desired confidence level of the standard Normal distribution function $Z$. As the uncertainty bound is evaluated on the basis of the confidence interval of the mean point pair residual, the central limit theorem is expected to hold. The latter may be violated, though, when using a high number of lag classes in combination with a small sample size for some of these. We thus encourage the user to carefully inspect the histogram of point pairs associated with all lag classes. Note that in the following we consider typical 95% confidence

intervals for the Berea sandstone sample variograms. This approach is employed for the Berea sandstone scenario, as no further information on actual observation uncertainties is available.

The second approach is based on the evaluation of semi-variance values for each lag class in the context of a $k$-fold statistical robustness test. The application implements options to subdivide each class associated with a given lag into 3, 5, 7, or 10 folds and evaluate the semi-variance $k$ times for $k-1$ folds comprised in the bin. Upon relying on 100 iterations, values of squared increments are allocated randomly to the folds and the uncertainty bounds are evaluated for the $i \times k$ estimated semi-variance values. The number of iterations can be adjusted by the user. The key assumption underlying this approach is that the robustness of semi-variance values calculated for a large number of smaller subsets strongly reflects the true uncertainty associated with the semi-variance. The main advantage of the methodology is that it does not require any particular assumptions about the residual distribution because it simply evaluates the actual semi-variance given the reduced size dataset. Otherwise, a weak element of the procedure is that it is quite sensitive to the settings of the robustness test (especially to sample size). If the number of pairs within each lag class is not sufficiently large, the $k$-subsets might be too small to infer robust statistics. Otherwise, when considering large samples, the computational demand for this iterative process needs to be carefully considered and might hamper its efficiency. The approach is well suited to tackle scenarios where the user cannot quantify observation uncertainties and the amount of data enables one to avoid resorting to the simple approach encapsulated in eq. 3.1.

The third approach implemented is set within a numerical Monte Carlo simulation context. It is here demonstrated considering the (re-sampled) field of observations resulting from the original data. The array of observations is replaced by a randomly generated array, given a specified aleatory uncertainty measure. Here, we consider three kinds of uncertainty metrics that can be propagated onto the variogram.

A first metric is based on considering measurement error to be represented by a uniform distribution with a given mean (corresponding to the observed value) and a minimum/maximum value specified by the user, which we will denote as *measurement error bounds*. This enables one to assign the same weight to all of the values included in the support of the distribution.

A second metric relies on the standard error of the mean (SEM) of the observations. The latter needs to be specified by the user as an input parameter to the procedure. By doing so one considers observation errors to be characterized by a

Normal distribution with a given mean (corresponding to the observed value) and standard deviation, $\sigma$, given by:

$$\sigma = SEM * \sqrt{N} \tag{3.2}$$

where $N$ corresponds to the sample size.

A third option considers specifying directly the standard deviation of the aforementioned Normal distribution.

Resorting to a given observation error metric depends on available metadata, i.e., on additional information eventually complementing the analyzed dataset. For example, some manufacturers of physical sensor devices might supply SEM values, while modeling results might rather be associated with a well defined error bound. It is quite often possible to estimate one of the three aforementioned metrics from expert knowledge. When knowledge on the uncertainty metrics described above is available, the Monte Carlo approach is preferable, as compared to the other options described, which are based on stronger assumptions. If available, SEM is possibly a preferred aleatory uncertainty measure, as it describes observation uncertainties by definition.

The empirical variogram is then represented through the evaluated uncertainty bounds. These embed the concept of uncertainty we propose to employ in the context of geostatistical analyses fully encapsulating uncertainty in the empirical variogram. In line with the spirit of our study, we then obtain a collection of variogram models (and ensuing parametrizations) that are consistent with an interpretation of a variogram based on the concept of uncertainty bounds. As previously stated, the ensuing collection of models (and parameters) can then be employed to propagate variogram uncertainty onto geostatistical analyses (i.e., in the context of estimation and/or simulation scenarios).

### 3.4.4 Theoretical model performance metrics

Accounting for uncertainty bounds of the empirical variogram enables one to consider **a)** multiple parameter sets conditional to a given model and/or **b)** multiple competing model formulations that are all consistent with the level of uncertainty associated with observations. Thus, model selection is a major epistemic source of uncertainty, directly tied to our research hypothesis H2 (Section 3.2). A key research question tackled through the tool hinges on the identification of theoretical variogram models that, following a given parametrization, are fully comprised

**Table 3.3:** Overview of all theoretical variogram model functions implemented in SciKit-GStat (modified after Mirko Mälicke, Möller, et al. (2021a)).

| Model | Identifier | Description | Implementation |
|---|---|---|---|
| Spherical | `'spherical'` | Short ranged correlation length, popular model in geoscience; for smooth, but steep gradients in fields. | Burgess and R. Webster (1980) |
| Exponential | `'exponential'` | Long ranged for smooth fields with less steep gradients. | A G Journel and Huijbregts (1976) |
| Gaussian | `'gaussian'` | Mid ranged for sharply changing fields | A G Journel and Huijbregts (1976) |
| Cubic | `'cubic'` | Similar to Gaussian models, but with a shorter correlation length. | Montero et al. (2015) |
| Matérn | `'matern'` | Has an additional smoothness parameter to adapt shapes between Exponential and Gaussian models. | Zimmermann et al. (2008) |
| Stable | `'stable'` | Has an additional shape (power) parameter to adapt the range. | Montero et al. (2015) |

within the identified uncertainty margins.
Model formulations available in the toolbox are listed in table 3.3.

The toolbox implements a variety of metrics to assess model performance, as detailed in the following Sections.

### Root Squared Mean Error - RMSE

An adjusted version of the root squared mean error (*RMSE*) can be used as a goodness-of-fit metric for a given variogram model parametrization. In this context, for uncertainty bounds of width $\Delta\gamma = u - l$ ($u$ and $l$ being an upper and lower limit) at a given lag and for a target model variogram $\gamma$, we set $RMSE := 0$ if $l < \gamma' < u$. Otherwise, *RMSE* is evaluated as:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N} min(\gamma'_i - u, \;\; l - \gamma'_i)^2}{N}} \tag{3.3}$$

where $N$ is the number of lags at which the empirical variogram (and hence the uncertainty bounds) is estimated from available data. We note that *RMSE* is used to assess the model solely on the basis of the fit of the theoretical model to the empirical variogram uncertainty bounds. As such, it does not provide information about the ability of a given model (or model parameter set) to correctly estimate or simulate the analyzed quantity at unobserved locations.

### Cross-validation through Ordinary Kriging

As a second metric that can be employed to evaluate the performance of a given variogram model, we also rely on a classical leave-one-out cross-validation. For $Z(s_N)$ observations, the model is applied considering $N - 1$ observations to estimate $Z(s_N)^*$ at the omitted location via Ordinary Kriging. The ensuing differences between observed and interpolated values are then assessed upon relying on their associated RMSE. A value of $RMSE = 0$ indicates that the model is capable of reproducing the observations. Increasing values of *RMSE* correspond to an increased mismatch between observation and interpolation-based estimates.

### Deviance Information Criterion - DIC

The application also allows for the evaluation of a given variogram model upon relying on model selection criteria. These are employed to evaluate the relative skill of a candidate model (as compared against other model analyzed) to interpret

available observations. We rely here on formal model selection criteria to evaluate (in a relative sense) the ability of each of the models we consider to be consistent with the estimated uncertainty bounds related to the empirical variogram. Among the various model selection criteria proposed in the literature to discriminate amongst models (see e.g. Höge et al. (2018) and Riva, Panzeri, et al. (2011)), we rest here on the Deviance Information Criterion *DIC* (Spiegelhalter, Best, Carlin, and Van Der Linde 2002; Spiegelhalter, Best, Carlin, and Van der Linde 2014), which is a generalization of the Akaike Information Criterion *AIC* (Akaike 1973; Hurvich and Tsai 1989).

The deviance $D$ of a given model (parameterized through a set of parameters collected in vector $\vec{\Theta}$) is given by:

$$D(\vec{\Theta}) = -2ln(L) \tag{3.4}$$

where $L$ is the likelihood function of the considered theoretical variogram model. Here, we consider the following definition of a negative log-likelihood function from (Lark 2000, eq. (14)):

$$\begin{aligned} L(r, s|\vec{\hat{m}}, \hat{\sigma}^2, \vec{z}) = \frac{n}{2}ln(2\pi) + \frac{n}{2} - \frac{n}{2}ln(n) + \frac{1}{2}ln|\vec{A}|+ \\ \frac{n}{2}ln\Big((\vec{z} - \vec{\hat{m}})^T A^{-1}(\vec{z} - \vec{\hat{m}})\Big) \end{aligned} \tag{3.5}$$

where $\vec{z}$ is a vector whose entries correspond to $n$ available observations; $r$ and $s$ are the range and sill of the considered variogram model, respectively; $\vec{\hat{m}}$ is a vector of maximum likelihood estimates of the available data at the observation points (see also Lark (2000, eq. (12))); $\hat{\sigma}^2$ is a maximum likelihood estimate of the sample variance (see also Lark (2000, eq. (13))); and $A$ is the auto-correlation matrix for the sample and specified (Lark 2000, eq. (9)) as follows:

$$\begin{aligned} \vec{A}(i,j) &= 1 && i = j, \\ &= s\{1 - f(\vec{x}_i - \vec{x}_j|r)\} && i \neq j \end{aligned} \qquad s = \frac{c}{c_0 + c} \tag{3.6}$$

Here, $i, j$ are the indices corresponding to the observation locations; $f(\vec{x}_i - \vec{x}_j|r)\}$ is the spatially structured component of the variogram model conditioned only to the range parameter, $r$; $s$ is a term associated with the nugget to sill ratio, $c$ and $c_0$ corresponding to the variogram sill and nugget, respectively.
We note that equation (3.5) underlies the assumption of Gaussian distribution for the associated variogram model parameters.

The deviance information criterion penalizes a model with respect to its competing counterparts through the complexity of its parametrization. The latter is quantified via the concept of *effective parameters*, $pD$, defined as:

$$pD = \overline{D(\vec{\Theta})} - D(\overline{\vec{\Theta}}) \tag{3.7}$$

where $\overline{\vec{\Theta}}$ is the mean of all parameters associated with a given model (i.e., a given functional format of the variogram) and $\overline{D(\vec{\Theta})}$ is the sample mean of deviance evaluated across all models and parameter sets.
Considering the sample probability density of model parameters, $DIC$ is then evaluated as:

$$DIC = \overline{D(\vec{\Theta})} + pD \tag{3.8}$$

Following Spiegelhalter, Best, Carlin, and Van Der Linde (2002), one could assess $pD$ upon relying on the mode or on the median of the distribution of model parameters assessed through model characterization on the basis of the uncertainty bounded empirical variogram. All of these options are implemented in the toolbox. As an additional option to evaluate $pD$, we also consider Gelman et al. (2014, eq.7.10):

$$pD = \frac{1}{2} var(D(\vec{\Theta})) \tag{3.9}$$

This formulation always yields positive values for $pD$, which, in turn, makes the use of $DIC$ very intuitive. Thus, we use the latter approach and formulation for this study and as a default option for the toolbox due to its readily intuitive nature.

#### Structural risk minimization

Another area where one usually needs to balance between model complexity and over-fitting is machine learning. In this context, an appealing framework is provided by the concept of structural risk minimization (Vapnik and Chervonenkis 1974). While the toolbox implements a variation of the latter, we not pursue it further in this study. The interested reader is referred to Appendix .1, where the available option from the toolbox is briefly illustrated.

### 3.4.5 Variogram model assessment

The toolbox function for manual variogram fitting enables the user to **a)** select any of the available theoretical variogram models and **b)** interactively parameterize these

for the desired number of model parameter sets while considering the estimated uncertainty bounds related to the given empirical variogram. The quantitative metrics described in Section 3.4.4 are evaluated for the collection of all models and associated parameters employed for data interpretation. We recall that the objective here is to sample the set of possible theoretical model functions and their parametrizations. We further note that other techniques conductive to (posterior) distributions of model parameters such as, e.g., acceptance-rejection sampling (e.g. Russian et al. 2017, and references therein) are not yet embedded in the toolbox. Otherwise, the modular nature of the toolbox facilitates the integration of additional simulation tools. Thus, users are foreseen to be able to choose among various approaches (as soon as these are implemented) to obtain a collection (i.e., an ensemble) of candidate theoretical models (and ensuing model parameter sets) in their scenarios of interest.

The collection of model functions and ensuing parameter sets are then filtered to retain the best-performing models. With reference to this issue, our toolbox implements an interactive, feature-rich selection interface. The user may perform model selection analysis upon relying on one of the metrics detailed in Sections 3.4.4 to 3.4.4 or comparing the results associated with the use of all of these. While the demo application is currently confined to a given number of options for model selection, its flexible structure enables one to seamlessly expand on these. The user can either **a)** retain a fixed amount of parameter sets (e.g 10 best ones), **b)** retain a fixed amount of parameter sets stratified by model type (like 3 Gaussian, 3 Spherical, and so on) or **c)** calculate a threshold by defining an acceptable relative deviation from the best parameter set. Only the selected parameter sets are then considered for the estimation of a Kriging uncertainty bound, as described in the following Section.

### 3.4.6 Kriging uncertainty bounds

Our toolbox includes four different Kriging algorithms from GSTools (Müller et al. 2021a). While the default option is Ordinary Kriging, the user may select to rely on either Simple or Universal Kriging. If auxiliary information is available, external drift Kriging can be used, incorporating such data as drift. For Simple Kriging, the mean of the field needs to be specified by the user. For Universal Kriging, a linear or a quadratic internal drift term is currently available.

To propagate uncertainties to a Kriging application, each of the selected models is used with each of the associated parameter sets to project the data onto a target grid. While the size of such grid can be specified interactively by the user, the

toolbox also implements some options to automatically evaluate the coordinate locations for each grid cell.
The following option is of interest for our demo software. In case the user uploads a field (such as, e.g., the pancake scenario we consider) and uses the toolbox to sub-sample it, the toolbox automatically uses the grid of the originally uploaded field, if Kriging is applied to the **sub-sample**. The advantage of this procedure is that one can associate a value from the originally uploaded field to any location of the target grid which is not tied to the sub-sample. This enables the user to objectively assess the overall performance of each kriged field.

The uncertainty propagated onto the Kriging-based estimates corresponds to the range of interpolation estimates associated with each grid location. We note that the number of available estimates matches the number of selected models and model parameter sets. In some cases, it is possible that a given parameter set is conducive to kriged estimates that markedly differ from those of the remaining models and model parameter sets, either across the whole target field or only within a certain region. Thus, an important feature implemented in the tool enables one to examine and compare the contribution of a given parameter set to the overall uncertainty of the results. We do so upon relying on the Shannon entropy (Shannon 1948) associated with the collection of predictions at each grid cell/node. The Shannon entropy is defined as:

$$H = -\sum_{i=0}^{M} p_i * log_2(p_i) \tag{3.10}$$

where $p_i$ is the empirical probability of non-exceedance of the $i-th$ value of the collection of estimates related to a target location in the domain. The Shannon entropy is well suited to analyze redundancy within a model and model parameter collection (R. Loritz et al. 2018; M. Mälicke et al. 2020, e.g.). Non-exceedance probabilities are evaluated upon subdividing the range of the obtained interpolated values across the whole domain into a number $M$ of bins, which is typically set to the number of selected parameter sets. This is tantamount to considering the same binning for obtaining $p_i$ at all grid locations.

In order to compare Shannon entropy across datasets and assess the agreement of estimates between the models and ensuing parameter sets, the Shannon entropy is normalized. A suitable normalization considers the Shannon entropy of a distribution of $M$ uniformly sized bins, $H_{max}$. For the whole domain, the same $H_{max}$ will be considered. The normalized Entropy $H_n = \frac{H}{H_{max}}$ is a measure of how close the distribution of estimates in each grid cell is to a uniform distribution (corresponding to $H_n = 1$). Thus, it can be used to identify grid locations of high estimate variability

within a set of Kriging results. It can also be used to compare results across multiple datasets, with respect to the number of parameter sets selected. We note that a $H = 0$ for a given grid location implies that all estimates reside within the same bin. This does not imply that all estimates are numerically close, because $M$ (i.e., the number of selected parameter sets) might be quite small in some cases. Here, we use the normalized Shannon Entropy to identify regions of the domain where there is high estimate variability, that can then be compared across multiple datasets.

## 3.5 Results

### 3.5.1 Variograms and related uncertainty

Here, we illustrate all details of the application with reference to the pancake sample. We then present and analyze our findings for the Berea sandstone sample (see fig. 3.2, a&b). Visual inspection suggests that the two fields display a similar spatial structure and their variograms exhibit a similar pattern (fig. 3.2 c&d). The two variograms differ clearly with respect to the width of their uncertainty bounds. The latter is larger for the Berea sandstone dataset. We note that, taking only the uncertainty bounds into account, almost any theoretical model might fit each of these empirical variograms and a prior selection of a specific model is not justified.

### 3.5.2 Theoretical variogram models and associated performance metrics

With respect to the uncertainty bound of the empirical variogram, almost none of the theoretical model (here manually parameterized) can be rejected (see fig. 3.3 a). Figure 3.3 b) provides a graphical depiction of the relevant metrics for all of these models. Here, the different model types are listed in the first column and each band represents one set of model parameters, color gradation being indicative of a given model type. The first connection to the second column ranks the models by their fit in terms of *RMSE* (eq. (3.3)). For visual reasons, *RMSE* values are ranked and grouped into quartiles, with the 25% best performing model parameter sets at the top of the column. The bands spread out significantly and are not grouped by model type anymore. This stresses the visual impression that no model instances are significantly off when considering empirical variogram uncertainty.

The third column in figure 3.3 b) ranks the model parameter sets by the corresponding *DIC* value (eq. (3.8)). By design the model parameter sets are grouped by

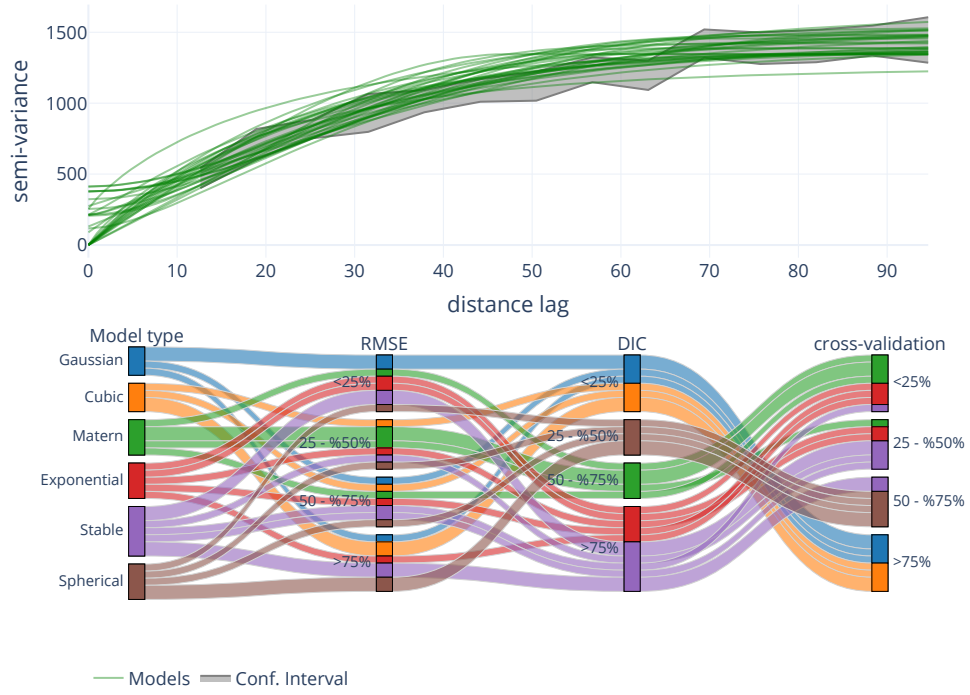



**Figure 3.3: a)** Uncertainty bounds (gray area) associated with the empirical variogram related to the **pancake** dataset, including with all theoretical variogram models fitted (green curves). **b)** Parallel coordinates plot for the models depicted in (a) showing the considered performance metrics, i.e., *RMSE* (2nd column), *DIC* (3rd column), and cross-validation (4th column). The first column groups the individual models by their type and corresponding color gradation. For each of the measures, the models are ranked into quartiles; as an illustrative example, we consider *< 25%* to delineate the collection of the best performing models.

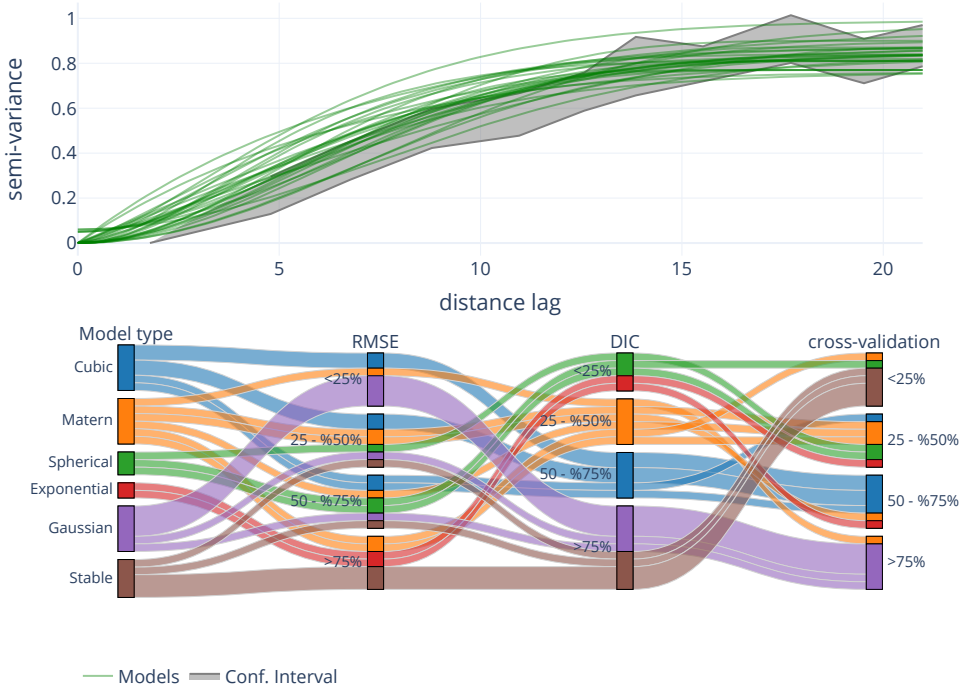



**Figure 3.4: a)** Uncertainty bounds (gray area) associated with the empirical variogram related to the **Berea sandstone** dataset, including all theoretical variogram models fitted (green curves). **b)** Parallel coordinates plot for the models depicted in (a) showing the considered performance metrics, i.e., *RMSE* (2nd column), *DIC* (3rd column), and cross-validation (4th column). The first column groups the individual models by their type and corresponding color gradation. For each of the measures, the models are ranked into quartiles; as an illustrative example, we consider $< 25\%$ to delineate the collection of the best performing models.

model type, as *DIC* is a performance metric on model type and does not distinguish among the different parametrizations. Similar to other information criteria, *DIC* grounds the suitability of a given model on the likelihood of the model parameters, given the sample distribution. In terms of *DIC*, the Cubic and Gaussian models perform best, while exponential and stable models are characterized by poorer performance. We further note that, due to manual parametrization, the model collection sizes are quite small and *DIC* values might change when additional parameter sets are added to the collection.

The last column in figure 3.3 b) provides a ranking of the model parameter sets grounded on cross-validation results. The predictive power of each (manually fitted) model and model parameter set is assessed by applying Kriging interpolation via a leave-one-out cross-validation for all observation points. Interestingly, all bands cross on the connection of the third and fourth column (see fig. 3.3 b). Thus, model and model parameter ranking change to favor Matérn parametrizations over Gaussian and cubic models. One has to keep in mind that for the purpose of our demonstration Kriging is only applied to the sample considered and model performance might differ for unobserved locations. This is expected to depend on the density and structure of the observation points.

Taking all of the above elements into consideration, one can conclude that the uncertain observations allow for various models and ensuing parametrizations to be considered as suitable in a virtually indistinguishable way. This is largely supported by the *RMSE* results. In practice all parameter sets would be accepted in any least-square based automatic procedure. The *DIC* criterion rests on variogram likelihoods given the sample distribution and does favor specific model types over others. This can loosely be seen as an assessment of how and which models an automated maximum likelihood approach favors. Results from cross-validation, which is conceptualized as a visualization of the training error of the model, are in contrast with those provided by *DIC*. This finding is unexpected and raises some interesting questions about when and how to apply automatic and semi-automatic fitting procedures.

The shape of the uncertainty bound is slightly different for the Berea sandstone sample and is characterized by a less pronounced increase of semi-variances within the first few lag classes. Similar to what can be observed for the pancake dataset, all theoretical variogram functions (green curves in fig.3.4a) appear to be equally compatible with the estimated uncertainty bounds. This results in a straightforward parametrization of Gaussian or Gaussian-shaped Matérn models. Otherwise, the exponential and exponentially shaped stable models appear not to be fully compatible with the estimated uncertainty bounds, with special reference to the upper

limit of these. This behavior is also reflected by the *RMSE* values in the second column (fig. 3.4 b), which rank the exponential model parametrizations slightly lower than for the pancake dataset.

According to the values of *DIC* (fig. 3.4 b), spherical and exponential models are highly ranked, as opposed to their Gaussian and stable counterparts. Similar to the pancake dataset, results of cross-validation based on Kriging appear to favor some models that were ranked low according to the other metrics employed. While the Gaussian models still perform worse than others, the stable models are ranked significantly higher according to this metric. It is worth noting that all parameter sets of the stable model lie in the best performing quartile in terms of Kriging cross-validation, even those that visually show notable deviations when juxtaposed to the uncertainty bound. The same finding holds for the exponential model. All parametrizations of the latter are ranked in the lowest quartile for RMSE, in the highest quartile for *DIC* and close to median for the cross-validation metric.

In summary, there is no model type that is ranked low consistently by all metrics across both datasets considered. Likelihood- and uncertainty bound- driven metrics do not yield a unique and unequivocal outcome when analyzed jointly and neither of these is entirely supported by Kriging cross-validation across the collection of the corresponding parameters. Possibly, a clear conclusion is that Gaussian models should be avoided, although they appear to fit the uncertainty bound best.
All of these results suggest that any kind of automatic variogram fit should always be complemented by careful inspection of results of the kind we illustrate, on the basis of multiple metrics, each revealing a particular aspect of uncertainty.

### 3.5.3 Kriging uncertainty bounds

A collection of about 30 different model parameter sets has been identified for the pancake dataset. A critical element in the analysis of the way variogram uncertainty propagates onto Kriging results is the possibility of ranking model parameter sets according to the performance metrics selected. This is accomplished through the implementation of a filtering step in the tool. The latter allows for various functions to filter the model parameter sets with respect to one of the performance metrics detailed in Section 3.4.4.
All models parameter sets are then ranked with reference to each of the metrics considered (i.e., *RMSE*, *DIC*, and cross-validation). The filter rejects the 10% worst parameter sets for each metric. For both datasets we find that six instances were rejected, most of these associated with Gaussian models, which is seen to be ranked lowest in more than one metric.

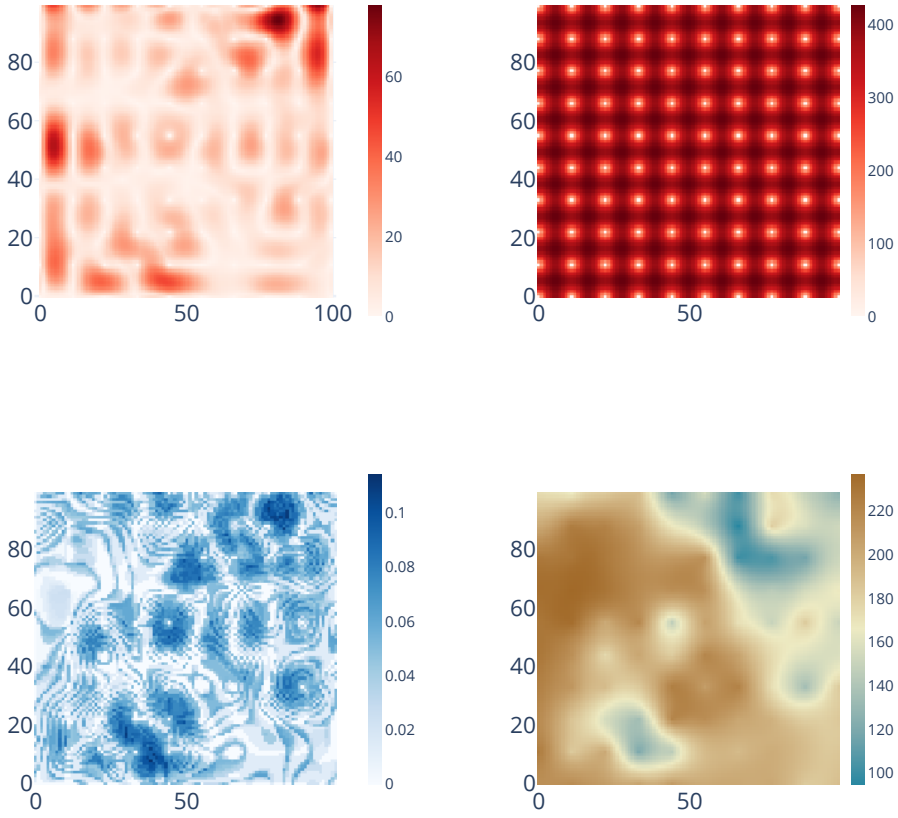



**Figure 3.5:** Kriging estimation results for the **pancake dataset** after re-sampling on a **regular grid**. Width of the interval of variability of **a)** Kriged values and **b)** Kriging error variance values associated with all models analyzed at each cell across the domain . Note that a zero variance value means that the Kriging variance is the same for all models. Large values imply a variable Kriging error variance (It's illustrating the variability of variances). **c)** Normalized Shannon entropy of all model estimates **d)** Kriging interpolation result for the best model parameter set (mean rank of RMSE, *DIC* and cross-validation).

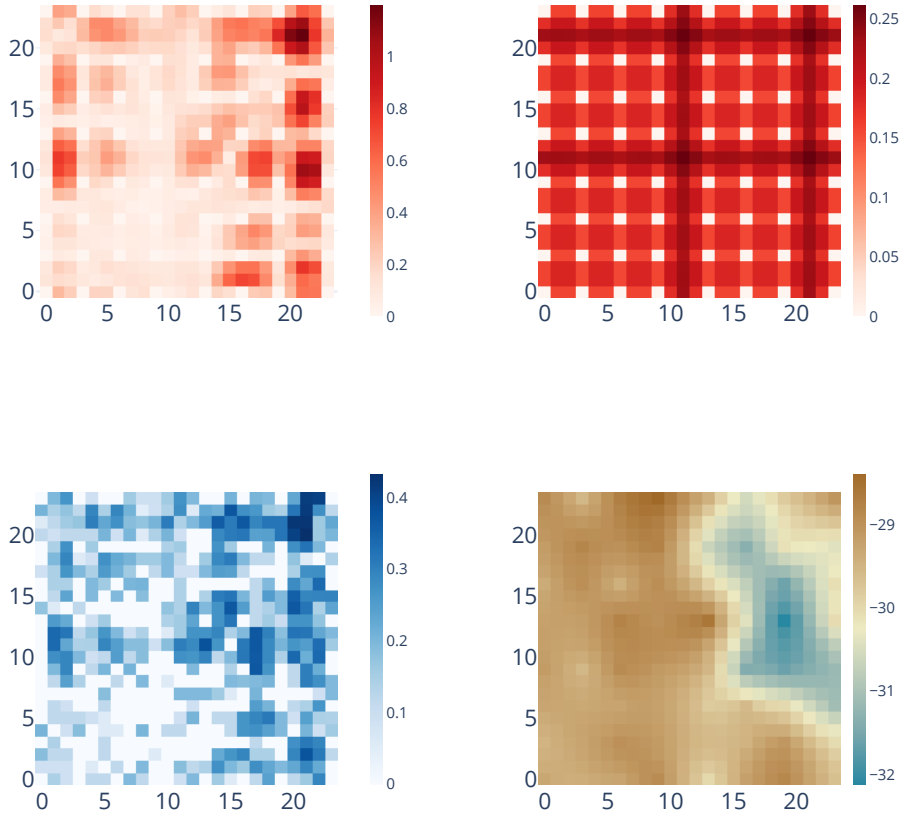



**Figure 3.6:** Kriging estimation results for the **Berea sandstone dataset** after re-sampling on a **regular grid**. Width of the interval of variability of **a)** Kriged values and **b)** Kriging error variance values associated with all models analyzed at each cell across the domain . Note that a zero variance value means that the Kriging variance is the same for all models. Large values imply a variable Kriging error variance (It's illustrating the variability of variances). **c)** Normalized Shannon entropy of all model estimates **d)** Kriging interpolation result for the best model parameter set (mean rank of RMSE, *DIC* and cross-validation).

Propagating variogram uncertainties onto the Kriging results generally leads to large corresponding uncertainty bounds. By taking different model parametrizations into consideration, one finds a spread of Kriging interpolation results which is typically of about 25 units, while attaining peaks of about 70 units (fig. 3.5 a), which corresponds to about 30% of the range of values of the available data. The width of the Kriging uncertainty bounds is highly heterogeneous in space. In some areas the uncertainty bounds are not much larger than the observation uncertainty propagated into the procedure, while being markedly larger in other regions. In general, uncertainty band widths correlate with the location on the grid and most model parameter sets seem to disagree in terms of Kriged values close to the domain boundaries.

As expected, the Kriging error variance generally tends to vanish close to observation locations. Figure 3.5 b) shows the range of Kriging error variances for all selected model parameter sets. As expected, and consistent with the dense sampling arrangement, no particular spatial differences can be identified. We remark that a value of 0 in figure 3.5 b) implies that all Kriging variance values coincide, all models being in agreement.

While the range of kriged values for a given unobserved location can be large, this can be due, in some cases, to a single parameter set or to a limited number thereof. This element can be investigate through the analysis of the entropy map of model Kriging results. Figure 3.5 c) depicts the spatial distribution of the values of the normalized information entropies. Here, a value of 1 or 0 implies large variability across the collection of estimates or that all estimates fall into the same bin, respectively. Values of the normalized information entropy of estimates (fig. 3.5 c) are largely spread evenly across the domain, even as some clusters are noted around a number of observation locations. Values are small in most areas. This finding suggests that only a few model parameter sets are driving the width of the uncertainty bands in figure 3.5 a). The entropy map displays a high level of spatial organization.

The Berea sandstone sample is characterized by similar results (fig. 3.6). There is a considerable overlap of larger normalized entropies and wider uncertainty bands. This is especially evident in the proximity of the right boundary of the domain. Normalized information entropy values are considerably larger for the Berea dataset (attaining values consistently $> 0.4$) than their counterparts associated with the pancake datset. This is consistent with the observation that a number of estimations differ by orders of magnitudes in these areas. Interestingly, one can also note the presence of the smallest values of the underlying field in this area. Figure 3.6 d) shows the Kriging interpolation result stemming from the model parameter set,

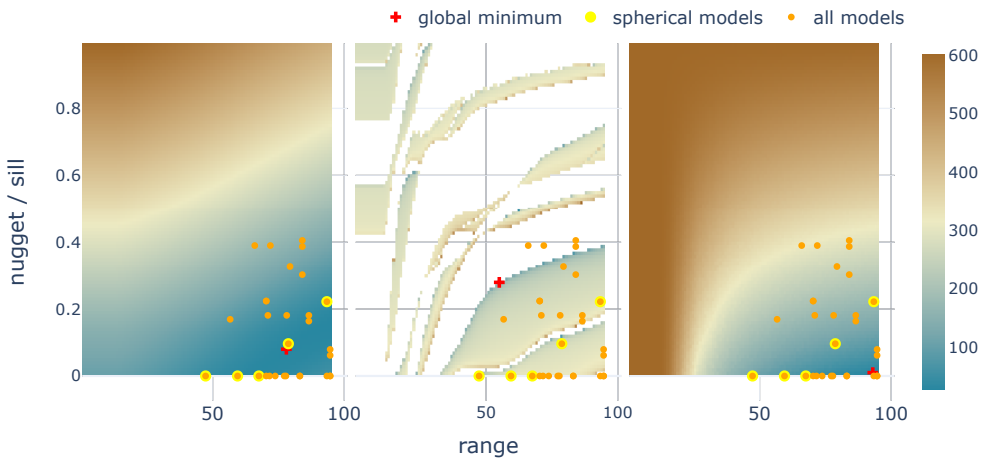



**Figure 3.7:** Results of the parameter testing phase for $100 \times 100$ combinations of sill/nugget ratio and effective range for a spherical model using the **pancake** dataset. **a)** *RMSE* (eq. 3.3) of the model fit to the uncertainty bound. Red or blue grading denotes larger or smaller metric values. **b)** deviance value for all parameter combinations. **c)** Leave-one-out cross-validation of the interpolated observation values. The orange symbols show the models and parametrizations used in manual fitting for all model types (the spherical model types are marked by thick yellow outline). The red cross marks the global minimum for each of the parameter tests.

best performing in terms of mean rank in all used performance metrics. From here, these areas, colored blueish, can easily be identified.

### 3.5.4 Identifiability of variogram model parameters for uncertain variograms

We exemplify the way one can select some parameter sets as optimal upon relying on the concept of variogram uncertainty bounds through a detailed analysis of the corresponding metrics based on the spherical variogram model. We do so because *a)* the model is seen to perform well for both datasets and *b)* this is the model type selected to demonstrate automatic fitting of empirical variograms with SciKit-GStat by (Mirko Mälicke, Möller, et al. 2021a).

Here, we use only the definition of the deviance in equation (3.4). As the mean deviance will be the same for all parameters, the value of *DIC* will be linearly

dependent on the deviance. At the same time, the negative log-likelihood function used in a maximum likelihood approach differs only by a factor of 2 from equation (3.4). Thus, this enables us to jointly interpret the results in terms of maximum likelihood (deviance) and method-of-moment (RMSE) approaches.

Each of the aforementioned metrics is evaluated (fig.3.7) for $100 \times 100$ combinations of range and nugget/sill ratios. A maximum nugget to sill ratio value of 1 was used (i.e., nugget and sill have the same (absolute) value). We note that considering values of this ratio larger than unity might hamper the usefulness of geostatistical approaches, which rest on the concept of a spatial correlation structure.
We added to the coordinates of the grid locations a white noise of about 0.1‰of the grid extent. Thus, any impact on the lag classes of the empirical variogram can be neglected. This was a necessary step to circumvent the issue that the Kriging system of equations be associated with too many instances of singular matrices. This likely originated from the regular spacing of the sampling locations, as further detailed in Section 3.6.3.

Large areas show a satisfactory performance in terms of *RMSE* of model fit (fig. 3.7 a). Moreover, figure 3.7 a) illustrates clearly that there is in fact parameter equifinality (K. Beven and Binley 1992) due to parameter interaction. We note that a global minimum is not readily identifiable across the parameter space. All of the results of the manual parametrizations here presented (orange dots) lie within the area of optimal parameter values (blue-graded region).
The deviance metric does not yield a result for several parameter sets (fig. 3.7 b); white regions). Here, the auto-correlation matrix $A$ (see eq. 3.6) is singular and could not be inverted. The extent of the blue-graded areas is considerably smaller than for the the *RMSE* metric. Finally, figure 3.7 c) illustrates the leave-one-out cross-validation metric for all $100 \times 100$ parameter combinations. Similar to the *RMSE*, all manually fitted parameter sets are contained in the blue-graded area within which good performance values of the metric are obtained. The global minimum is very close to the lower right corner (range of 93 and nugget to sill ratio of 0.01). We note that the *RMSE* and cross-validation metrics appear to be in a substantial overall agreement.

## 3.6 Discussion

Our analysis provides a clear evidence that **a)** uncertainty of the experimental variogram should not be ignored and **b)** the presented toolbox markedly facilitates assessment and propagation of such uncertainty onto a set of acceptable theoretical

variogram models and corresponding Kriged fields. The presented test cases yield insights into the ability of different performance metrics and on the goodness of individual members of a family of acceptable variogram models (in terms of their ensuing parameters). Interestingly, ranks of individual models and parameter sets is not the same for the different metrics. We show these elements for two data sets, by propagating uncertainties into the empirical variogram, assessing acceptable theoretical variogram models and their associated parameter sets, and comparing their kriged estimates across the system (also considering cross-validation) as well as the spreading of Kriging results at unobserved locations. Overall, a clear choice of a superior variogram model type or the identification of a best parameter set cannot be identified. A key asset of the presented toolbox is that it also provides enhanced understanding about how and where these uncertainties are caused. While a variety of selection algorithms or variogram parameter optimization approaches could be considered, for the purpose of our demonstration we choose a straightforward approach and eliminate models which perform poorest with respect to each of the performance metrics.

We acknowledge that the current stage of our work and version of the associated tool is restricted to an isotropic spatial covariance. Otherwise, a variety of environmental variables/quantities exhibit an anisotropic spatial covariance, also depending on the scale at which they are considered. As an example, we mention cold front precipitation bands, topography or macropores in soils, as well as sedimentological attributes or parameters characterizing variably saturated subsurface flow. The standard approach to detect a geometric anisotropy is to use directional experimental variograms. While our method to estimate uncertainties can be readily applied to the assessment of directional experimental variograms, this task is beyond the scope of the current study.

In the following we discuss **a)** the way the toolbox can assist interactive geostatistical analyses, **b)** variogram estimation under uncertainty and the related model evaluation, and **c)** the assessment of our driving hypotheses.

### 3.6.1 Interactive geostatistical analysis

Our software toolbox is built on established and well-tested software packages for numerical computing, visualization, and geostatistics. The implementation focuses on well-defined datasets. By providing clear interfaces and metadata, our API can be used to automate common tasks and build user interfaces such as those associated with the illustrated sample application. This is not only convenient but also considerably speeds up analysis workflows. As such, it empowers early

stage researchers and students to dive deeper into the material and scientists and practitioners to operate on data more effectively. This will ultimately favor practical implementations of new approaches. As an example: We would not have been able to manually parameterize so many models more effectively and faster than automatic approaches if it would not have been for an interactive slider element that enables one to adjust variogram parameters on the fly.

All this convenience comes at the cost of the implementation effort. As the user is less engaged with the actual, technical implementation than in more traditional scripting approaches, the software to be employed needs to be built in a generalized way. Achieving this element, in turn, needs comprehensive tests to ensure technical correctness. Tutorials and a complete and detailed documentation are equally important. Otherwise, the user will not be able to identify misuses and errors. A website and video tutorials are available for `SKGstat-Uncertainty`. Remarkably, the essential core of all calculations is implemented within other software products, each of these being carefully chosen to entail comprehensive testing and documentation. This enables the user to focus on analysis and visualization while being confident in the technical correctness of the results.

### 3.6.2 Uncertain variogram estimation and model evaluation

Using different methods for uncertainty propagation and estimation, we evaluate uncertainty bounds for the empirical variogram associated with the two showcases illustrated. This is a key result, as by simple visual inspection it is possible to estimate variogram parameters manually, thus favoring enhanced understanding on the system behavior. We note that at least one of each available theoretical variogram model type could be parameterized to fit into the uncertainty bound, or at least very close to it. This result confirms our hypothesis **H2**, as it makes the epistemic uncertainty relate to a prior model choice way more obvious than through a classical fitting procedure targeting empirical variograms. We rely here on manual procedures, due to their ease of usage and pedagogic potential. Otherwise, we stress that the toolbox is not limited to manual parametrization. Any suitable alternative approach implemented in Python (such as, e.g., ensemble learning methods or acceptance rejection) can be readily implemented into a new chapter of the tool. An additional added value of tool resides in the observation that data management and processing chapters, as well as subsequent analysis chapters, are modular.

An important limitation to our illustrative results, though, is that only one instance of an empirical variogram was estimated. The estimation is known to be sensitive to sampling strategies, especially sampling size, binning procedures, and

amount of lag classes used. While our modeling choices are based on expert knowledge, there might be a more suitable empirical variogram candidate, especially for the Berea sandstone setting. The purpose of this work, however, is to demonstrate the software package for exemplary analyses. Thus, we are confident that the illustrated insights can be adapted and transferred to focus on other critical aspects of empirical variograms, such as uncertainty bands based on systematic testing for different sample sizes.

Each selected variogram model parameter set was used in a Kriging interpolation context. As shown in figure 3.5 and 3.6, the corresponding interpolation results differ substantially. Considering all of the interpolation results, it was not possible to identify a unique model type (or parameter set) that clearly describes the spatial correlation structure of the field unequivocally better than all others. Otherwise, by combining insights from three different kinds of evaluation metrics, which focus on different aspects of a variogram fit, we can exclude model types. This is considered as an additional key result of our study and approach.

The Gaussian and cubic models are found to interpret adequately the empirical variogram associated with the pancake dataset (in terms of its uncertainty bounds). While *DIC* favors these two model formulations, the leave-one-out cross-validation excludes both of them regardless of their parametrizations. Inspection of the single interpolated grids based on Gaussian variogram models revealed that all of them produced considerable amounts of kriged values far outside the observation value space. We encourage the user of the application to interpret the results by considering the critical message that observation uncertainties exist and need to be comprehensively addressed. As analysis results may differ vastly, one could at least rely on insights obtained through modeling under uncertainty to exclude models (or parametrizations). This enables one to learn by rejection and enhance our knowledge from quantification of uncertainties, instead of neglecting these.

We assess sources of uncertainty that affect the different kinds of fitting procedures (least squares, maximum likelihood, manual) in very different ways and demonstrate the significance of our results in geostatistical applications . These insights, if taken into account, can assist in limiting the parameter space for a geostatistical analysis and lead to new knowledge about a field under investigation. It also bears an important implication with respect to quality and precision of the measured data. A smaller sample of highly precise observations will result in a small variogram uncertainty similar to what one could obtain through a large sample of less accurate data.

### 3.6.3 Model fitting and model parameters

A core decision taken for the application and in the exemplary study we present is the focus on manual variogram parametrization. Such a manual parametrization is a valid operational and educational choice. This is especially relevant in cases where one might select to renounce to some computing speed for a more thoughtful and detailed variogram analysis. Manual parametrization is straightforward, reflects a deep understanding of the variogram concept, and can be applied without the need for the implementation of optimization algorithms.

During the systematic exhaustive testing of variogram parameter values, the leave-one-out cross-validation calculation failed in several instances, especially for medium and small values of the effective range parameter. Due to the repetitive pattern underlying a grid, the number of distinct separating distances was significantly decreased in our examples. For the pancake sample, while the upper triangle of the distance matrix contains more than 4500 entries, these hold only 34 different distance values. The main reason for this is conceptually illustrated in figure 3.8. For the center point (in red), the Kriging equation system may be built solely from the surrounding blue points, which are all 1 or $\sqrt{2}$ units away from the center point. The two blue points at $y = 1$ are symmetrical with respect to the center. This means that the Kriging equation system is characterized by duplicated rows at the index of exactly these two points. This makes the Kriging matrix singular thus hampering its inversion. This was verified to happen quite often in our examples, as the Kriging algorithm built into SciKit-GStat limits the neighbor selection by the effective range of the variogram.
The same principle underpins the failure of the likelihood-based calculations observed in most cases.

In line with our first hypothesis that empirical variograms are uncertain, we present evidence that (Kriging) interpolation results cannot be simply limited to rely on a unique parametrization. Thus, geostatistical applications need to fully consider empirical variogram uncertainty bounds. Moreover, the parametrization of the variogram itself is markedly affected by different kinds of uncertainty. Our exemplary scenarios provide strong evidence of the basic assumption that propagating observation uncertainties into the variogram would lead to broad ranges of variogram parameter values that can be employed in a practical application. It is also apparent that a global minimum for a given metric can not be identified easily. Furthermore, our results show that there is no evidence that any automatic procedure would perform better, even if only one set of parameters is considered to be valid. And finally, the best manual fit is very close to the global minimum of RMSE,

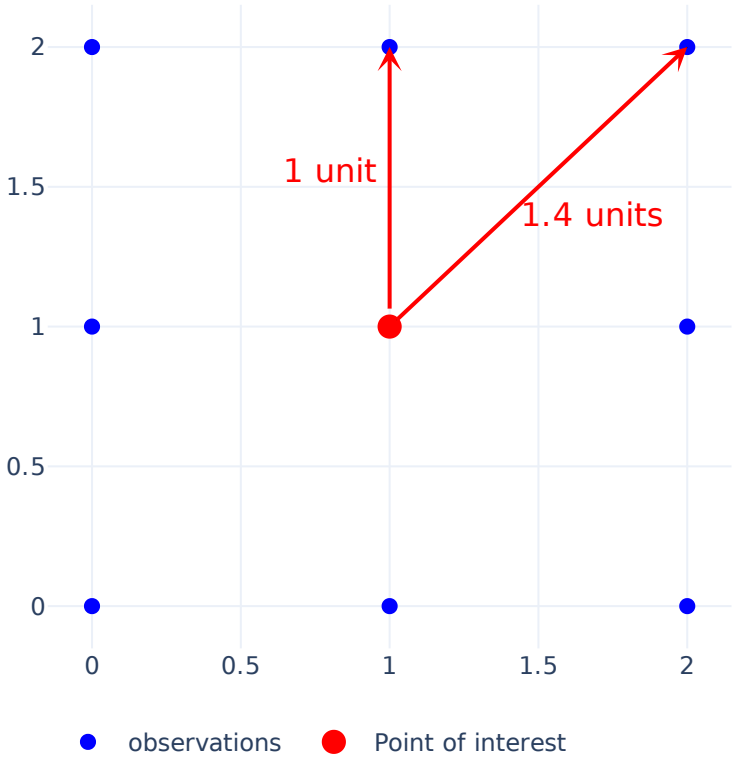



**Figure 3.8:** Conceptual figure of an observation grid in a dimensionless Cartesian space. Blue dots correspond to observation locations. The red dot is the point of interest, which is subject to estimation in a leave-one-out cross-validation. The figure illustrates the repetitive pattern of just two different distances being used in a Kriging application.

even if the difference to adjacent parameter sets is considered to be significant. While, in general terms, *RMSE* (fig. 3.7 a) and cross-validation (fig. 3.7 b) agree in identifying some sets of well-performing parameters (blue-graded), they also show some disagreements. These two figures suggest that even if a variogram model fits well to the uncertainty bound (or to the empirical variogram), cross-validation adds an additional (enriching) dimension against which the goodness of a performance should be assessed.

## 3.7 Conclusions

We introduce the toolbox *SciKit-GStat Uncertainty* and exemplify its use upon relying on sample data-sets pertaining to two different processes and scenarios. Our work leads to the following major conclusions

1. The toolbox is envisioned as a required extension of existing geostatistically-oriented computational tools and software. Our toolbox is built in a Python environment and includes the implementation of existing and new approaches to analyze, visualize, and quantitatively propagate uncertainties in variogram estimation onto kriging-based estimates and the associated variance. Its interactive nature empowers one to tackle uncertainty in a straightforward way and underpins the potential of the tool to play a key role in the context of research and educational contexts.

2. The toolbox enables one to explore the way various sources of uncertainty can imprint the results of a geostatistical analysis. Uncertainties considered through the toolbox arise from measurements (in terms of observations and location associated with these) as well as from the choice of an interpretive model and its parameters. Thus, the user can readily inspect various dimensions of uncertainty during variogram analyses. As a notable research element, we introduce and embed in *SciKit-GStat Uncertainty* the concept of replacing an empirical variogram (or semi-variance) through uncertainty bounds. This provides an original way to explore uncertainty, as imprinted onto the way one can evaluate the ability of a collection of models and ensuing parameters to perform variogram analysis upon relying on such uncertainty bounds. This is accomplished through a processing module that implements a suite of methods for the quantification of uncertainty associated with empirical variograms.

3. The software allows operating in a multi-model context and enhances our ability to interpret spatially correlated (random) fields. Exemplifying the

use of our toolbox with emphasis on manual variogram parametrization enables us to emphasize the value of the toolbox in the context of a pedagogical/educational perspective. The user can then explore the benefit of resorting to the joint use of various metrics, each of them providing a specific insight on the quantification of uncertainty, to yield a comprehensive depiction of system behavior and characterization. In this context, we investigate the way coupling the concept of variogram uncertainty-bounds with the joint analysis of multiple methods and metrics can contribute to disregard some models and parametrizations over others.

## .1 Structural risk minimization

Another available option to assess model parameter performance is derived form structural risk minimization. The work of Vapnik and Chervonenkis (1974) is focused on classification problems and support vector machines, its basic idea can be transferred to the scenario we consider. This is consistent with the observation that model selection can be viewed as a classification problem driven by uncertainty. The methodology introduced by Vapnik and Chervonenkis (1974) is designed to balance training errors and an expected over-fitting. Similar to an information criterion, the so-called capacity of the parameter space is employed, which in turn should measure model complexity through the minimization of:

$$J(\vec{\Theta}) = \varepsilon_{train}(\vec{\Theta}) + \lambda H(\vec{\Theta}) \tag{.11}$$

Here, $\varepsilon_{train}(\vec{\Theta})$ is a measures of training error and $H(\vec{\Theta})$ is a regularization term. The latter penalizes models with a higher level of complexity, in terms of parametrization. The value of the weight *lambda*, needs to be set by the user. We adapt this concept by interpreting the parametrization of a variogram model as the training of our model and combine it with the *pD* as described in Section 3.8. The scientific demo application evaluates $\varepsilon_{train}(\vec{\Theta})$ either with the *RMSE* (see Section 3.4.4) or the MAE as suggested by Vapnik and Chervonenkis (1974):

$$MAE = \sum_{i=1}^{N} min(l - \gamma_i', \;\; \gamma_i' - u) \tag{.12}$$

Where $\gamma_i'$ the modeled semi-variance at the $i-th$ lag class. As such, $MAE := 0$ for $l < \gamma' < u$.
With reference to the regularization term $H(\vec{\Theta})$ in equation (.11) one can set it either to equation (3.7) or equation (3.9).

The toolbox implements all combinations to evaluate equation (.11), but the exemplary demo application does not make use of these metrics.

# Part II

# Scientific software development

# A SciKit-GStat - scientific geostatistical software

# SciKit-GStat - a SciPy flavored geostatistical variogram estimation toolbox written in Python

The following chapter A is already published as a model description paper in *Geoscientific Model Development* as:

*Mälicke, M.: SciKit-GStat 1.0: a SciPy-flavored geostatistical variogram estimation toolbox written in Python, Geosci. Model Dev., 15, 2505–2532, https://doi.org/10.5194/gmd-15-2505-2022, 2022.*

Data and Code availability:

The source code of SciKit-GStat is available on Github. Additionally, each minor version is published as a code publication (Mirko Mälicke, Möller, et al. 2021a). The code to reproduce the figures made with SciKit-GStat, including the data samples shown, is available on Github and Zenodo (Mirko Mälicke 2021). Note that the data samples are also part of the SciKit-GStat documentation.

## A.1 Abstract

Geostatistical methods are widely used in almost all geoscientific disciplines, i.e. for interpolation, re-scaling, data assimilation or modeling. At its core geostatistics aims to detect, quantify, describe, analyze and model spatial covariance of observations. The variogram, a tool to describe this spatial covariance in a formalized way, is at the heart of every such method. Unfortunately, many applications of geostatistics rather focus on the interpolation method or the result, than the quality of the estimated variogram. Not least because estimating a variogram is commonly left as a task for computers and some software implementations do not even show a variogram to the user. This is a miss, because the quality of the variogram largely determines, whether the application of geostatistics makes sense at all. Furthermore, the Python programming language was missing a mature, well-established and tested package for variogram estimation a couple of years ago.

Here I present SciKit-GStat, an open source Python package for variogram estimation, that fits well into established frameworks for scientific computing and puts the focus on the variogram before more sophisticated methods are about to be applied. SciKit-GStat is written in a mutable, object-oriented way that mimics the typical geostatistical analysis workflow. Its main strength is the ease of usage and interactivity and it is therefore usable with only a little or even no knowledge in Python. During the last few years, other libraries covering geostatistics for Python developed along with SciKit-GStat. Today, the most important ones can be interfaced by SciKit-GStat. Additionally, established data structures for scientific computing are reused internally, to keep the user from learning complex data models, just for using SciKit-GStat. Common data structures along with powerful interfaces enable the user to use SciKit-GStat along with other packages in established workflows, rather than forcing the user to stick to the authors programming paradigms.

SciKit-GStat ships with a large number of predefined procedures, algorithms and models, such as variogram estimators, theoretical spatial models or binning algorithms. Common approaches to estimate variograms are covered and can be used out of the box. At the same time, the base class is very flexible and can be adjusted to less common problems, as well. Last but not least, it was made sure, that a user is aided at implementing new procedures, or even extending the core functionality as much as possible, to extend SciKit-GStat to uncovered use-cases. With broad documentation, user guide, tutorials and good unit-test coverage, SciKit-GStat enables the user to focus on variogram estimation, rather than implementation details.

## A.2 Introduction

Today, geo-scientific models are more available than they have ever been. Hence, producing in-situ datasets to test and validate models is as important as ever. One challenge that most observations of our environment have in common is they are non-exhaustive and often only observe a fraction of the observation space. A prime example is the German national rainfall observation network. Considering the actual size of a Hellmann observation device the approx. 1900 stations, the meteorological service operates, sum up to only 38 $m^2$. Compared to the area of Germany, these are non-exhaustive measurements.
If one takes an aerial observation, such as a rainfall radar, into account, at face value this can seem to be different. But a rainfall radar is actually only observing a quite narrow band in height, which might well be a few thousand meters above ground (Marshall et al. 1947). And it observes the atmosphere's reflectivity, not the actual rainfall. Consequently, the *rainfall* input data for geo-scientific models, which is often considered to be an *observation*, is rather non-exhaustive or a product of yet another modelling or processing step. Methods that interpolate, merge or model datasets can often be considered to be geostatistical, or at least rely upon them (Goovaerts 2000; Jewell and Gaussiat 2015).
I hereby present SciKit-GStat, a Python package that implements the most fundamental processing and analysis step of geostatistics: the variogram estimation. It is open source, object oriented, well documented, flexible and powerful to overcome the limitation other software implementation may have.

The successful journey of geostatistics started in the early 1950s and continuous progress has been made ever since. The earliest work was published 1951 by the South African engineer David Krige (Krige 1951). He also lent his name to the most popular geostatistical interpolation technique *kriging*. Nevertheless, Matheron (1963) is often referenced as the founder of geostatistics. His work introduced the mathematical formalization of the variogram, which opened geostatistics to a wider audience, as it could easily be applied to other fields than Mining.

From this limited use case, geostatistics gained importance and spread annually. A major review work is publish almost every decade, illustrating the continuous progress of the subject. Today, it's a widely accepted field that is used throughout all disciplines in geoscience. P. A. Dowd (1991) reviewed the state of the art works from 1987 to 1991 in the fields of geostatistical simulation, indicator kriging, fuzzy kriging and interval estimation. But also more specific applications such as hydrocarbon reservoirs and hydrology are reviewed. Peter M. Atkinson and Tate (2000) reviewed geostatistical works specifically focused on scale issues. The authors highlight the

main issues and pitfalls when geostatistics are used to upscale or downscale data, especially in remote sensing and GIS. A few years later, Hu and Chugunova (2008) summarized 50 years of progress in geostatistics and compared it to more recent developments in multi-point geostatistics. These methods infer needed multivariate distributions from the data to model covariances. Recently, Sarann Ly et al. (2015) reviewed approaches for spatial interpolation, including geostatistics. This work focuses on the specific application of rainfall interpolation needed for hydrological modelling.
Such works are only a small extract from what has been published during recent years. They are only outnumbered by the many domain-specific studies that focus on improving geostatistical methods for specific applications.

In recent years the field of geostatistics has experienced many extensions. Many processes and their spatial patterns studied in geoscience are not static but dynamically change on different scales. A prime example is soil moisture, which changes on multiple temporal scales exposing spatial patterns that are not necessarily driven by the same processes throughout the year (M. Mälicke et al. 2020; Vanderlinden et al. 2012; Vereecken et al. 2008; Andrew W. Western, Zhou, et al. 2004). The classic Matheronian geostatistics assumes stationarity for the input data. Hence, a temporal perspective was introduced into the variogram, modeling the spatial covariance accompanied by its temporal counterpart (Christakos 2000; De Cesare et al. 2002; Ma 2002; Ma 2005). In parallel, approaches were developed, that questioned and extended the use of Euclidean distances to describe proximity between observation locations (J. B. Boisvert et al. 2009; Jeff B. Boisvert and Clayton V. Deutsch 2011; Curriero 2005). Last but not least, efforts are made to overcome the fundamental assumption of Gaussian dependence, that underlies the variogram function. This can be achieved for example by sub-Gaussian models (Guadagnini et al. 2018) or copulas (Bárdossy 2006; Bárdossy and Li 2008). Non-Gaussian geostatistics are, however, not covered in SciKit-GStat.

The variogram is the most fundamental means of geostatistics and a prerequisite to apply other methods, such as interpolation. It relates the similarity of observations to their separating distance using a spatial model function. This function, bearing information about the spatial covariance in the dataset, is used to derive weights for interpolating at unobserved locations. Thus, any uncertainty or error made during variogram estimation, will be be propagated into the final result. As described, geoscientific datasets are often sparse in space and that makes it especially complex to choose the correct estimator for similarity and decide when two points are considered *close* in space. Minor changes to spatial binning and aggregations can have a huge impact on the final result, as will be shown in this work. This is

an important step that should not entirely be left to the computer. To foster the understanding and estimation of the variogram, SciKit-GStat is equipped with many different semi-variance estimators (table A.1) and spatial models (table A.2), where other implementations only have one or two options if any at all. Spatial binning, can be carried out utilizing one of ten different algorithms to break up the tight corset that geostatistics usually employs for this crucial step. Finally, SciKit-GStat implements various fitting procedures, each one in weighted and unweighted variation, with many options to automate the calculation of fitting weights. Additionally, even a utility suite is implemented, that can build a maximum likelihood function at runtime for any represented variogram to fit a model without binning the data at all (Lark 2000). Appendix A.9.3 briefly summarizes the tutorial about maximum likelihood fitting. These tools enable a flexible and intuitive variogram estimation. Only then, is the user able to make an informed decision, whether a geostatistical approach is even the correct procedure for a given dataset at all. Otherwise, Kriging would interpolate based on a spatial correlation model, which is in reality not backed-up with data.

De-facto standard libraries for geostatistics can be found in a number of commonly used programming languages. In FORTRAN, there is `gslib` (Clayton V Deutsch and André G Journel 1998), a comprehensive toolbox for geostatistical analysis and interpolation. Spatio-temporal extensions to gslib are also available (De Cesare et al. 2002). For the R programming language, the `gstat` package (Gräler et al. 2016; E. J. Pebesma 2004) can be considered the most complete package, covering most fields of applied geostatistics.
For the Python programming language, there was no package comparable to `gstat` in 2016. A multitude of Python packages, that were related to geostatistics could be found. A popular geostatistics related Python package is `pykrige` (Murphy et al. 2021). As the name already implies, it is mainly intended for kriging interpolation. The most popular kriging procedures are implemented, however, only limited variogram analysis is possible. HPGL is an alternative package offering very comparable functionality. Unlike pykrige, the library is written in C++, which in wrapped and operated through Python. The authors claim to provide a substantially faster implementation than gslib (which is written in FORTRAN). Another geostatistical Python library that can be found is *pygeostat*. It mainly focuses on geostatistical modeling. Unfortunately, obtaining the files and then installing it in a clean Python environment turned out to be cumbersome[9].
All of the reviewed packages focus only on a specific part of geostatistics and in general, interfacing options were missing. Thus, I decided to develop an open source

**9** At the time, several undocumented issues raised and solving them was not straightforward

geostatistics package for the Python programming language called **SciKit-GStat**. In the course of the following years, another Python package with similar objectives was developed called `gstools` (Müller et al. 2021b). Both packages emerged at similar times; SciKit-GStat was first published on Github in July 2017, `gstools` in January 2018. With streamlining developments between these two packages, the objective of SciKit-GStat shifted and is today mainly focused on variogram estimation. Today, both packages work very well together and the developers of both packages collaborate to discuss and streamline future developments. Further details driving this decision are stated throughout this work, especially in section A.3.2, A.5.2 and A.6.3. One of the goals of this work is to present differences between SciKit-GStat and existing other packages and illustrate, how it can be interfaced and connected to them. This will foster the development of a unique geostatistical working environment that can satisfy any requirement in Python.

A number of works were especially influential during the development of SciKit-GStat. An early work by Burgess and R. Webster (1980) published a clear language description, of what a variogram is and how it can be utilized to interpolate soil properties to unknown locations. In the same year Cressie and Hawkins (1980) published an alternative variogram estimator to the Matheron estimator introduced 20 years earlier. This estimator is an important development, as its contained power transformation makes it more robust to outliers, that we often face in geoscience. A noticeable amount of functions implemented in SciKit-GStat are directly based on equations provided in Bárdossy and Lehmann (1998). This work does not only provide a lot of statistical background to the applied methods, but also compares different approaches for kriging. Finally, a practical guide to implement geostatistical applications was published by Montero et al. (2015). A number of model equations implemented in SciKit-GStat are directly taken from this publication.

SciKit-GStat is a toolbox that fits well into the SciPy environment. For scientific computing in Python, SciPy (Virtanen et al. 2020) has developed to be the de facto standard environment. Hence, using available data structures, such as the numpy array (van der Walt et al. 2011), as an input and output format for SciKit-GStat functions makes it very easy to integrate the package into existing environments and workflows. Additionally, SciKit-GStat uses SciPy implementations for mathematical algorithms or procedures wherever available and feasible. I.e., the SciPy least squares implementation is used to fit a variogram model to observed data. Using this common and well-tested implementation of least squares makes SciKit-GStat less error prone and fosters comparability to other scientific solutions also based on SciPy functionality.

SciKit-GStat enables the user to estimate standard, but also more exotic variograms. This process is aided by a multitude of helpful plotting functions and statistical output. In other geostatistical software solutions, the estimation of a variogram is often left entirely to the computer. Some kind of evaluation criterion or objective function takes the responsibility of assessing the variograms suitability for expressing the spatial structure of the given input data in a model function. Once used in other geostatistical applications, such as kriging, the theoretical model does not bear any information about its suitability or even goodness of fit to the actual experimental data used. Further advanced geostatistical applications do present a variogram to the user, while performing other geostatistical tasks, but this often seems as a passive information that the user may recognize or ignore. The focus is on the application itself. This can be fatal as the variogram might actually not represent the statistical properties well enough. One must remember that the variogram is the foundation of any geostatistical method and unnoticed errors within the variogram will have an impact on the results even if the maps look viable. The variogram itself is a crucial tool for the educated user to interpret whether data interpolation using geostatistics is valid at all.

SciKit-GStat takes a fundamentally different approach here. The variogram itself is the main result. The user may use a variogram and pass it to a kriging algorithm, or use one of the interfaces to other libraries. However, SciKit-GStat makes this a manual step by design. The user is put from a passive into an active role, and is therefore close to geostatistical textbooks, which usually present the variogram first.

SciKit-GStat is also designed for educational applications. Both students and instructors are specifically targeted within SciKit-GStat's documentation and user guide. While some limited knowledge of the Python programming language is assumed, the user guide starts from zero in terms of geostatistics. Beside a technical description of the SciKit-GStat classes, the user is guided through the implementation of the most important functionality. This fosters a deeper understanding of the underlying methodology for the user. By using SciKit-GStat documentation, a novice user does not only learn how to use the code, but also what it does. This should be considered a crucial feature for scientific applications, especially in geostatistics where a multitude of one-click software is available, producing questionable results if used by uneducated users.

SciKit-GStat is well documented and tested. The current unit test coverage is >90%. The online documentation includes an installation guide, the code reference and an user guide. Additionally, tutorials are available, that are suitable for use in higher education level lectures. To facilitate an easy usage of the tutorials a Docker

image is available (and the Dockerfile is part of SciKit-GStat). SciKit-GStat has a growing developer community on Github and is available under a MIT license.

The following section will give a more detailed overview of SciKit-GStat. Section A.4 introduces the fundamental theory behind geostatistics as covered by SciKit-GStat. Section A.5 guides through the specific implementation of the theory, section A.6 gives details on user support and contribution guidelines.

## A.3 SciKit-GStat general overview

The source code repository contains the Python package itself, the documentation and sample data. This work will focus mainly on the Python package, starting with a detailed overview in section A.3.2. The documentation is introduced to some detail in section A.3.2 and section A.6. Most data distributed with the source code is either artificially created for a specific chapter in the documentation, or originally published somewhere else. In these cases either the reference or license is distributed along with the data itself. For this publication, all figures were created with the same data, wherever suitable. This is further introduced in section A.3.1 and appendix A.9.2.

### A.3.1 Data

There are already some benchmark datasets for geostatistics, such as the meuse dataset distributed with the R package `sp` (Bivand et al. 2008; E. Pebesma and Bivand 2005), which is also included in SciKit-GStat. In order to provide a dataset of a random field (not only a sample thereof), which has obvious spatial covariance structure, an image of a pancake was utilised (figure A.1). This approach was employed to enable the implementation of custom sampling strategies and the ability to analyse the dataset at any level of sampling density within such an image. Furthermore, with a pancake, one does not focus too much on location specifics or properties of the random field, as it will happen with i.e. a remote sensing soil moisture product from an actual location on earth. The pancake browning (figure A.1) shows a clear spatial correlation, the field is exhaustive at the resolution of the camera device and creating new realizations of the field is possible as well. Processes forming spatial structure in browning might be different from processes dictating the spatial structure of i.e. soil moisture, but they are ultimately also driven by physical principles. Testing SciKit-GStat tools not only with classic geoscientific data, but also with pancakes made the implementation more robust. But it also illustrates that the geostatistical approach holds beyond geoscience. A technical

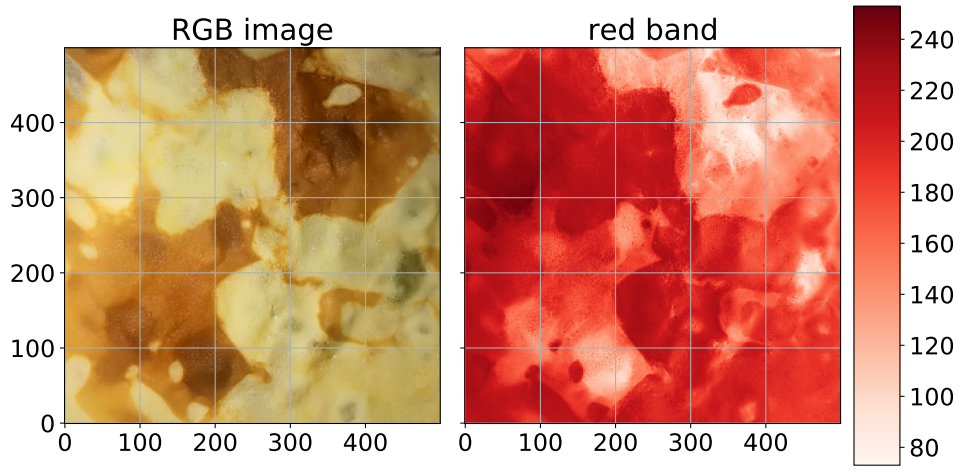



**Figure A.1:** Original photograph of the pancake used to generate the pancake dataset. The white points indicate the 500 sampling locations that were chosen randomly, without repeating. The observation value is the red channel value of the RGB value of the specified pixel.

description of how to cook your own dataset is given in appendix A.9.2. The meuse dataset is used in the tutorials of SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a). Appendix A.9.1 summarizes the dataset and the tutorial briefly and can be used to compare this to the pancake results presented.

Neither the pancake nor the meuse dataset provide space-time data. To demonstrate the support of space-time variogram within SciKit-GStat, another dataset of distributed soil temperature measurements is utilized and distributed with the software. The data is part of a dense network of cosmic-ray neutron sensors (B. Fersch et al. 2020), located in the Rott headwater catchment at the TERENO Pre-Alpine Observatory (Kiese et al. 2018) in Fendt, Germany. The distributed soil temperature measurements consist of *"55 vertical profiles [...] covering an area of about 9 ha [...] record[ing] permittivity and temperature at 5, 20 and 50 cm depth, every 15 min."* (B. Fersch et al. 2020, section 3.8.1, p.2298). In order to decrease the computing demands, I only used temperature measurements at 20 cm and used only every 6th measurement[10].

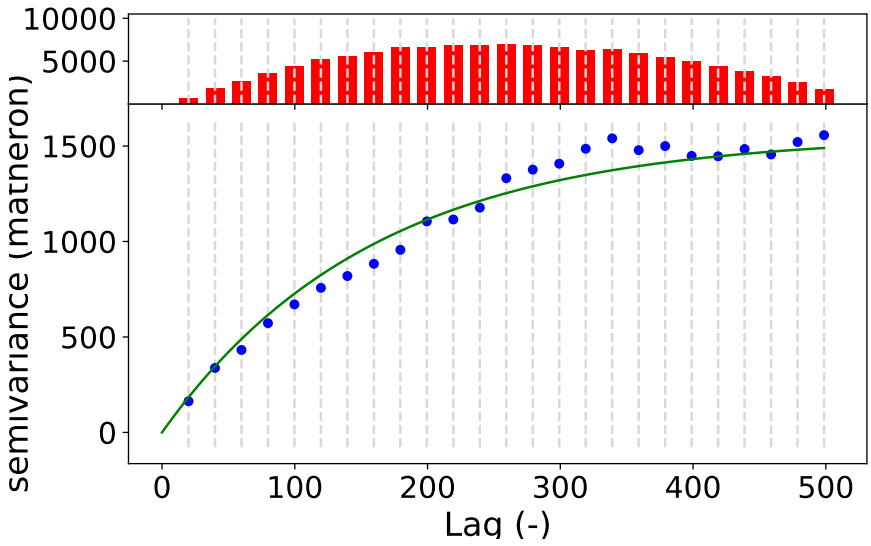



**Figure A.2:** Default variogram plot of SciKit-GStat using the matplotlib backend. The variogram was estimated with the pancake dataset using the exponential model (green line) fitted to a experimental variogram (blue dots) resolved to 25 evenly spaced lag classes, up to 500 units (the axe length of the sampled field). The histogram in the upper subplot shows the amount of point pairs for each lag class. The histogram shares the x-axis with the variogram, to identify the corresponding lag classes with ease.

## A.3.2 Package description

SciKit-GStat is a library for geostatistical analysis written in the Python programming language. The Python interpreter must be of version 3.6 or later. The source files can be downloaded and installed from the Python package index using pip, which is the standard tool for Python 3[11]. All dependencies are installed along with the source files. This is the standard and recommended procedure for installing and updating packages in Python 3. Additionally, the source code is open and available on Github and can be downloaded and installed from source. SciKit-GStat is published under a MIT license.

**10** Note that I only used a measurement every 1.5 hours and did not aggregate the time series.

**11** SciKit-GStat is also available on Conda Forge, the largest community driven Anaconda channel. This package is not covered here, as the content is the same and installation requires the presence of an Anaconda environment and some knowledge of the system. Nevertheless, Anaconda is widely spread among scientists and it might be worth mentioning the existence for anaconda users.

The presented module is built upon common third party packages for scientific computing in Python, called `scipy`. In recent years, the SciPy ecosystem has become the de facto standard for scientific computing and applications in Python. SciKit-GStat makes extensive use especially of `numpy` (Oliphant 2006; van der Walt et al. 2011) to build data structures and numerical computations, `matplotlib` (Hunter 2007a) and `plotly` (Inc. 2015) for plotting and the `scipy` library itself (Virtanen et al. 2020) for solving some specific mathematical problems, such as least squares or matrix operations.

An object oriented programming approach was chosen for the entire library. SciKit-GStat is designed to interact with the user through a set of classes. Each step in a geostatistical analysis workflow is represented by a class and its methods. Argument names passed to an instance on creation are chosen to be as close as possible to existing and common parameter names from geostatistical literature. The aim is to make the usage of SciKit-GStat as intuitive as possible for geoscientists with only little or no experience in Python.

The main focus of the package is variogram analysis. Ordinary kriging is also implemented into SciKit-GStat, but the main strength is variogram analysis. Kriging is available as a valuable tool to cross-validate the variogram by interpolating the observation values. For flexible, feature rich and fast kriging applications, the variogram can be exported to other libraries with ease. SciKit-GStat offers an extensible and flexible class that implements common settings out of the box but can be adjusted to rather uncommon problems with ease. An example variogram is shown in figure A.2. By default, the user has an experimental variogram, a well fitted theoretical model and a histogram to estimate the point pair distribution in the lag classes at ones disposal. This way, the plot of the variogram instance helps the user at first sight to not only estimate goodness of fit, but also the spatial representativity of the variogram for the sample used. All parameters can be changed in place and the plot can be updated, without restarting Python or creating new unnecessary variables and instances.

**Table A.1:** Overview of all semi-variance estimator functions implemented in SciKit-GStat. Using *Normalized Range* and *Percentile* is only advised to users understanding the implications as explained in section A.5.1.

| Estimator | Identifier | Description | Reference |
|---|---|---|---|
| Mathéron | `'matheron'` | Default, most popular estimator | Matheron (1963) |
| Cressie-Hawkins | `'cressie'` | Power transformation based - robust to outliers | Cressie and Hawkins (1980) |
| Dowd | `'dowd'` | Median based, fast estimator for non-normal distributed residuals | Dowd (1984) |
| Genton | `'genton'` | Percentile-based estimator - powerful for skewed residuals, but very computationally intensive | Genton (1998) |
| Shannon Entropy | `'entropy'` | Information theoretic measure focusing information content of residuals | Shannon (1948) |
| Normalized Range | `'minmax'` | Experimental estimator using only the spread of residuals | |
| Percentile | `'percentile'` | Uses any user-defined percentile as semi-variance, but untransformed. Experimental | |

**Table A.2:** Overview of all theoretical variogram model functions implemented in SciKit-GStat.

| Model | Identifier | Description | Implementation |
|---|---|---|---|
| Spherical | `'spherical'` | Short ranged correlation length, popular model in geoscience; for smooth, but steep gradients in fields. | Burgess and R. Webster (1980) |
| Exponential | `'exponential'` | Long ranged for smooth fields with less steep gradients. | A G Journel and Huijbregts (1976) |
| Gaussian | `'gaussian'` | Mid ranged for sharply changing fields | A G Journel and Huijbregts (1976) |
| Cubic | `'cubic'` | Similar to Gaussian models, but with a shorter correlation length. | Montero et al. (2015) |
| Matérn | `'matern'` | Has an additional smoothness parameter to adapt shapes between Exponential and Gaussian models. | Zimmermann et al. (2008) |
| Stable | `'stable'` | Has an additional shape (power) parameter to adapt the range. | Montero et al. (2015) |
| Isotonic Regression | `'harmonize'` | Data harmonization algorithm to directly monotonize the experimental variogram, without fitting | Pedregosa et al. (2011a) |

**Table A.3:** Overview of all lag class binning methods implemented in SciKit-GStat.

| Function | Identifier | Description | Implementation |
|---|---|---|---|
| Equidistant lags | `'even'` | *N* lags of same width; Almost always used. | Mirko Mälicke, Möller, et al. (2021a) |
| Uniform lags | `'uniform'` | *N* lags of same sample size; Estimats are based on the same sample size & no empty bins | Mirko Mälicke, Möller, et al. (2021a) |
| Sturge's rule | `'sturges'` | Equidistant lags derived from Sturge's rule; use for small normal distributed distance matrices | Virtanen et al. (2020) |
| Scott's rule | `'scott` | Equidistant lags derived from Scott's rule; use for large datasets | Virtanen et al. (2020) |
| Freedman-Diaconis estimator | `'fd'` | Equidistant lags; use for small datasets with outliers in the distance matrix | Virtanen et al. (2020) |
| Square-root | `'sqrt'` | Equidistant lags; Very fast function, but usually not recommended | Virtanen et al. (2020) |
| Doane's rule | `'doane'` | Equidistant lags; based on data skewness, use for small non-normal distance matrices | Virtanen et al. (2020) |
| K-Means | `'kmeans'` | Non-equidistant lags; clustered distance matrix is used as binning; slow but statistically robust | Pedregosa et al. (2011a) |
| Hierachical Clusters | `'ward'` | Non-equidistant lags; clustered distance matrix is used as binning; Based on Ward's criterion for minimizing cluster variance. Computational intensive | Pedregosa et al. (2011a) |
| Stable Entropy | `'stable_entropy'` | Non-equidistant lags; Bin edges are set by minimizing the deviations of per-lag Shannon entropy | Mirko Mälicke, Möller, et al. (2021a) |

SciKit-GStat contains eight different semi-variance estimators (overview in table A.1) and seven different theoretical variogram model functions (overview in table A.2). At the same time implementing custom models and estimators is supported by a decorator function that only requires the mathematical calculation from the user, which can be formulated with almost no prior Python knowledge. Often with a single line of code.

SciKit-GStat offers a multitude of customization options to fit variogram models to experimental data. The model parameters can be fitted manually or by one of three available optimization algorithms: Levenberg-Marquardt, Trust Region Reflective and parameter Maximum Likelihood (see section A.5.1). It is also possible to combine both. Furthermore, it is possible to weight experimental data. Such weighting of experimental data is a crucial feature to make a variogram model fit data at short lags more precisely than distant observations. The user can manually adjust weights or use one of the many predefined functions, that define weights i.e. dependent on the separating distance. Closely related is the way how SciKit-GStat handles spatial aggregation. The user can specify a function that will be used to calculate an empirical distribution of separating distance classes, which are the foundation for spatial aggregation. Especially for sparse datasets which base their aggregation on small sample sizes, even adding or removing a single lag class can dramatically change the experimental variogram. The default function defines equidistant distance lag classes, as mostly used in literature. However, SciKit-GStat also includes functionality for auto-deriving a suitable number of lag classes or cluster based methods, which have to my knowledge, not been used so far in this context. A complete overview of all functions is given in table A.3.

Interfaces to a number of other geostatistical packages are provided. SciKit-GStat defines either an export method or a conversion function to transform objects that can be read by other packages. Namely, the `Variogram` can export an parameterized custom variogram function, which can be read by kriging classes of the `pykrige` package. A similar export function can transform a variogram to a covariance model as used by `gstools`. This package is evolving to be the prime geostatistical toolbox in Python. Thus, a powerful interface is of crucial importance. Finally, a wrapping class for `Variogram` is provided that will make it accessible as a scikit-learn (Pedregosa et al. 2011a) estimator object. This way, scikit-learn can be used to perform parameter search and use variograms in a machine learning context.

SciKit-GStat is easily extensible. Many parts of SciKit-GStat were designed to keep the main algorithmic functions clean. Overhead, like type checks and function mapping to arrays are outsourced to instance methods wherever possible. This

enables the user to implement custom functions with ease, even if they are not too familiar with Python. As an example, implementing a new theoretical model is narrowed down to only implementing the mathematical formula this way.

Documentation provided with SciKit-Gstat are tailored for educational use. The documentation mainly contains a user guide, tutorials and a technical reference. The user guide for SciKit-GStat does not have any prerequisites in geostatistics and guides the reader through the underlying theory, while walking through the implementation. For users with some experience in Python, geostatistics and other fields of statistics, tutorials are provided. The tutorials focus on a specific aspect of SciKit-GStat and demonstrate the application of the package. Here, a sound understanding of geostatistics is assumed. Finally, the technical reference does only document the implemented functions and classes from a technical point of view. It is mainly designed for experienced users that need an in-depth understanding of the implementation or for contributors that want to extend SciKit-GStat.

SciKit-GStat is 100% reproducible through docker images. With only the docker software installed (or any other software that can run docker containers), it is possible to run the scikit-gstat docker image, which includes all dependencies and common development tools used in scientific programming. This makes it possible to follow the documentation and tutorials instantly. The user can use a specific SciKit-GStat version (from 1.0 on) and conduct analysis within the container. That will fix all used software versions and, if saved, make the analysis 100% reproducible. At the same time the installation inside docker container does not affect any existing Python environment on the host system and is therefore perfect to test SciKit-GStat.

SciKit-GStat is recognized on Github and has a considerable community. Issues and help requests are submitted frequently and are usually answered in a short amount of time by the author. At the same time, efforts are made to establish a broader developer community, to foster support and development. Additionally, the development on SciKit-GStat is closely coordinated with `gstools` and the parenting Geostat-Framework developer community.

## A.4 Main geostatistical components

### A.4.1 Variogram

In geostatistical literature, the terms *semi-variogram* and *variogram* are often mixed or interchanged. Although closely related, two different methods are described by these terms. In most cases, the *semi-variogram* is used, but called simply *var-*

*iogram*. Here, I follow this common nomenclature and both terms describe the semi-variogram in the following.

At its core, the semi-variogram is a means to express how spatial dependence in observations changes with separating distance. An observation is here defined to be a sample of a spatial random function. While these functions are usually two or three dimensional in geostatistical applications, they can be N-dimensional in SciKit-GStat (including 1D). A more comprehensive and detailed introduction to random functions in the context of geostatistics is given in Montero et al. (2015, chapter 2.2, p. 11 ff.). The most fundamental assumption that underlies a variogram, is therefore, that proximity in space leads to similar observations (proximity in value). To calculate spatially aggregated statistics on the sample, the variogram must make an assumption up to which distance two observations are still close in space. This is carried out by using a distance lag over the exact distances, as two point pairs will hardly be at exactly the same spatial distance in real world datasets.

Separating distance is calculated for observation point pairs. For different distance lag classes (e.g. 10 m to 20 m), all point pairs $s_i, s_j$ within this class are aggregated to one value of (dis)similarity, called semi-variance $\gamma$. A multitude of different estimators are defined to calculate the semi-variance. For a specific lag distance $h$ (e.g. 10 m), the most commonly used Matheron estimator (Matheron 1963) is defined by equation (A.1):

$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} (Z(s_i) - Z(s_{i+h})) \tag{A.1}$$

Where $N(h)$ is the number of point pairs for the lag $h$ and $Z(s)$ the observed value at the respective location $s$. The obtained functions is called an *experimental variogram* in SciKit-GStat. In literature, the term *empirical variogram* is also quite often used and is referring more or less, to the same thing. In SciKit-GStat, the empirical variogram is the combination of the lag classes and the experimental variogram. All estimators implemented in SciKit-GStat are described in detail in section A.5.1.

To model spatial dependencies in a data set, a formalized mathematical model has to be fitted to the experimental variogram. This step is necessary, to obtain parameters from the model, in a formalized manner. These describe spatial statistical properties of the model, which may (hopefully) be generalized to the random field. These parameters are called variogram parameters and include:

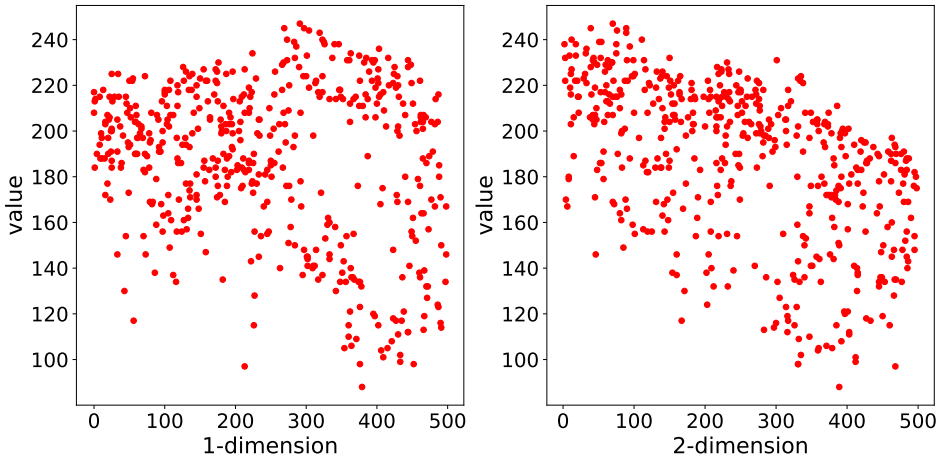



**Figure A.3:** Scatter plot of the observation values in the pancake dataset related to only one coordinate dimension. As the pancake dataset was 2D, *1-dimension* corresponds to the x coordinate and *2-dimension* to the y-coordinate of the pancake sample. This plotting procedure can help the user to identify a dependence on the location for the data sample, which can violate the second order stationarity.

1. *nugget* - the semi-variance at lag $h = 0$. This is the variance, that cannot be explained by a spatial model and is inherit to the observation context. (i.e. measurement uncertainties or small-scale variability).

2. *sill* - the upper limit for a spatial model function. The nugget and sill add up to the sample variance.

3. *effective range* - the distance, at which the model reaches 95% of the sill. For distances larger than the range, the observations become statistically independent. Variogram model equations also define a **model** parameter called range, which leads to misunderstandings in the geostatistical community. To overcome these problems, SciKit-GStat formulated all implemented models based on the *effective range* of the variogram and not the range model parameter. Consequently, the given formulas might differ from some common sources by the transformation of effective range to range model parameter. This transformations are straightforward and reported in literature, but for some models (i.e. Gaussian) not commonly the same. In these cases, the user is encouraged to carefully check the implementation used in SciKit-GStat.

Closely related to these parameters is the nugget to sill ratio. It is interpreted

as the share of spatially explainable variance in the sample and is therefore a very important metric to reject the usage of a specific variogram model at all.

The theoretical model is a prerequisite for spatial interpolation. For this to happen, a number of geostatistical assumptions need to be fulfilled. Namely, the observations have to be of second-order stationarity and the intrinsic hypothesis has to hold. This can be summarized as the requirement, that the expected value of the random function and its residuals must not dependent on the location of observation, but solely on the distance to other points. This assumption has to hold for the full observation space. Hence, the semi-variance is calculated with the distance lag $h$ as the only input parameter. A more detailed descriptions of these requirements is e.g. given in Montero et al. (2015, chapter 3.4.1 p. 27 ff.) or Bárdossy and Lehmann (1998) and Burgess and R. Webster (1980). An important tool to learn about trends in the input dataset is a scatter plot like shown in figure A.3. The same variogram instance that was used for figure A.2 is used here. The two subplots show the observation values related to only one dimension of their coordinates. This scatter plot can help the user to identify a dependence of observations on the location, which could violate the assumptions named above. The pancake sample observations are independent of the x-axis coordinates (1. dimension). For the second dimension there might be a slight dependence of large observation values on the y coordinate. This readily available plot is useful to guide the user into the decision of utilizing statistical trend tests to test for statistical significance and finally detrending input data.

The other requirement for variogram models is that it has to be monotonically increasing. A drop in semi-variance would imply that observations become more similar with increasing distance, which is incompatible to the most fundamental assumption in geostatistics of spatial proximity. This requirement can only be met by a statistical model function and not the experimental variogram, which is often not monotonically increasing in a strict sense. This may happen due to the fact that (spatial) observations are not exhaustive and measurements might be uncertain.

### A.4.2 Kriging

One of the most commonly used applications of geostatistics is kriging. A sample result is shown in figure A.4. The interpolation was made with the same variogram instance used to produce figures A.2 and A.3. The center sub-figure shows the result itself, along with the original field (left) and a kriging error map (right), which will be introduced later. In this example, the spatial properties and correlation lengths of the original are well captured by the result.

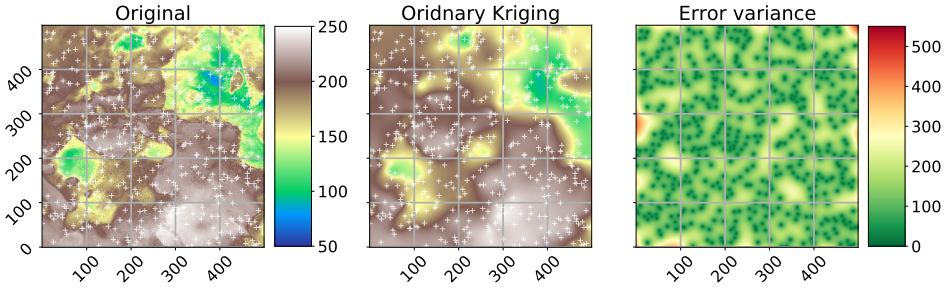



**Figure A.4:** Ordinary Kriging result of the pancake dataset sample used in figure A.2 and A.3. The kriging was performed with default parameters on a grid of same resolution as the original field. The original image and ordinary kriging result share the value space and thus, the colorbar between the two subplots is valid for both. The original is the same as in fig. A.1, but using a different color-scale to make differences more pronounced. The white crosses indicate the sample positions. The third subplot (right panel) indicated the associated kriging error variance as returned by the algorithm.

Kriging estimates the value for an unobserved location $s_0$ as the weighted sum of nearby observations as shown in equation (A.2).

$$Z^*(s_0) = \sum_{i=1}^{N} \lambda_i Z(s_i) \tag{A.2}$$

Where $Z^*(s_0)$ is the estimation and $\lambda_i$ are the weights for the $N$ neighbors $s_i$. The kriging procedure uses the theoretical variogram model fitted to the data to derive the weights from the spatial covariance structure. Furthermore, by requiring all weights to sum up to one (equation (A.3)) the unbiasedness of the prediction is assured.

$$\sum_{i=1}^{N} \lambda_i = 1 \tag{A.3}$$

A single weight can thereby be larger than one or smaller than 0. As the weights are inferred from the spatial configuration of the neighbors, this can require stronger influence ($\lambda > 1$) or even negative influence ($\lambda < 0$) of specific observations. Combined with unbiasedness, this is one of the most important features of a kriging interpolation and can make it superior to, i.e. spline-based procedures in an environmental context. Deriving weights from the spatial properties of the data is especially helpful, as the local extreme values have likely not been observed, but their influence is present in the spatial covariance of the field close to it.

To obtain the weights for one unobserved location, a system of equations is formulated, called the kriging equation system (KES). By expecting the prediction errors to be zero (equation (A.4)) and substituting equation (A.2) in equation (A.4), the KES can be formulated.

$$E[^*(s_0) - Z(s_0)] = 0 \tag{A.4}$$

The final kriging equation (A.5) is taken from Montero et al. (2015, equation 4.16, p. 86) and its derivation is given in chapter 4.3.1 of the same source (Montero et al. 2015, p.84-90).

$$\begin{cases} \sum_{j=1}^{N} \lambda_j \gamma(s_i - s_j) + \alpha = \gamma(s_i - s_0), \quad i = 1, \ldots, N \\ \sum_{i=1}^{N} \lambda_i = 1 \end{cases} \tag{A.5}$$

Where $\alpha$ is the Lagrange multiplier needed to solve the KES by minimizing the estimation variance subject to the constraint of equation (A.3). By minimizing the prediction variance and requiring the weights to sum to one, it is possible to obtain the best linear, unbiased estimation. Thus, kriging is often referred to as being a BLUE (Best Linear Unbiased Estimator). Using Kriging an estimate of the variance of the spatial prediction can be obtained. This is shown in the right panel of figure A.4. Such information is vital to assess the quality of the prediction. Finally, the setup of Kriging makes it a smooth interpolation, as the predictions very close to observation locations are approaching the observation values smoothly. The kriging variance is significantly higher in less densely sampled regions (figure A.4), which enables the user to visually assess the spatial representativity of the obtained results.

### A.4.3 Directional variogram

The standard variogram as described in section A.4.1 handles *isotropic* samples. That means the spatial correlation length of the random field is assumed to be of comparable length in each direction. Usually, one refers only to the directions along the main coordinate axes. However, direction can be defined with any azimuth angle and does not have to match the coordinate axes. If the spatial correlation length differs in direction this is referred to as *anisotropy*. There are two different kinds of anisotropy: geometric and zonal anisotropy (Wackernagel 1998). Considering geometric anisotropy, the effective range differs for the two perpendicular main directions of the anisotropy. In the zonal case, sill and range differ. Geometric anisotropy can be handled by a coordinate transformation (Wackernagel 1998).

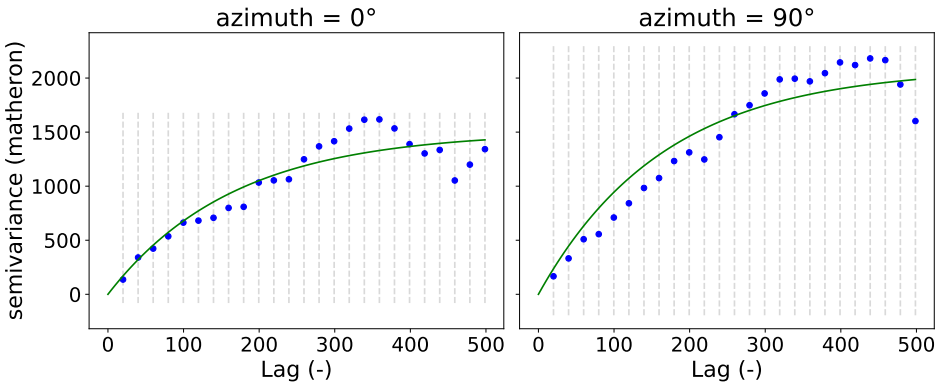



**Figure A.5:** Two directional variograms calculated for the pancake dataset. Both variograms use the same parameters as the instance used to produce figure A.2. In addition, the direction is taken into account. The two variograms shown differ only in the azimuth used, which is 0 ° (left) and 90 ° (right).

These cases can be detected by directional variograms. For an application, the main directions of anisotropy must be identified to then estimate an isolated variogram for each direction.

For each directional variogram, only point pairs are considered that are oriented in the direction of the variogram. For two observation locations $s_1$, $s_2$ the orientation is defined as the angle between the vector $\vec{u}$ connecting $s_1$ and $s_2$ and a vector along the first dimension axis: $\vec{e} = [1, 0]$. The cosine of the orientation angle $\Theta$ can be calculated using equation (A.6):

$$cos(\Theta) = \frac{\vec{u} \circ \vec{e}}{|\vec{e}| \cdot |(1,0)|} \tag{A.6}$$

The directional variogram finally defines an azimuth angle, defined analogous to equation (A.6) and a tolerance. Any point pair which deviates less than tolerance from the azimuth, is considered to be oriented in the direction of the variogram and will be used for estimation.

The example data used so far shows a small anisotropy (figure A.5). The two variograms used exactly the same data and parameters as used for figure A.2. The only difference is that both are directional and they use two different directions of 0 ° and 90 °. There is a difference in effective range and sill in the 90 ° directional variogram.

As long as more than one directional variogram is estimated for a data sample, the

difference of the estimated variogram parameters describes the degree of anisotropy. In a kriging application, the data sample can now be transformed along the main directions at which the directional variograms differ until the directional variograms do not indicate an anisotropy anymore. The common variogram of the transformed data can be used for kriging and the interpolated field is finally transformed back. Transformations are not part of SciKit-GStat. The `scipy` and `numpy` packages offer many approaches to apply transformations. Alternatively, `gstools` implements anisotropy directly and can use it for covariance models and kriging. In these cases the user needs to identify the directions manually and specify them on object creation.

### A.4.4 Space-time variogram

At the turn of the millennium, geostatistics had emerged to a major tool in environmental- and geoscience, the demand for new methods was rising. Datasets collected in nature are usually dynamic in time, which can easily violate the second order stationarity assumptions underlying classic geostatistics. Hence, substantial progress had been made to incorporate temporal dimensions into variograms.

The classic variogram is modelling the semi-variance of a sample in dependence of the separating distance of the underlying point pairs. For a space-time variogram, this dependence is expanded to time-lags. That means the data is not only segmented in terms of spatial proximity, but also temporal proximity. The resulting model will be capable to identify co-variances over space and time at the same time (figure A.6). SciKit-GStat uses a 3D plot by default. The plot can be customized and exclude the fitted model or plot the experimental variogram rather as a surface, than a scatter plot. While figure A.6 might contain both, the experimental and the theoretical variogram, it is also quite overloaded and not always helpful. Finally, a printed 3D plot cannot be rotated, and the usage in publications is discouraged. To overcome these limitations, SciKit-GStat implements 2D contour plots of the experimental variogram in two variations, which differ only in visualization details (figure A.7). The contour plot is the more appropriate means to inspect the covariance field as estimated by the space-time variogram. With the given example, one can see that the auto-correlation (temporal axis) is dominant and except for a few temporal lags (50 - 60, or 30 - 40), the variogram shows almost a pure nugget along the spatial axis. Note that the contour lines smooth out the underlying field to close lines to rings wherever possible. This can lead to the impression that the experimental variogram is homogeneously smooth along the two axis. In fact, this is not the case and the smoothing is due to the implementation of contour lines. Thus, the contour

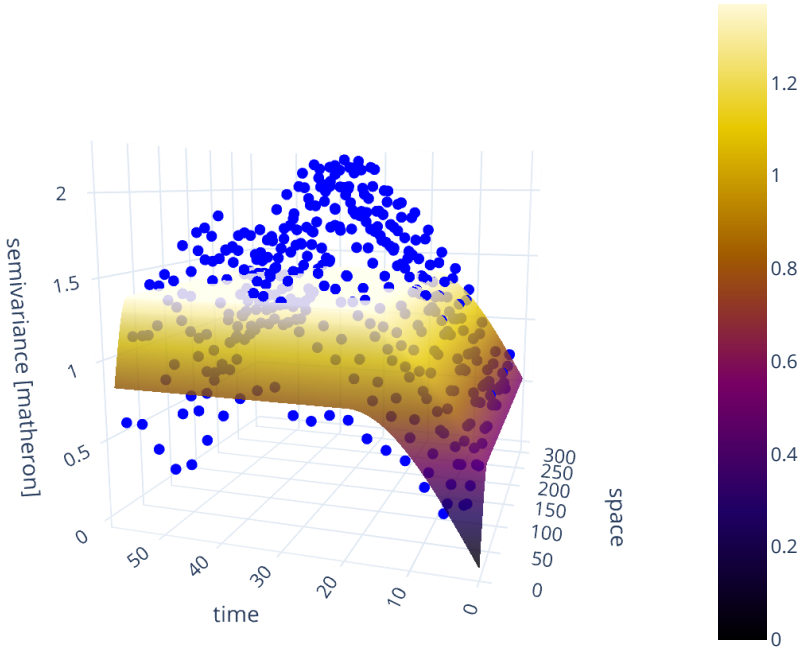



**Figure A.6:** Default 3D scatter plot of a space-time variogram (blue points), with fitted product-sum model (surface). The variogram is estimated from the in-situ soil temperature measurements at 20 cm depth (WSN product) published in B. Fersch et al. (2020). To decrease the computational workload, only every sixth measurement was taken from the timeseries.

plot should be used to get a general idea of the experimental variogram. To inspect the actual semi-variance values, the experimental variogram can be accessed and plotted using a matrix plot.

To build a separable space-time variogram model, the two dimensions are first calculated separately. Non-separable space-time variogram models are not covered in SciKit-GStat. The two experimental variograms are called marginal variograms and relate to the temporal or the spatial dimension exclusively, by setting the other dimension's lag to zero. Finally, these two variograms are combined into a space-

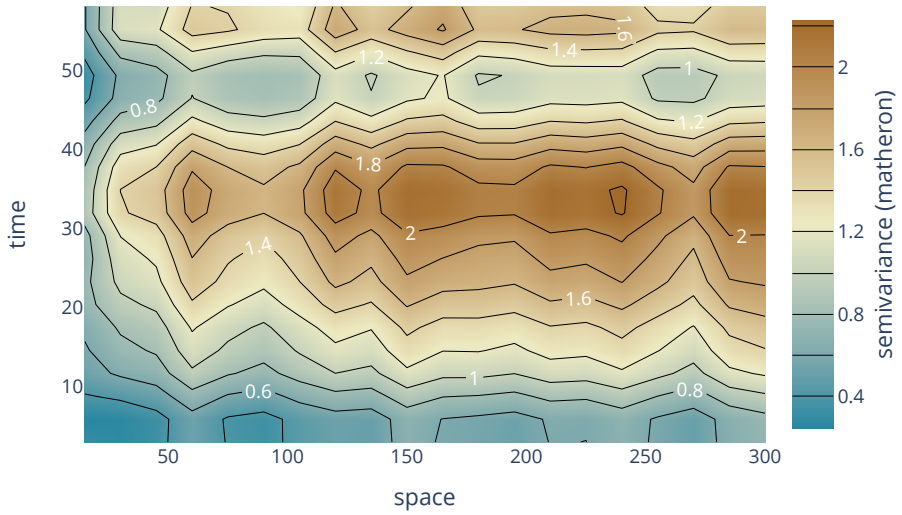



**Figure A.7:** Contour plot of an experimental space-time variogram, without theoretical model. The shown variogram is from exactly the same instance as used for A.6, without any modifications. The contours are calculated for the semi-variances (z-axis) and thus contain the same information as the scatter plot in figure A.6. The color is indicating the magnitude of the semi-variance according to the colorbar.

time variogram model. SciKit-GStat implements three models: the sum model, product model and product-sum model. For each of the marginal experimental variograms, a theoretical model is fitted, as described in section A.4.1. These two models $V_x(h)$ (spatial) and $V_t(t)$ (temporal) are then used to combine their output into the final model's return value $\gamma$. The space-time model defines how this combination is archived.
For the sum model, $\gamma$ is simply $V_x(h) + V_t(t)$. The product and product-sum models are implemented following De Cesare et al. (2002, equation (4), (6)).

## A.5 Software implementation

This section focuses on the implementation of SciKit-GStat. It aims to foster an understanding of the most fundamental design decisions made during development.

Thus, the reader will gain a basic understanding how the package works, where to get started and how SciKit-GStat can be extended or adjusted.

### A.5.1 Main classes

SciKit-GStat is following an object-oriented programming (OOP) paradigm. It exports a number of classes, which can be instantiated by the user. Common geostatistical notions are reflected by class properties and methods to relate the lifetime of each object instance to typical geostatistical analysis workflows. At the core of SciKit-GStat stands the `Variogram` class for variography. Other important classes are:

- `DirectionalVariogram` for direction dependent variography.
- `SpaceTimeVariogram` for space-time variography.
- `OrdinaryKriging` for ordinary kriging interpolations.

#### Variogram

The `Variogram` is the main class of SciKit-GStat and the only construct the user will interact with, in most cases. Each instance of this class represents the full common analysis cycle in variography. That means, each instance will be associated to a specific data sample and holds a fitted model. Other than other libraries, there is no abstraction of variogram models and fitted models are not an entity of their own. If alternate input data (not parameters) is used a new object must be created. This makes the transfer of variogram parameter onto other data samples a conscious action performed by the user and not a side-effect of the implementation. At the same time parameters are mutable and can be changed at any time, which will cause re-calculation of dependent results. While this design decisions makes the usage of SciKit-GStat straightforward, it can also decrease performance. I.e., in SciKit-GStat, a variogram model is always fitted, even if only the experimental variogram is used. This can be a downside, especially on large datasets. For cases where the full variogram instance is not desired or needed, possible pathways are described in section A.5.1 and A.5.1, but the usage of `gstools` might be preferable in these cases.

The second design decision for `Variogram` was interactivity. To take full advantage of OOP, every result, parameter and plot is accessible as an instance attribute, property or method. This always clearly sets ownership and provenance relations for data samples and derived results and properties, as there are no floating results

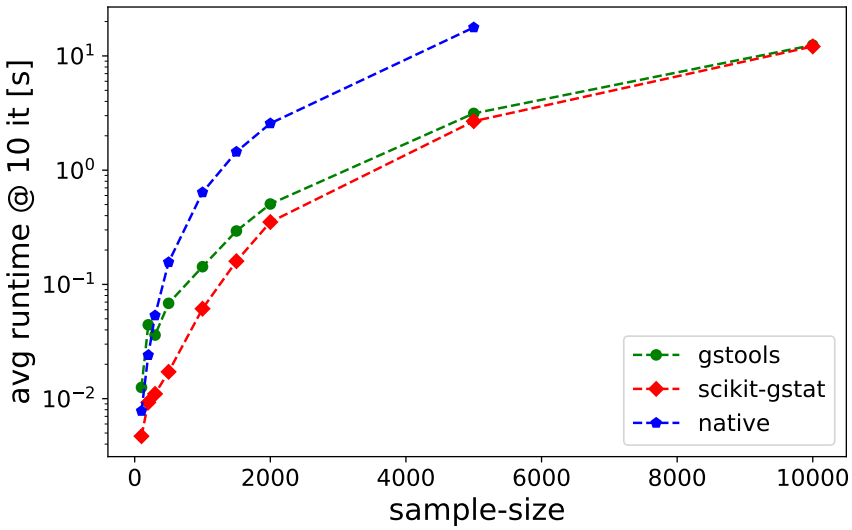



**Figure A.8:** Benchmark test for estimating an experimental variogram. For each sample size, the mean runtime of ten repetitions is shown. The experimental variogram was calcualted with a native Python implementation (blue), `gstools` (green) and SciKit-GStat (red).

that have to be captured in arbitrarily named variables. Moreover, parameters that might be changed during a variogram analysis are implemented in a mutable way. A substantial effort was made to store as few immutable parameters as possible in the instance. Thus, whenever a parameter is changed at run-time, depending derived attributes and results will be updated. This convenient behavior for analysis comes at the cost of performance. This is another major difference to the `gstools` library, in which the author assumes performance to be a driving design decision. To illustrate this as an example: When a variogram instance is constructed without further specifying the spatial model that should be used, it will default to the spherical model. The instance is fitted to this model after construction and can be inspected by the user i.e. by calling a plot method. The user wants to check out another semi-variance estimator, such as the Cressie-Hawkins estimator, because there are a lot of outliers in the dataset. Changing the estimator is as easy as setting the literal estimator name to the estimator property of the variogram. The experimental variogram will instantly be dropped and re-calculated as well as all depending parameters, such as the variogram parameters. The spherical model is fitted a second time now. The user might then realize that a spherical model is not

suitable and can simply change the model attribute, i.e. to the Matérn model. As a direct effect, the variogram parameters are dropped again, as they are once again invalidated, and a new fitting procedure is invoked. This behavior is extremely convenient, as it is easy, interactive, expressive and instant. But it is also slow, as i.e. the theoretical model had been fitted three times, before the user even looked into it. To add some context to *slow* calculations, an experimental variogram estimation run-time test[12] has been performed (figure A.8). One can see, that SciKit-GStat and gstools are very comparable in this case and both significantly faster than a native Python implementation, especially for larger datasets. Note the log-scaled y-axis, indicating differences of magnitudes for larger sample sizes. Interactively adjusting variogram parameters will invoke additional calculations of given run-times.

Although most attributes are mutable, they use common data types in their formulation. This enables the user to intercept the calculation at any point using either primitive language types or numpy data types, which are most accepted by the scientific community as the prime array and matrix data types. Thus, there is no need for the user to learn about custom data-, parameter or result structures using SciKit-GStat.

### distance lag classes

Possibly the most crucial step to estimate a suitable variogram is the binning of separating distances into distance lag classes. In some parts, SciKit-GStat also includes information theoretic methods. Here, to calculate the basic measure, Shannon Entropy (Shannon 1948), the input data has to be binned to calculate empirical non-exceeding probabilities. To distinguish the information theoretic binning from the procedure of binning separating distances into classes, I will refer to the latter as *lag classes*. In the literature, lag classes are commonly referred to as bins, lags, distance lags or distance bins.

SciKit-GStat implements a large number of methods to form lag classes. They can be split into two groups: some are adjusting class edges to fit the requested amount of lag classes. The other group will adjust the number of lag classes to fit other, statistical properties of the resulting lag classes. All methods can be limited by a maximum lag. This is a hyper-parameter, that can be specified by the user, but is not set by default. There are various options for the maximum lag. The user can set the parameter by an absolute value, in coordinate units and larger than one.

**12** This only tests the estimation of the experimental variogram and does not test any other functionality. I.e. kriging implementations in gstools are substantially faster than in SciKit-GStat. The test was not performed in an isolated environment, but repeated several times.

Alternatively, a number between 0 and 1 can be set. Then, the `Variogram` class will set the maximum lag to this share of the maximum pairwise distance found in the distance matrix. I.e. if 0.5 is used, the maximum lag is set to half of the largest point pair distance found. Note, that this is not a median value. Finally, a string can be set as maximum lag. This can either request the arithmetic mean or the median value of the distance matrix as the maximum lag. Typical values from geostatistical textbooks are the median or 60% of the maximum lag (value of 0.6 in SciKit-GStat).

The default behavior is to form a given amount of equidistant lag classes, from 0, to the maximum lag distance. This procedure is used in the literature in almost all cases (with different max-lags), and is thus, a reasonable default method.
Another procedure, takes the number of lag classes and will form lag classes of uniform size. That means, each lag class will contain the same amount of point pairs and, thus be of varying width. This procedure can be explicitly useful to avoid empty lag classes, which can easily happen for equidistant lag classes. Another advantage is that the calculation of semi-variance values will always be based on the same sample size, which makes the values statistically more comparable. These advantages come at the cost of less comparable lag classes. Care must be applied when interpreting lag-related variogram properties such as the effective range. There might be lag ranges that are supported by only a very small amount of actual lag classes.

The next group of procedures use common methods from histogram estimation to calculate a suitable amount of lag classes. This is carried out, either directly, or by estimating the lag class width and deriving the amount of classes needed from this.
The first option is to apply Sturge's rule (Scott 2009) as shown in equation (A.7):

$$n = log_2(s + 1) \tag{A.7}$$

Where $s$ is the sample size and $n$ is the number of lag classes. This rule works good for small, normally distributed distance matrices, but often yields too small $n$ for large datasets.
Similar to Sturge's rule, the square-root rule estimates the number of lag classes as given in equation (A.8):

$$n = \sqrt{(s)} \tag{A.8}$$

This rule is not recommended in most cases. It comes with similar limitation as Sturge's rule but in contrast, it usually yields too large $n$ for large $s$. The main advantage of this rule is that it is computationally by far the fastest of all implemented

rules.
Scott's rule (Scott 2010) does not calculate $n$ directly, but rather $h$, the optimal width for the lag classes using equation (A.9):

$$h = \sigma\left(\frac{24 * \sqrt{\pi}}{s}\right)^{\frac{1}{3}} \tag{A.9}$$

Where $\sigma$ is the standard deviation of $s$. By taking $\sigma$ into account, Scott's rule works good for large datasets. It's application does not work very well on distance matrices with outliers, as the standard deviation is sensitive to outliers.
If Scott's rule does, due to outliers, not yield suitable lag classes, the Freedman-Diaconis estimator (Freedman and Diaconis 1981) can be used. This estimator is similar to Scott's rule, but makes use of the inter-quartile range as shown in equation (A.10):

$$h = 2\frac{IQR}{s^{1/3}} \tag{A.10}$$

The inter-quartile range ($IQR$) is robust to outliers, but in turn the Freedman-Diaconis estimator usually estimates way too many lag classes for smaller datasets. The author cannot recommend to use it for distance matrices with less than 1000 entries.
Finally, Doane's rule (Doane 1976) is available. This is an extension to Sturge's rule, that takes the skewness of the sample into account. This makes it especially suitable for smaller, non-normal datasets, where the other estimator do not work very good. It is defined as given in equation (A.11):

$$\begin{aligned} n &= 1 + \log_2(s) + \log_2\left(1 + \frac{|g|}{k}\right) \\ g &= E\left[\left(\frac{x - \mu_g}{\sigma}\right)^3\right] \\ k &= \sqrt{\frac{6(s-2)}{(s+1)(s+3)}} \end{aligned} \tag{A.11}$$

Here, $g$ is the skewness, $\sigma$ is the standard deviation, $\mu_g$ is the arithmetic mean and $x$ is each element in $s$.
All rules that calculate the number of lag classes use the `numpy` implementation of the respective methods (van der Walt et al. 2011).

All histogram estimation methods given above just calculate the number of lag classes. The resulting classes are all equidistant, except for the first lag class, which has 0 as a lower bound, instead of $min(s)$.

Finally, SciKit-GStat implements two other methods. Both are based on a clustering approach and need the number of lag classes to be set by the user. The distance matrix is clustered by the chosen algorithm. Depending on the clustering algorithm, the cluster centers (centroids) are either estimates of high density or points in the value space, where most neighboring values have the smallest mean distance. Thus, the centroids, are taken as a best estimate for lag class centers. Each lag class is then formed by taking half the distance to each sorted neighboring centroid as bounds. This will most likely result in non-equidistant lag classes.
The first option is to use the K-Means clustering algorithm, which is maybe the most popular clustering algorithm. The method is often attributed to MacQueen et al. (1967), but there are thousands of variations and applications published. The implementation of K-Means used in SciKit-GStat is taken from `scikit-learn` (Pedregosa et al. 2011a). One important note about K-Means clustering is, that it is not a deterministic method, as the starting points for clustering are taken randomly. In practice this means, that exactly the same `Variogram` instantiated twice can result in different lag classes. Experimental variograms are very sensitive to the lag classes. In some unsystematic tests undertaken by the author, the variations in lag class edges could be as large as 5% of the distance matrix range, which would result in substantially different experimental variograms. Thus, the decision was made to seed the random start values. For this reason, the K-Means implementation in SciKit-GStat is deterministic and will always return the same lag classes for the same distance matrix. The downside is, that the clustering loses some of it's flexibility and can't be cross-validated. Additionally, the K-Means might not converge. In these cases the `Variogram` class raises an exception and invalidates the variogram. Furthermore, the K-Means will find one set of lag classes, not necessarily the best one. However, the user can still calculate lag class edges externally, using K-Means, and pass the edges explicitly to the `Variogram` class.
The other clustering algorithm is a hierarchical clustering algorithm (Johnson 1967). These algorithms group values together based on their similarity. SciKit-GStat uses an agglomerative clustering algorithm, which uses Ward's criterion (Ward Jr and Hook 1963) to express similarity. Agglomerative algorithms work iteratively and deterministic, as at first iteration each value forms a cluster on its own. Each cluster is then merged with the most similar other cluster, one at a time, until all clusters are merged, or the clustering is interrupted. Here, the clustering is interrupted as soon as the specified number of classes is reached. The lags are

then formed similar to the K-Means method, either by taking the cluster mean or median as center. Ward's criterion defines the one other cluster as the closest, that results in the smallest intra-cluster variance for the merged clusters. That, finally results in slightly different lag class edges than K-Means. The main downside of the agglomerative clustering is that it is by far the slowest method. In some cases, especially for larger datasets, the clustering took longer than the full workflow to estimate a variogram and fit a theoretical model, by magnitudes.
The implementation follows `scikit-lean` (Pedregosa et al. 2011a). Using the `AgglomerativeClustering` class with the `linkage` parameter set to `'ward'`.

One method of utilizing clustered lag classes is to compare the K-Means lag edges with the default settings. The idea is to minimize the deviation of both while searching a suitable amount of classes. This combines the advantages of K-Means, while yielding equidistant lag classes, that have the best match to clustered centroids. SciKit-GStat makes that possible, while leaving the interpretation to the user.

Another option available is called *stable entropy*. This is a custom optimization algorithm, that has not been reported before. The algorithm takes the number of lag classes as a parameter and starts with the equidistant lag classes as a initial guess for optimization. It seeks to adjust bin edges until all lag classes show a comparable Shannon entropy. The Shannon entropy is calculated using equation (A.15), with a static binning created analogous to equation (A.8), the square-root rule for histogram estimation. The lag classes are optimized by minimizing the absolute deviation in Shannon entropy, at a maximum of 5000 iterations. The algorithm uses the Nelder-Mean optimization (Gao and Han 2012) implemented in `scipy` (Virtanen et al. 2020). As the Shannon entropy is a measure of uncertainty based on information content, it is expected to yield statistically robust lag classes. At the same time it is expected to show the same limitations as the uniformly sized lag classes, such as a potentially difficult interpretation of variogram parameters.

#### sub-module: estimators

SciKit-GStat implements a number of semi-variance estimators. It includes all semi-variance estimators that are commonly used in the literature.

**numba support**: The `numba` package offers function decorators, that enable just-in-time compilation of Python code. Although there are ways to compile code even more effectively (i.e. `cython`, `nuitka` packages), `numba` comes at zero implementation overhead and fair calculation speed ups. The `numba` decorator is implemented for the matheron, cressie, entropy and genton estimators. For the

other estimators, the just-in-time compilation adds more compiling overhead, than a compiled version actually gains performance on reasonable data sample sizes. The main reason is, that the remaining estimators are already covered mathematically by a numpy function, which are in most cases already implemented in a compiled language.

**matheron**: The `matheron` function implements the Mathéron semi-variance $\gamma$ (Matheron 1963). This estimator is so commonly used, that it is often referred to just as *semi-variance* and thus the obvious default estimator in SciKit-GStat. It is defined in equation (A.1).

**cressie** implements the Cressie-Hawkins estimator $\gamma_c$ (Cressie and Hawkins 1980). As given in equation (A.12):

$$2\gamma_c(h) = \frac{(\frac{1}{N(h)} \sum_{i=1}^{N(h)} |Z(s_i) - Z(s_{i+h})|^{0.5})^4}{0.457 + \frac{0.494}{N(h)} + \frac{0.045}{N^2(h)}} \tag{A.12}$$

Where $N(h)$ is the number of point pairs $s, s_i$ at separating lag $h$ and $Z(s)$ is the observation value at $s$.

**dowd** implements the Dowd estimator $\gamma_D$ (Dowd 1984). As given by equation (A.13):

$$2\gamma_D(h) = 2.198 * median(Z(s_i) - Z(s_{i+h}))^2 \tag{A.13}$$

This estimator is based on the median value of all pair-wise differences $s_i, s_{i+h}$ separated by lag $h$, where $Z(s)$ is the observation value at location $s$. Thus, the Dowd estimator is very robust to outliers in the pair-wise differences and very fast to calculate.

**genton** implements the Genton estimator $\gamma_G$ (Genton 1998). As given by equation (A.14):

$$\begin{aligned} \gamma_G(h) = 2.2191\{|Z_i(s_i) - Z_j(s_j)|; i < j\}_{(\frac{k}{q})} \\ k = \begin{pmatrix} [N(h)/2] + 1 \\ 2 \end{pmatrix} \\ q = \begin{pmatrix} N(h) \\ 2 \end{pmatrix} \end{aligned} \tag{A.14}$$

Where the pair-wise differences $Z(s_i), Z(s_j)$ at separating lag $h$ are only used if $i < j$. The $n^{th}$ percentile is calculated from $k, q$, which are both binomial that only depend on the number of point pairs $N(h)$. The implementation in SciKit-GStat

simplifies the application of the equation by setting $\frac{k}{q} := 0.25$ for $N(h) >= 500$. This avoids the necessity to solve very large binomials at negligible errors, as $\lim_{N(h)\to\infty} \frac{k}{q} = \frac{1}{4}$. The author has found the Genton estimator to yield a reasonable basis for variogram estimation in many environmental applications (a personal, maybe biased observation). However, calculating the binomials requires some time. Especially if there are a lot of lag classes and a considerable amount of them does not fulfill the $N(h) >= 500$ constraint, it will slow the calculation down by many magnitudes compared to the other estimators.

**`minmax`** implements a custom estimator. The author is not aware of any publication of this estimator. It was introduced during development, as it has quite predictable statistical properties. However, I am also not aware of any useful practical applications of this estimator and can thus not recommend using it in typical geostatistical analysis workflows.
The MinMax estimator divides the value range of pairwise differences by their mean value.

**`entropy`**: Is an implementation of the Shannon Entropy $H$ (Shannon 1948) as a semi-variance estimator. An successful application of Shannon Entropy as a measure for similarity in dependence of spatial proximity has been reported by Thiesen et al. (2020). The Shannon Entropy is defined with equation (A.15):

$$H(h) = -\sum_{i=1}^{N(h)} p_i log_2(p_i) \tag{A.15}$$

Where $p_i$ is the empirical exceeding probability of $Z(s_i) - Z(s_{i+h})$ for each separating lag $h$. To calculate the empirical probabilities of occurrence, a histogram of all pairwise differences is calculated. This histogram has evenly spaced bin edges and the user can set the amount of bins as a hyper-parameter to `entropy`. Alternatively, the bin edges can be set explicitly. One has to be aware that the Shannon Entropy relies on a suitable binning of the underlying data. This might need some preliminary examination of $Z(s_i) - Z(s_{i+1})$, which is readily accessible as a property. It is highly recommended to use exactly the same bin edges for all separating distances $h$ needed to process a single variogram. Otherwise the entropy values and their gradient over distance is not comparable and the whole variogram analysis turns meaningless.

Finally, it is possible to use custom user-defined functions for estimating the semi-variance. The function has to accept a one dimensional array of pair-wise differences, as these are already calculated by the `Variogram` class. The return

value must be a single floating point value. This can either be the primitive Python type or a 64 bit `numpy` float. The given function is finally mapped to all separating distance lags automatically, thus there is no need to implement any overhead, such as sorting or grouping, by the user. This empowers users with little or no experience in Python to define new semi-variance estimators as only the mathematical description of the semi-variance is needed as Python code.

### sub-module: models

SciKit-GStat implements a number of theoretical variogram models. The most commonly used models from literature are available. However, during researching theoretical models, the author brought an almost limitless number of models, or variations thereof to light. Thus, the process of implementing new models was eased as far as possible, instead of implementing anything that could be useful. Any variogram model function (implemented and custom) will receive the *effective range* as a function argument and is fitted using it. In case the mathematical model of a variogram function uses the range parameter, one has to implement the conversion into the model function as well.

The core design decision for SciKit-GStat's theoretical variogram models was to implement a decorator, that wraps any model function. This decorator takes care of handling input data and aligning output data. Thus, the process of implementing new variogram models is simplified to writing a function that maps a single given distance lag to the corresponding semi-variance value.
Each model will receive the three variogram parameter effective range, sill and nugget as function arguments. The nugget is implemented as a optional argument with a default value of zero, in case the user disables the usage of a nugget in the `Variogram` class. Custom variogram models have to reflect that behavior.

**`spherical`** is the implementation of the spherical model, which is one of the most commonly used variogram models. Thus, the spherical variogram model is the default model, in case the user did not specify a model explicitly. The model equation is taken from Burgess and R. Webster (1980) and given in equation (A.16):

$$\gamma(h) = \begin{cases} b + C_0 * \left(1.5 * \frac{h}{a} - 0.5 * \frac{h}{a}^3\right) & h < a \\ b + C_0 & h \geq a \\ & a := r \end{cases} \tag{A.16}$$

Where $h$ is the distance lag and $b, C_0, a$ are the variogram model parameters: nugget,

sill and range. The range of a spherical model is defined to be exactly the effective range $r$.

**`exponential`** is the implementation of the exponential variogram model. The implementation is taken from A G Journel and Huijbregts (1976) and given in equation (A.17):

$$\begin{aligned} \gamma(h) &= b + C_0 * \left(1 - e^{-\frac{h}{a}}\right) \\ a &= \frac{r}{3} \end{aligned} \tag{A.17}$$

Where $h$ is the distance lag and $b, C_0, a$ are the variogram model parameters: nugget, sill and range. For the exponential model, the effective range $r$ is different from the variogram range parameter $a$.

**`gaussian`** is the implementation of the Gaussian variogram model. The implementation is taken from A G Journel and Huijbregts (1976) and given in equation (A.18):

$$\begin{aligned} \gamma(h) &= b + c_0 * \left(1 - e^{-\frac{h^2}{a^2}}\right) \\ a &= \frac{r}{2} \end{aligned} \tag{A.18}$$

Where $h$ is the distance lag and $b, C_0, a$ are the variogram model parameters: nugget, sill and range. For the Gaussian model, the effective range $r$ is different from the variogram range parameter $a$. In SciKit-GStat, the conversion from effective range to range parameter is implemented as shown in equation (A.18). However, the author is aware of other implementations in literature. The package does not allow to somehow switch the conversion and the user has to implement a new Gaussian model, in case another conversion is desired.

**`cubic`** is the implementation of the cubic variogram model. The implementation is taken from Montero et al. (2015) and given in equation (A.19):

$$\gamma(h) = \begin{cases} b + C_0 * \left[7 * \left(\frac{h^2}{a^2}\right) - \frac{35}{4} * \left(\frac{h^3}{a^3}\right) + \frac{7}{2} * \left(\frac{h^5}{a^5}\right) - \frac{3}{4} * \left(\frac{h^7}{a^7}\right)\right] & h < a \\ b + C_0 & h \geq a \end{cases} \quad a := r \tag{A.19}$$

Where $h$ is the distance lag and $b, C_0, a$ are the variogram model parameters: nugget, sill and range. For the cubic model, the effective range $r$ is exactly the variogram range parameter $a$.

**`matern`** in the implementation of the Matèrn variogram model. The implementation is taken from Zimmermann et al. (2008) and given in equation (A.20):

$$\gamma(h) = b + C_0\left(1 - \frac{1}{2^{\upsilon-1}\Gamma(\upsilon)}\left(\frac{h}{a}\right)^{\upsilon} K_{\upsilon}\left(\frac{h}{a}\right)\right) \qquad a = \frac{r}{2} \tag{A.20}$$

Where $h$ is the distance lag, $\Gamma$ is the gamma function and $b, C_0, a$ are the variogram model parameters: nugget, sill and range. Additionally, the Matérn model defines a fourth model parameter $\upsilon$, which is a smoothness parameter. For the Matérn model, the effective range $a$ is a fraction of the variogram parameter range $r$.

**`stable`** is the implementation of the stable variogram model. The implementation is taken from Montero et al. (2015) and given in equation (A.21):

$$\gamma(h) = b + C_0 * \left(1. - e^{-\frac{h}{a}^s}\right) \qquad a = \frac{r}{3^{s^{-1}}} \tag{A.21}$$

Where $h$ is the distance lag and $b, C_0, a$ are the variogram model parameters: nugget, sill and range. Additionally, the stable model has a shape parameter $s$. The effective range of the variogram is a fraction of the variogram range parameter, dependent on this shape. Generally, the effective range will increase with larger shape values.

**`harmonize`** is an implementation, that is rather uncommon in geostatistics. It is based on the idea of monotonizing a data sample into a non-decreasing function. That means, there is no model fitting involved and the procedure bypasses all related steps. A successful application in geoscience was reported by Hinterding (2003). For SciKit-GStat, the more generalized approach of isotonic regression (Chakravarti 1989) was used which is already implemented in `scikit-learn` (Pedregosa et al. 2011a).
Note, that a harmonized model might not show an effective range, in which cases the library will take the maximum value as the effective range for technical reasons. Thus, the user has to carefully double-check harmonized models for their geostatistical soundness. Secondly, the harmonized model cannot be exported to `gstools`, which makes it unavailable for most kriging algorithms.

### Fitting theoretical models

As soon as an estimated variogram is used in further geostatistical methods, such as kriging or field simulations, it is necessary to describe the experimental, empirical data by a model function of defined mathematical properties. I.e., for kriging, a variogram has to be monotonically increasing and positive definite. This is assured, by fitting a theoretical model to the experimental data. The models available in SciKit-GStat are described in section A.5.1.

Fitting the theoretical model to the experimental data is crucial, as any uncertainty caused by this procedure will be propagated to any further usage of the variogram. Almost any geostatistical analysis workflow is based on some kind of variogram and hence, the goodness of fit will influence almost any analysis. The `Variogram` class can return different parameters to judge the goodness of fit, among other the coefficient of determination, root-mean squared error and mean squared error. Beyond a direct comparison of experimental variogram and theoretical model, the `Variogram` class can run a leave-one-out cross validation of the input locations to assess the fit based on kriging. As the experimental values and their modeled counterparts are accessible for the user at all times, implementations of any other desired coefficient are straightforward.

When fitting the model, SciKit-GStat implements four main algorithms, each one in different variations. A main challenge of fitting a variogram model functions is, that closer lag classes result in higher kriging weights and are therefore of higher importance. A variogram model that might show a fair overall goodness of fit, but is far off on the first few lag classes, will result in poorer kriging results, than an overall less well fitted model that hits the first few lags perfectly. On the other hand, emphasizing the closer lags is mainly done by adjusting the range parameter. The only other degree of freedom for fitting the model is then the sill parameter. Thus, if the modeling of the closer lag classes is put too much into focus, this happens at the cost of missing the experimental sill, which is basically the sample variance, in case the nugget is set to zero. If the nugget is not zero, a insufficient sill will change the nugget to sill ratio and one might have reject the variogram at all. A kriging interpolation of reasonable range is able to reproduce the spatial structure of a random field, but if the sill is far off, the interpolation is not able to reproduce the value space accordingly and the estimations will be inaccurate. In the extreme case of a pure nugget variogram model, kriging will only estimate the sample mean (which is the correct behavior, but not really useful). Thus, the fitting of a model has to be evaluated carefully by the user and SciKit-GStat is aiming to support the user with this.

A procedure that is frequently used to find optimal parameters for a given model to fit a data sample is *least squares*. These kinds of procedures find a set of parameters, that minimize the squared deviations of the model to observations. A robust, widely spread variant of least squares is the Levenberg-Marquardt algorithm (Moré 1978). It is a robust and fast fitting algorithm that yields reasonable parameters in most cases. However, Levenberg-Marquardt is an unbounded least-squares algorithm, meaning that value space for the parameters can neither be limited, nor constrained. In the specific case of variogram model fitting, there are a number of assumptions that actually do constrain the parameter space. Thus, in some occasions, Levenberg-Marquardt is failing to find optimal parameters, as it is searching parameter regions, that would not be valid variogram parameters, anyway. The implementation for Levenberg-Marquardt least squares is taken from the `scipy` package (Virtanen et al. 2020).

Another least-squares approach is Trust-Region Reflective (TRF) (Branch et al. 1999). A major difference to Levenberg-Marquardt is that TRF is a bounded least-squares algorithm. That means, the `Variogram` class can set lower and upper limits for each of the parameters. Thus, the TRF is, from what I can say, always finding suitable parameters and is therefore the default fitting method in SciKit-GStat.
The adjustable variogram model parameters are the effective range, sill, nugget, if used, and a shape parameter for the Matèrn and stable model. The lower bound for all parameters is zero, as all parameters have to be positive by definition. The upper bounds can also be defined for all parameters. The effective range is bounded to the maximum lag, or largest separating distance observed, if the maximum lag was not specified by the user. The sill is bounded by the largest semi-variance value that was estimated for the experimental variogram. As nugget and sill effectively sum up to sample variance, it consequently has to be smaller than any individual semi-variance value. The same has to hold for the nugget, due to the implementations given in section A.5.1. For technical reasons, the sill must not be 0. The nugget has the same upper bound as the sill, as TRF does not take constrains, only parameter bounds (a constraint would put a dependency of one parameter to the other into the algorithm, which would be the more appropriate handling here).
The implementation for Trust-Region Reflective least squares is taken from the `scipy` package (Virtanen et al. 2020).

The third fitting method is a maximum likelihood approach. The theoretical model is fitted to the experimental data by minimizing the negative log-likelihood of the variogram parameters. Each of the parameters samples from a normal distribution with the last parameters predictions mean and standard deviation as first and second moment. In the current implementation, an unbounded and

unconstrained Nelder-Mead solver (Gao and Han 2012) is used to minimize the log-likelihood function. The implementation is taken from `scipy` (Virtanen et al. 2020). For rare cases where this solver is not able to find valid variogram parameters, the SLSQP (Kraft et al. 1988) algorithm can be used. It is substantially slower but more flexible and will search the best parameters in a valid parameter space only. Without having performed a systematic testing beyond unit-tests for the maximum likelihood option, it seems like the maximum likelihood estimation often struggles with larger nugget values and does not find optimal variogram parameters. Note that this approach is optimizing the variogram parameters by their likelihood of fitting to the experimental data, it is not a maximum likelihood fitting of the variogram model to the sample auto-corrleation as described i.e. by Lark (2000). The latter approach is briefly described in appendix A.9.3.

The last option is not an algorithm. The `Variogram` class has the ability to directly take the variogram parameters from the user as hyper-parameters. In these cases the class will bypass the fitting procedures and just set the user input as fitting coefficients. This is convenient for cases where the user receives the parameters externally. It is also possible to switch to custom fitting, after another algorithm had already been used. This can be helpful to fine-tune automatically fitted parameters. On the other hand, the implementation does also bypass all checks and constrains made to the parameter space and the user could i.e. pass invalid values. An example is a negative nugget value, which is mathematically applicable (there is i.e. no runtime error), but does not make any sense from a geostatistical point of view. Ensuring variogram validity is completely in the responsibility of the user in these cases.

All fitting mechanisms except for the manual fit, can be further refined by setting an array of fitting weights. This enables the user to focus only a few lag classes for fitting and achieve a higher goodness of fit on specific lags. The weights are, following the logic of `scipy`, actually not weights, but uncertainties. Thus, if one has only weights available, their inverse has to be used. It is possible to pass a numeric value array to the `Variogram` class, that has to be of same length as the number of lag classes. If not set, the `Variogram` will equally weight all lag classes. In most other cases the user will want to apply decreasing weights with increasing separating distance, to put more focus on the first few lag classes. SciKit-GStat conveniently includes a number of functions, that calculate an uncertainty array that will effectively apply decreasing weights.
The first option is a linear decrease of weights with increasing lags. The second option uses the square-root of the the normalized lag as an approximation. The third option uses the inverse of the normalized lag squared as a weight. This results

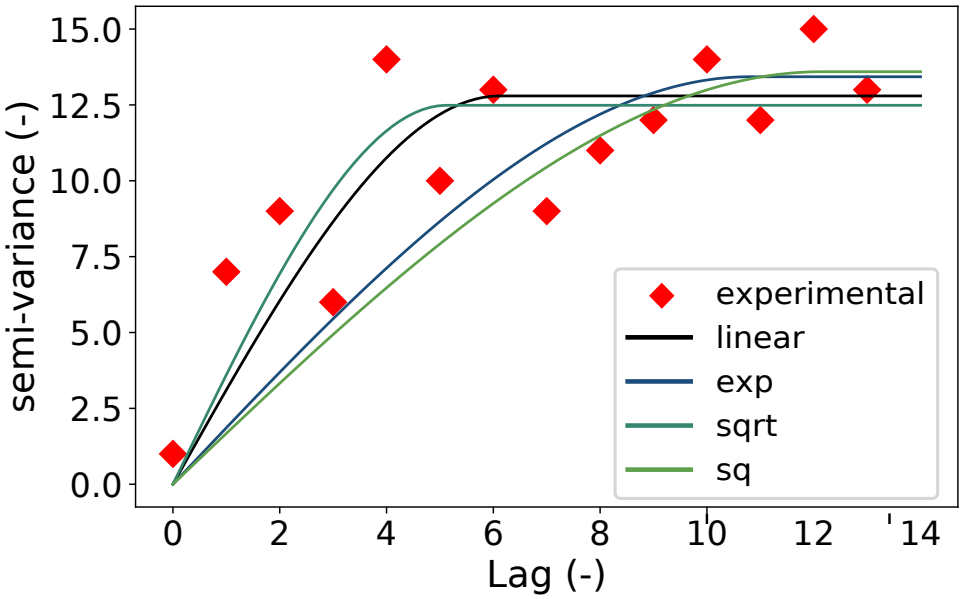



**Figure A.9:** Red squares show a sample experimental variogram (values are made up) with four different spherical variogram models. All four models are fitted using Trust-Region Reflective fitting procedure and distance depended weights. The weights are **linear** decreasing with distance (blue line), decreasing by the **square-root** of the normalized distance (green line), the **squared** normalized distance (red line) and decreasing by **e-function**, as shown in equation (A.22) (yellow line).

in completely neglecting any lag class but the first two or three, depending on the total amount. The last function applies an exponential function as given by equation (A.22):

$$\frac{1}{w} = e^{lag_n^2} \tag{A.22}$$

Where $w$ is the calculated weight and $lag_n$ the normalized lag.
All four distance-dependent weighting functions are compared in figure A.9. All four functions show very comparable coefficients of determination, calculated over all lag classes. That means, the four models describe the experimental variogram equally well. It is now up to the user to decide which one to use. SciKit-GStat does not apply any of these distance weighting functions automatically. This example illustrates, how important it is to examine experimental variograms and the many possibilities how one can capture its properties in a theoretical model, before approaching more complex geostatistical methods like kriging or field generation. Otherwise, the choice of model and model parameters might seem arbitrary. To illustrate this, the four models resulting solely from a different weighting of the lag

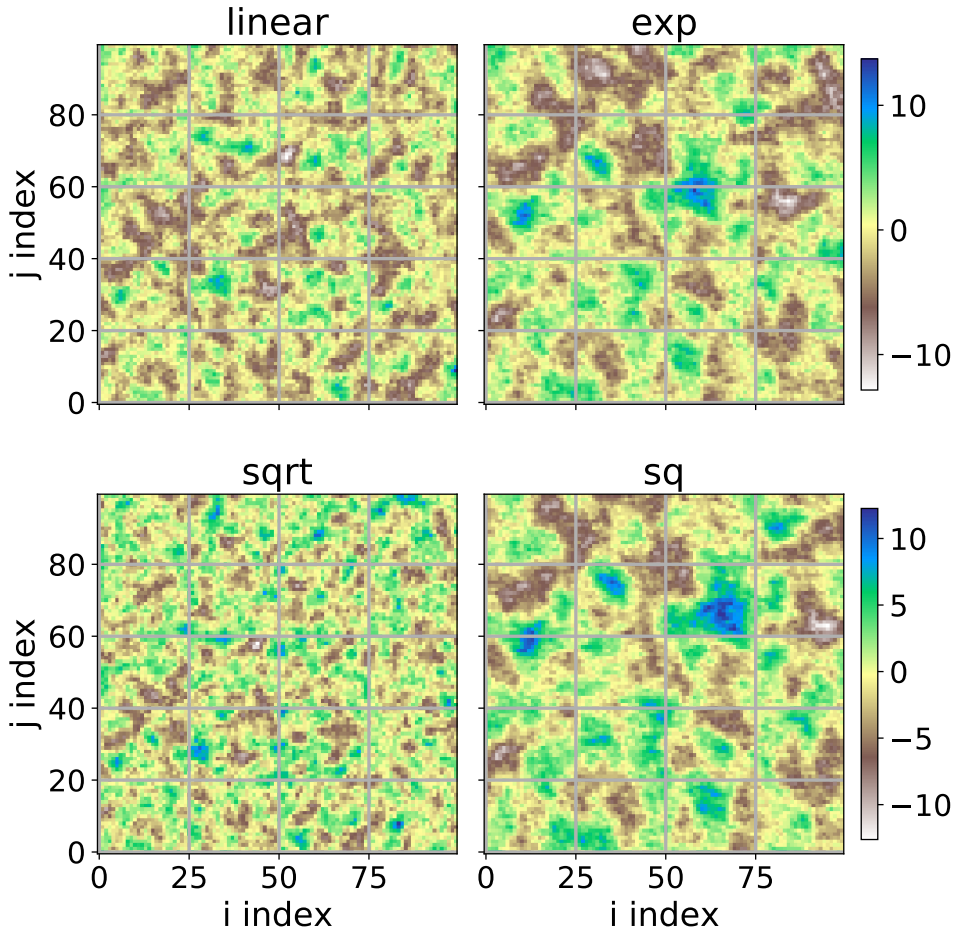



**Figure A.10:** Four random fields generated using the same seed for randomization, which results in exactly the same field for same input. The only differing input parameter is the automatic distance weighting function that was used for fitting the theoretical variogram model. All four fields share the same value range. As the underlying models were made up, neither the values nor the axis coordinates have any meaning. The two coordinate axes name correspond to the index of the random field in matrix form $i$, $j$.

classes for fitting (figure A.9) were used to generate a random field. The generation of the random field was seeded with a fixed value, in order to create reproducible results and hence the only difference in the fields originates from the choice of weighting function (figure A.10). Finally, the fitting of variogram models is usually neither exposed to the user (as sometimes even not the variogram itself), nor does the user have control on the internals of fitting. In the shown example (figure

A.10), only one parameter that influences fitting was changed, and that shows dramatic effects. SciKit-GStat seeks to give the user more options to assume control over this important step. Each of the other options for fitting might well produce similar dramatic changes in field generation. Hence, it is so important to assess automatically derived fitting results, because finally it should be up to human interpretation whether a variogram should be used or not.

Another predefined possibility to determine weights for fitting is information theory. Unlike the other functions, this option is not based on an inverse of weights. The information theory based weighting option calculates the uncertainties directly, by using the Shannon Entropy (Shannon 1948). It is calculated for the empirical distribution of point pairs within each distance lag class. This will link the weight during fitting directly to the information content of that lag. From a practical point of view, the resulting weights are usually closer to uniform weights, than the distance dependent weights. For the distance weighted procedures, the larger lags are almost completely ignored. With the information theoretic approach this will only happen for very thin populated lag classes.

### Directional variograms

Directional variograms can be estimated in SciKit-GStat using the `DirectionalVariogram` class. It inherits from `Variogram`, making all its properties and methods available. Only methods that actually work on the distance matrix are re-implemented to intercept calculations with a spatial filter. This let's the user interact with the class as learned with the base class, focusing only on the differences between a directional variogram calculation and a classic.

`DirectionalVariogram` only overwrites one internal method and one property of the base class. This is the logic assigning the correct lag group to each point pair calculated and then deriving the lag bin edges from this. In both cases, point pairs are filtered by their orientation, before the calculation is continued. This way, `DirectionalVariogram` only adds necessary calculations steps and the base class does not have to handle data, information or logic (such as point pair orientation) that does not affect the classic calculation. This conscious design decision leaves the code as readable as possible to make contributions easier for others.

Three new attributes are introduced, that can be set by the user. For all three parameters SciKit-GStat retains the name, implementation and usage as close to Montero et al. (2015) as possible.
The **azimuth** of the directional variogram is the direction for which the directional

variogram will be calculated. It is given in degrees as a counter-clockwise deviation from the coordinate x-axis (which will be East in most cases). The **tolerance** is an angle in degrees, which defines the limit at which a deviation from the azimuth is still acceptable. Only these point pairs will be taken into account, which orientation as calculated with equation (A.6) are within the tolerance of the azimuth. The tolerance defaults to 45 degrees.
As the tolerance is given in degrees, the absolute deviations in the unit of the coordinate system can be quite considerable for larger separating distances. Therefore, it is possible to set a **bandwidth**. This parameter limits the maximum acceptable perpendicular distance from the azimuth vector in coordinate units and default to the 33 % percentile of the distance matrix. It can be set as a percentile or as a absolute limit, in coordinate units.

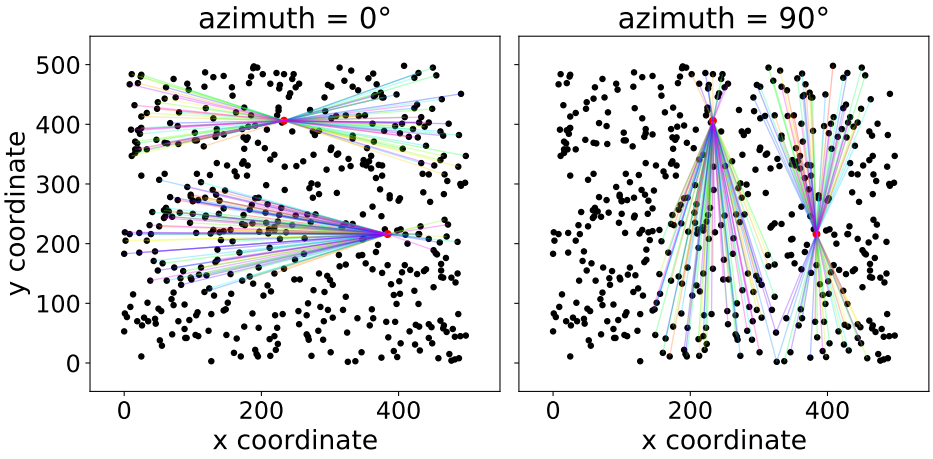



**Figure A.11:** Pair Field plot of two directional variograms. The plot was created with exactly the same two directional variogram instances as used in figure A.5. Both figures show the network graph for two observation points (index 42 and 170 in the sample file), in both directions of the variogram. The lines connect all point pairs that were taken into account for these two points. The line colors have no meaning and are just included for visual reasons.

Apart from the basic hyper-parameters that define a directional variogram, there are different implementations, how to apply them. SciKit-GStat denotes these implementations as *directional models* and implements two different.
The default *triangle* model is applying the three directional parameters as most often reported in literature (Montero et al. 2015), by constructing a triangle in the direction of the azimuth using the tolerance as a opening window. For larger

distances, the triangle is bounded by the bandwith and turned geometrically into a rectangle.
The unbounded version of the *triangle* model is called *compass*, which simply ignores the bandwidth parameter. Thus, it will only restrict point pairs to be oriented into a specific direction.
For convenience and to further inspect the point pairs which are actually taken into account, there is an additional auxiliary plotting method. This plots a network graph for all input locations with a edge for each point pair that will be taken into account for calculation (figure A.11). Unlike other network graphs, the vertices keep their real locations in the coordinate space to identify specific input data points. A plot like this can be helpful to specify reasonable azimuth and tolerance values, which will highly impact the result.

### Spatio-temporal variogram

For calculating spatio-temporal variograms, SciKit-GStat has a class called `SpaceTimeVariogram`. Other than the `DirectionalVariogram` class, `SpaceTimeVariogram` does not inherit from `Variogram`, but is an independent class. For a spatio-temporal variogram, any processing step is not only dependent on a spatial lag, but also on a temporal lag. This actually changes the function signatures for almost all methods and, thus it was decided to re-implement the whole class without any inheritance. Nevertheless, `SpaceTimeVariogram` and `Variogram` share attribute and method names wherever possible.

At the core of all implemented theoretical variogram methods for the spatio-temporal variogram is the estimation of two marginal variograms. The class will estimate a *temporal* and a *spatial* marginal variogram. These are both instances of the `Variogram` class. The spatio-temporal models themselves expect both marginal variograms as an attribute.

Finally, the `SpaceTimeVariogram` implements a rich plotting method. It can plot the the experimental spatio-temporal variogram and the fitted theoretical model as a 3D or 2D plot (figure A.6, A.7). For 2D plotting, different plot types are implemented, i.e. a contour plot for semi-variance values. Both 2D and 3D plots are available. 3D plots allow for the user to interactively rotate, pan and zoom the plot, enabling the user to inspect a spatio-temporal variogram. 2D plots are helpful for printed material.

### sub-module: stmodels

SciKit-GStat implements three different theoretical spatio-temporal variogram models: the sum, product and sum-product model. In line with the `models` sub-module, the `stmodels` sub-module has a decorator functions to wrap the models. This decorator takes care of the data flow and leaves the implementation of the mathematical formula to the user, if custom models should be used.
In the following equations the marginal variograms represented by $\gamma_x, \gamma_t$ refer only to the spatial lag $h$ or temporal lag $t$, respectively. They are estimated and modeled as described in section A.4.1 using any of the semi-variance estimators from section A.5.1 and any model described in section A.5.1.

**`sum`** is the implementation of the sum model. This is the most basic spatio-temporal model, a sum of a spatial marginal variogram $V_x(h)$ and a temporal marginal variogram $V_t(t)$ as shown in equation (A.23):

$$\gamma(h,t) = \gamma_x(h) + \gamma_t(t) \tag{A.23}$$

Where $\gamma_x, \gamma_t$ are the semi-variance estimations by the two marginal variograms and are not restricted to a specific semi-variance estimator or theoretical model.
The sum model provides an understanding of the idea and workflow of spatio-temporal models. However, it should not be used for real data in almost all cases. It assumes the covariance field to be isotropic across temporal and spatial dimensions. A situation which can be considered rarely true. Moreover, it might not be positive definitive, as required for variogram models (Dimitrakopoulos and Luo 1994; Myers and A. Journel 1990).

**`product`** is the implementation of the product model. The implementation is taken from De Cesare et al. (2002, equation (4), p.207) as shown in equation (A.24):

$$\gamma(h,t) = C_x * \gamma_t(t) + C_t * \gamma_x(h) - \gamma_x(h) * \gamma_t(t) \tag{A.24}$$

Where $C_x$ is the sill parameter of the spatial marginal variogram $\gamma_x(h)$ and $C_t$ is the sill of the temporal marginal variogram $\gamma_t(t)$.

**`product_sum`** is the implementation of the product-sum model. The implementation is taken from De Cesare et al. (2002, equation (6)) as shown in equation (A.25):

$$\gamma(h,t) = [k_1 C_T + k_2] * \gamma_x(h) + [k_1 C_s + k_3]\gamma_t(t) - k_1 \gamma_x(h) x \gamma_t(t) \tag{A.25}$$

Here, $k_1, k_2, k_3$ are additional fitting parameters needed for the product-sum model.

All three parameters need to be positive and may not be larger than any of the marginal sill parameters $C_x, C_t$.

#### Ordinary Kriging

SciKit-GStat implements a ordinary kriging algorithm. It is implemented following Montero et al. (2015) and can be used using the class `OrdinaryKriging`. The user needs to pass a instance of `Variogram` as a parameter. In a majority of other kriging implementations, the procedure accepts the observations and estimates a variogram automatically. Sometimes even as an internal processing step. For SciKit-GStat, the decision was made to focus on variogram estimation. The kriging class should be seen as an auxiliary class to implement the full typical geostatistical analysis workflow. The user is encouraged to take a closer look on the variogram, utilizing all the plotting routines and descriptions, before passing it on to the kriging class. This should have a positive effect on geostatistical applications.

It must be noted that the `OrdinaryKriging` class is mainly implemented for cross validating variogram models. It does not claim to be a very performing implementation of the kriging algorithm. Nor is it implemented with the flexibility and analysis tools, the `Variogram` has. The author is also aware, that further kriging algorithms exist and ordinary kriging might not be the most useful one. Thus, SciKit-GStat is more focused on implementing interfaces to other libraries, that including other kriging methods. Namely these are `gstools` and `pykrige`. To date, the two aforementioned libraries are aligned to each other, future `pykrige` iterations will implement `gstools` co-variograms. This will leave SciKit-GStat only with the need for a powerful interface to `gstools` to provide the full power of `pykrige` to SciKit-GStat users. The SciKit-GStat `Variogram` class has an interface function, that can instantiate any `gstools` kriging algorithm from a SciKit-GStat variogram. More details on SciKit-GStat and `gstools` and their future co-existance are given in section A.5.2.

### A.5.2 SciKit-GStat and `gstools`

SciKit-GStat has three interfaces to `gstools`, all three implemented as instance methods of the `Variogram` class. The first option is to export the empirical variogram. This is the combination of the lag classes edges with the experimental variogram. The lag classes edges can optionally be shifted to the class centers, as this is the notation that `gstools` uses for empirical variograms. This interface is useful in case one of the many binning functions or semi-variance estimators was

used, that is not available in `gstools`.
The second, major, option is to translate the theoretical model into a fitted covariance model instance of `gstools`, which is their respective base class. With that in place, one can use the covariance model in conjunction with all the great methods available in `gstools`.
For the specific case of kriging, a third interface exports the variogram directly into a `gstools` kriging class instance. At the time of writing, currently available kriging algorithms are simple kriging, ordinary kriging, universal or regression kriging, kriging with external drift and kriging the mean (Müller et al. 2021b).

Both libraries chose different avenues, how the user may interact with the library. For `gstools`, the user defines a covariance model and passes it to one of the rich set of geostatistical functions, which can be found in `gstools`. The user then captures the return value of the function and uses it for further development and analysis. In SciKit-GStat, as described in this paper, the user rather instantiates one object and mutates it during the analysis.

## A.6 Support, Application and Contribution

### A.6.1 User support

Users are supported by a comprehensive documentation that includes API reference, installation instructions, getting started guide, a detailed user guide and tutorials. The user guide is written at the example of a lecture script. No geostatistical prior knowledge is necessary. Only some limited experience in Python and basic knowledge of univariate statistics is advantageous. Additionally, the user guide includes a number of technical notes, that discuss some specialities of SciKit-GStat in great detail.

SciKit-GStat is managed and hosted on Github under a MIT license. For technical problems, questions and feature requests, the Github issues ticketing system is used. To date any issues arising have been processed by the author himself. As some of the raised issues discussed fundamental geostatistical principles and basic applications of SciKit-GStat, these closed issues are also a valuable resource for new users to SciKit-GStat as well as geostatistics. The evaluation of these issues was taken into account for compiling the user guide.

To use SciKit-GStat in production environments and also for rapid installation, a docker image is offered. The Dockerfile is also included into the SciKit-GStat repository ,and therefore, also distributed under MIT license enabling users to

adapt and utilize it. The associated docker image includes an interactive jupyter notebook environment, which auto-starts the tutorials. These tutorials are also included into the documentation and accompany the descriptions. In classroom situations, each student can easily start with the interactive tutorials, while the teacher can follow the documentation. The student should implement the core functionality of SciKit-GStat themselves to fully understand geostatistical analysis workflows. This knowledge can then be applied to SciKit-GStat emphasising the correct application of the package and geostatistics in general. Finally, the student can easily apply the learned techniques to real problems with a production-ready Python package. The overall aim is to rather teach geostatistics with the given resources at the example of SciKit-GStat, than narrowing geostatistics down to the application of SciKit-GStat only.

### A.6.2 Contributions

Contributions to SciKit-GStat are managed via Github. Generally, anyone can create a private copy of the full source code. Adaptions, enhancements or corrections to the source code of SciKit-GStat can be merged into the official code base via Github. With respect to coding style, technical correctness and overall objective of the library any possible contribution is reviewed by the author or any other maintainer of the package. To further guarantee technical correctness, SciKit-GStat is covered by unit-tests, which test all main functionality in isolated test cases. Due to technical challenges, most plotting routines are not covered by unit-tests. Historically, there have been a number of tests, but they require a lot of maintenance and are to a specific degree dependent on the host platform. Thus, it can be doubted that this is actually beneficial for the user. Additionally, a few tests in the style of end-to-end test (e2e) were added to run a full analysis against an expected result. Such e2e tests also assess the performance, measured as test run-time. However, dropping performance does not cause a test failure, but can be used by the author and contributors to assess contributions with respect to their influence on performance. It was also decided to not accept any new contributions that decrease test coverage significantly, by adding automatic coverage reports to new contributions. This can be considered important to assure a specific level of technical correctness for SciKit-GStat, especially because the open source MIT license does not put any warranties in place, that the user could rely on.

### A.6.3 Integration into other libraries

The main interface to `gstools` is already discussed in section A.5.2. SciKit-GStat has an interface to `pykrige`, which makes it possible to export a `Variogram` instance as kriging parameters directly into `pykrige`. However, as `pykrige` is fundamentally changing, it is not yet clear if the interface will still work in the future. Nevertheless, as the code restructuring is finished, the more powerful interface to `gstools` can be used to interact with `pykrige` in a more feature rich, natural and native way.

`scikit-learn` is the most popular data science and machine learning framework in Python. Beside that, `scikit-learn` developed a tool-chain pipeline over the past years, that is used way beyond data science. This enables the user to quickly change isolated parts of large and complex automated analysis workflows. SciKit-GStat implements an interface to the corresponding class in `scikit-learn`, which makes variogram analysis available in any workflow. At the same time, `scikit-learn` implements a great number of data transformation algorithms as usually used in machine learning. By adopting the pipeline tool-chain, these preprocessing steps can be used together with SciKit-GStat, as many of them are useful for geostatistical preprocessing as well. A prime example is trend detection and detrending, which is often necessary in geostatistics.

## A.7 Discussion

Most limitation and notes on application have already been mentioned in the respective sections, along with implementation details. This section is discussing general comments to SciKit-GStat. SciKit-GStat is toolbox for variogram estimation, equipped with a large amount of methods. Most of these methods and settings do not make sense in every situation. SciKit-GStat is generally leaving any assessment of estimated variograms, beyond numerical goodness of fit values, to the user. From this, it is further clarified, that SciKit-GStat is a variogram estimation toolbox, which is used for building geostatistical methods or conducting analyses. It is not a analysis framework itself.
This limitation also applies to preprocessing. While geostatistical prerequisites, like the intrinsic hypothesis, are mentioned and further literature is referenced, SciKit-GStat does not contain any diagnostic tool to i.e. check given input data any further than by offering the presented scatter plots in figure A.3 for visual inspection. External software needs to be used to test and transform input data. This applies to coordinate transformations as well as observation normalization if required. For

both cases flexible and powerful Python packages are available (`scipy`, `numpy`, `scikit-learn`). Hence, I had the impression that anything implemented into SciKit-GStat can't come close to existing software. Furthermore, I cannot claim to overlook all geoscientific fields in enough detail to be able to offer generic integrity checks and preprocessing for just any kind of input data. On the other hand, from my personal experience in answering Github issues, non-transformed, misused and non-applicable datasets in combination with rather uncommon variogram estimations already lead to some confusion. As an example: If one uses the stable entropy method to find lag classes, the method tries to assure that all classes are of comparable entropy. As a consequence, using the entropy as a variogram estimator will yield nugget effect models by design. If not, it is due to a weakness in method and not a statistical feature of the sample. SciKit-GStat will not stop you from doing so, nor does it stop the user from using this model for external drift kriging, which will solely use the external drift variable for interpolation, then. One might be under the impression that a sophisticated geostatistical interpolation was performed and the result is backed by the covariance of observations. In fact, one did only apply a computationally intensive averaging overlayed by a simple linear regression of the external drift term. It is up to the user to inspect the variogram and be aware of these implications. Not everything SciKit-GStat calculates is automatically correct beyond technical correctness.

Another general comment concerns spatio-temporal geostatistics. I want to clearly state here, that spatio-temporal variograms cannot be exported to any other Python package and SciKit-GStat does not include spatio-temporal kriging. An implementation is neither planned by the author, nor for `gstools` or `pykrige`, as far as I am aware. Thus, from what I can say, one has to use the wonderful `gstat` package and the R programming language, or `gslib` in FORTRAN right now. Due to the lack of kriging procedures, the spatio-temporal variogram representation of SciKit-GStat falls way behind the base class in terms of functionality and interactivity. Similar statements can be made for the directional variogram. While it is as functional, interactive and powerful as the base class, it can't be exported either. The original intention was to build a diagnostic variography tool for detecting anisotropy. It turned out, that the current design of the directional variogram is incompatible to the design in `gstools` and `pykrige`. Hence, the user has to detect anisotropy and in the case of geometric anisotropy and then transform the input data manually. This can be cumbersome and `gstools` might offer the better approach here, if kriging or field generation are the final steps.

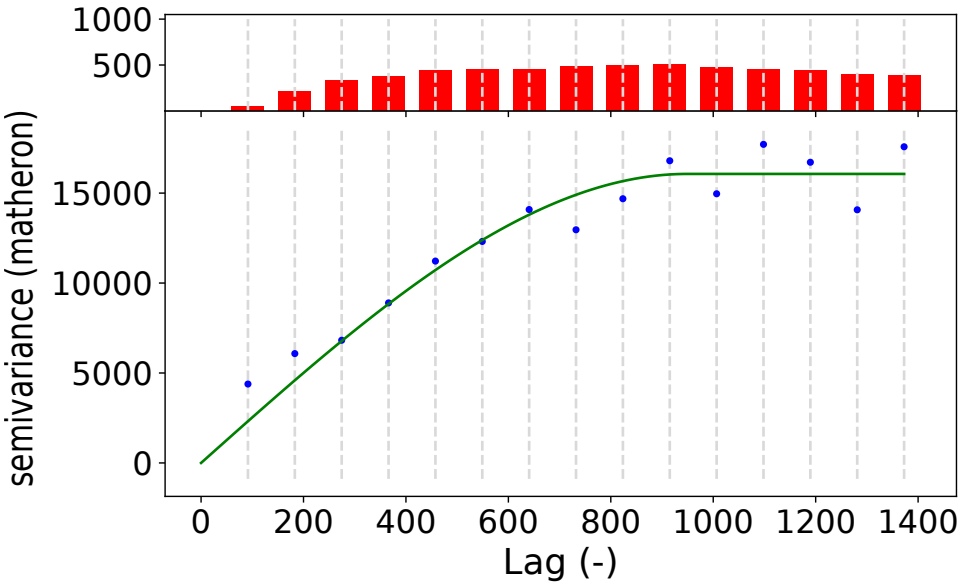



**Figure A.12:** Default sample experimental variogram (blue points) with fitted spherical model (green line) of the Meuse dataset (Bivand et al. 2008; E. Pebesma and Bivand 2005). The histogram in the upper sub-figure shows the count of point pairs for each of the 15 lag classes.

## A.8 Conclusion

With SciKit-GStat, the scientific Python community has gained a flexible, well documented and well written package for variogram estimation. SciKit-GStat enables the user to estimate variograms in almost limitless variations in a language-natural and efficient manner. Many quality measures and especially plotting routines accompany the library, to not only *do the hard work*, but also help the user to understand what was actually done. Such an educational aspect of SciKit-GStat is as important as the technical implementation details. Even the best code can be applied the wrong way to draw incorrect or skewed conclusions. If one does not write the code himself, this risk might be even higher. With SciKit-GStat the focus is on the variogram. Variograms that are better understood by a user, lead to better models, which are beneficial not only in application, but also as an educational tool.

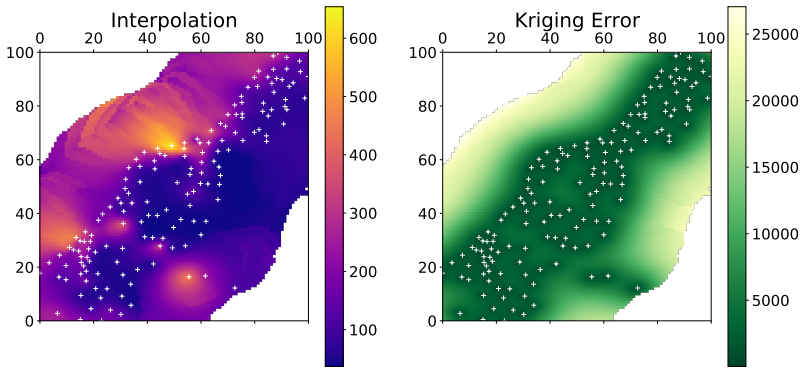



**Figure A.13:** Ordinary kriging application using the theoretical variogram model shown in figure A.12. The kriging procedure estimated the lead concentration on a 100x100 grid.

## A.9 Appendix

### A.9.1 Meuse Data

Users of SciKit-GStat that relate easier to geoscientific data samples, than to pancakes, are referred to the tutorial section of SciKit-GStat (Mirko Mälicke, Möller, et al. 2021a), which includes a sample variogram and kriging application of the Meuse dataset. This dataset is published along with the R package sp (Bivand et al. 2008; E. Pebesma and Bivand 2005) and contains 155 samples of heavy metal ions (cadmium, copper, lead, zinc) along the river Meuse in the Netherlands. In the tutorial, the lead measurements are used. While the original R package description (E. Pebesma and Bivand 2005) is not specifying the coordinate reference system used, I am confident its Amersfoort / RD New (EPSG: 28992), which projects the sample locations next to the town Stein in the Netherlands. The sample variogram (fig. A.12) is calculated for 15 lag classes up to the median of all separating distances. The spherical theoretical model is fitted using Trust-Region reflective method without a nugget effect. The model is under-estimating the semi-variance for the first two lag classes, which could either hint on a nugget or suggest a different spatial model. This requires a detailed assessment of the dataset in any application. One needs to cross-validate at least a Matérn and a stable model, with and without nugget each, before making any decision. However, for this demonstration the variogram is sufficient.

The model was used to interpolate the sample on a 100x100 sized grid (fig. A.13). This grid size is used to decrease the calculation workload and hardware demand for demonstration purposes only. The grid is bounded by the bounding box of the input coordinates. This results in an irregular cell size of 27.85 m along the x-axis and 38.97 m along the y-axis. Further, one has to be aware, that anything estimated outside of the convex hull of the measurement locations (white points in fig. A.13) is extrapolated and should be not be further used.
While this example demonstrates the ease of usage of SciKit-GStat, as data sources can simply be exchanged, the application of geostatistics can be way more complicated. SciKit-GStat can help with easily approachable methods and algorithms, but the user still needs expert knowledge to estimate useful variograms and set meaningful hyper-parameters.

### A.9.2 Pancake Data

Using a photograph of a pancake for geostatistics was fun, but not only a joke. When I first saw the browning-pattern in the pan I was just curious if means of geostatistics work for this example as well. The application was easy and straightforward and I took literally the first photograph made. I find it striking how well the variogram estimation worked. I have no other geoscientific real world or even artificial data example at hand that yielded more textbook-like variograms than this pancake. Today, I would conclude that while a pancake is not a geoscientific phenomenon, the browning of the dough is largely driven by thermodynamic principles which are universally applicable. Thus, this *'artificial'* data set was great for development and has become my prime benchmark data set for geostatistical method development. I personally prefer artificial datasets over real world examples here, as sampling sizes and locations can be altered. With real world datasets I, personally, tend to focus too much on the system that the data actually represents and not the method development. On the other hand, generating a random field by *putting* a covariance structure represented by a specific variogram into the field and then reproducing the very same variogram from a sample of the field is not much of a surprise. In these use cases I found pancakes to be very useful.

To bake your own data, there are a few technical instructions, which should help to produce comparable pancakes. The photograph was taken with a Canon Powershot 540SX digital camera at 3267x2305 resolution. The camera position was as orthogonal as possible at about 60 cm height. The original image was re-scaled to 709x500 pixels by cubic interpolation and finally cropped to 500x500 pixels centered along the x-axis. To sample the pancake, 300 random pixel positions are chosen,

without replacement to form the array of coordinates. The red band value at these pixels form the corresponding observations array. The photograph is a PNG, thus the value range is of a unsigned 8-bit integer (0 <= *value* <= 255).
Finally, my pancake dough is very liquid (more like a Crêpe and less like an American pancake). From my experience, liquid dough and high temperatures (short time in the pan) are the key to spatially structured pancakes. I would expect a classic American pancake to be way more homogeneous browned. I use 500 g of flour, 2 medium sized eggs, about a half liter of milk, a bit of salt and about 50g sugar. Finally I add water to the dough until it is about as liquid as warm motor oil. Usually, that sums up at least to another half liter (of water). Maybe a bit more. To bake the random field, use oil, not butter. I could produce similar results with two different pans on two different stoves (a very old one and a new induction stove). My final advice is to archive only a digital copy of the pancake and eat the actual one with maple syrup.

### A.9.3 Maximum Likelihood fitting

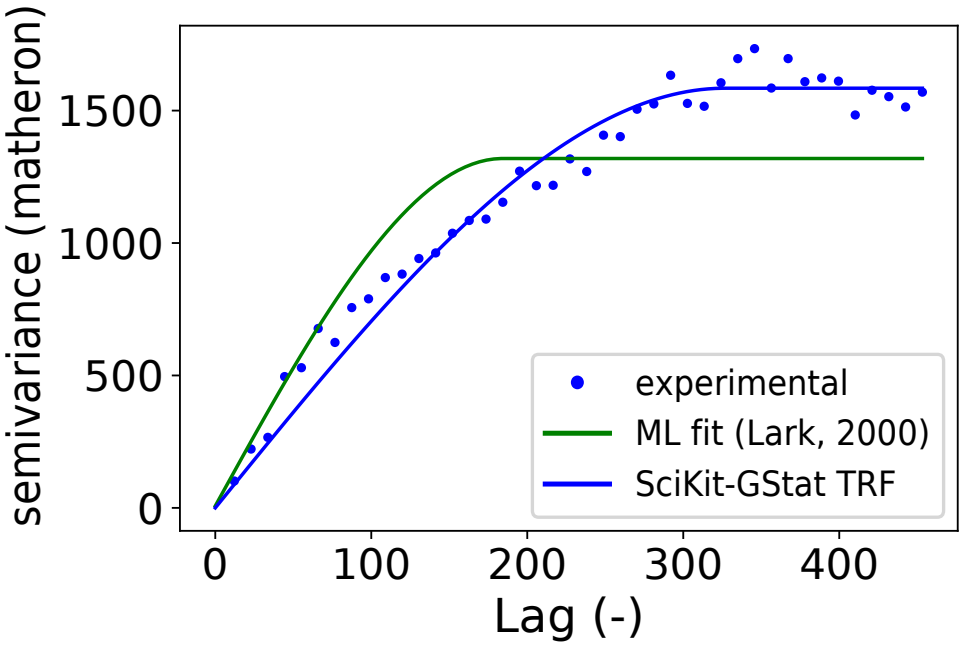



**Figure A.14:** Default SciKit-GStat Trust-Region reflective fit of the pancake dataset (blue line) to the experimental variogram (blue dots) compared to a maximum likelihood approach following Lark (2000) (green line).

With version 1.0 SciKit-GStat introduced a utility suite that can generate negative log-likelihood functions for any given `Variogram` instance. The definition of a negative log-likelihood function taken from Lark (2000, eq. 14). To construct this function, the utility suite is reading the distance matrix and the theoretical model type from the variogram instance at runtime and constructs an auto-correlation matrix as defined in eq. 9 of Lark (2000). The utility module covers all theoretical functions except the harmonized model, which can't be fitted.

This appendix briefly summarizes the tutorial introducing the utility function. A prime application for using this function is fitting a theoretical variogram model using a maximum likelihood approach (Lark 2000). SciKit-GStat does only return the likelihood function. It is designed to be used along with SciPy's minimization function (Virtanen et al. 2020) to find optimal variogram parameters by minimizing the negative log-likelihood of the model. The maximum likelihood fit is performed for the pancake sample as well (fig. A.14). For comparison, the default trust-region reflective fit is calculated for the same sample (blue line). In order to highlight a difference between both fits, the binning of the sample used here was changed to Scott's rule (table A.3). While the least-squares fit (blue line) follows the experimental variogram, the maximum likelihood fit does not involve any estimation of an experimental variogram. It covers the first few bins significantly better, but at the cost of sample variance, which is underestimated by the maximum likelihood fitted model's sill. From a technical point of view, the maximum likelihood approach should only be used for very small sample sizes. The least squares approaches implemented into SciKit-GStat are by magnitudes faster than minimizing a likelihood function. The computational demand is depending on the auto-correlation matrix for all sample points, which has to be inverted for each evaluation.

# B Geostat API - interoperable geostatistics on demand

## B.1 Abstract

I present a framework for combining reusable research software into replicable scientific analysis workflows. The technical details to conduct new workflows as well as the limitations of the framework are presented and discussed. The framework also introduces a specification of how input and output interfaces for research software can be implemented to increase the transparency of such tools. By separating parameters from data, the distinction between a reusable workflow, to reproduce existing results and replicable analysis to apply previously obtained research findings to new data in a different context is made more clear.
I revisit the analysis from chapter 2 to build such a workflow and present a robust analysis that consequently builds on the latter findings. Modularized research tools enabled the combination of different programming languages and entirely different libraries to easily test the suitability of force-directed graphs, a tool to examine and visualize relationships between entities in mathematical network graphs, for the use in the context of chapter 2. An original method is presented to visualize the covariance structure of representative variograms from the soil moisture observations in the Attert catchment in an intuitive and informative way. Each of these variograms was related back to different catchment states and physical processes in chapter 2, with respect to propagated observation uncertainties in the calculation of statistical dispersion functions. Using force-directed graphs, a few promising approaches to either interpret the graph itself or way how it forms with respect to the covariance of the related variogram are presented and discussed.

## B.2 introduction

This chapter serves two purposes. It is the synthesis of all preceding chapters and also the synthesis of all I have learned so far. An exemplary framework is presented that can combine arbitrary reusable research software into replicable analysis workflows. That can increase productivity and shifts the limits of how much information can be processed within the scope of a scientific research project. Even more importantly it shifts the limits of the tools that can be used and combined as due to its modular nature, previous analysis can be replicated.
The work presented in chapter 2 is implemented using the proposed framework. Limitation to the work as discussed in the respective chapter are systematically tested as a starting point to build on the previous work. Finally, the technical details of the framework are overcome to conduct new analysis and showcase, how a better implementation of the previous research software can advance new insights and

research.
One of the objectives in 2 was to present new ways how one can analyze, but also visualize recurrent spatial patterns in soil moisture. We used a spatial dispersion function to describe these patterns, which has not only proven to be a powerful statistical description, its also a handy visualization. For this chapter, an entirely different approach to visualize *spatial covariance* is taken.

A number of datasets have been presented throughout this work. Different methods were developed and applied to better understand the dynamics, patterns, or characteristics of the systems represented by these samples. To summarize, the utilization of modeling frameworks may provide a helpful stride towards reproducible research, given that they partition indivisible, monolithic user-developed scripts. This way, pre-, and postprocessing can be decoupled from the actual simulation, which enhances the reusability. At the same time, the model presents itself more transparently as inputs and outputs to the core algorithm need to be clearly defined within the modeling framework. Without a framework requiring the user to pass data and parameters to the building blocks of the framework, the researcher might more likely end up with a script that does not separate data preprocessing from data processing. Here, the preprocessing is linked to the specific dataset used and should ensure data quality, while the data processing is already part of the model itself. The core model algorithm is directly bound to the methodology and theory, the research is built upon, or even investigating. The preprocessing is setting the frame, how these methods need to be applied in a specific case, thus giving the theory a context for interpreting and discussing results. That makes preprocessing a prerequisite to research, not part of the research. Without a transparent definition of processing steps, it might remain unclear if, ie. scaling of particular data to a specific (spatial) grid is a peculiarity of the research question and/or catchment this script is built for, or an inherent processing step of the model itself, which might, in turn, limit use cases and/or applications.

When it comes to hydrological modeling, a number of attempts have been made to build modeling frameworks that can be applied in as many contexts as possible. An abstract description of model compartments or processes, that the model seeks to represent is common to most of these attempts. In a commentary, Weiler and Beven (2015) criticizes the lack of a common hydrological model, similar to the WRF model (Skamarock and Klemp 2008) widely used in meteorology. The commentary mentions several initiatives to build such a model and also some attempts to introduce a meta-modeling framework, ie. FUSE (Clark et al. 2008) or FARM (Euser et al. 2013). An even earlier attempt to establish a generic rainfall-runoff model, that I am aware of was published by Leavesley, Stannard, et al. (1995).

Despite all these efforts, none of these models and frameworks can be considered a common community standard and Weiler and Beven (2015) even note that the amount of different hydrological models and frameworks is increasing ever since. This suggests the prospect that the hydrological discipline has not yet agreed on a consensus on a common conception of hydrological processes in a catchment or that the attainment of such a consensus is impeded by the uniqueness of catchments (K. J. Beven 2000).

Besides more 'classic' conceptual or physically-based hydrological models, machine learning and more specifically, artificial neural network approaches are entering the hydrological modeling domain. Two recent reviews on machine-learning in hydrology (Nearing et al. 2021; Reichstein et al. 2019) illustrate the potential applications in hydrology, but also the sheer number of different network architectures. While artificial neural networks are implemented by only a very limited number of frameworks (compared to hydrological models), the implementation itself is still unguided and, following the existing literature, almost always bound very closely to the context of a case study. Thus, from a software engineering perspective in the context of this work, a distinction between 'classic' hydrological models and machine-learning approaches is not useful. In both cases, the inputs and outputs of an application are not standardized and the different compartments of the used model structure are not interoperable.

To enhance the modularization and interoperability of geostatistical preprocessing routines, the presented work collects the most crucial methods into a consolidated tool definition, with a special focus on the standardization of inputs and outputs and the interoperability of methods. Rather than introducing an additional framework for data processing or confining the tools to a particular pre-existing modeling framework, the tool definition is tailored to facilitate the development of application programming interfaces (APIs).

APIs are well suited to potentially enhance and formalize environmental models and the necessary data processing steps (Y.-D. Choi et al. 2021). Especially in 'classic' software engineering several approaches exist to standardize APIs to enhance communication between applications.

To exemplify the usefulness of such a framework, I seek to not only reproduce the findings of chapter 2, but extend the method. The clustering introduced in chapter 2 helped to aggregated spatial information under consideration of uncertainty. It was used to link spatial dependencies and how they emerge back to meteorological processes affecting soil moisture. At the same time, these spatial dispersion functions and their clustering represent a visualization of spatial dependence in

a dataset beyond the 'classic' variogram. This chapter will not only extend the technical implementation of the previous work, but actually use the implementation to develop a new approach for visualizing the spatial properties of the dataset.

The presented approach aims at visualizing the co-variance of the sample represented by a dispersion function, or variogram, directly. The variogram aggregates point-pairs at specific separating distances into a single statistical moment, called semi-variance. This is the expected value of the point pairs observation squared differences, representative of the separating lag class's median distance. Instead of aggregating the matrix of separating distances and squared differences for any given lag class, a network graph is used to visualize **i)** which points are actually connected at this lag (thus, form a point pair at all), and **ii)** how *far away* two points are in terms of observation value.

In a network graph, observation points of the dataset are represented by points, called nodes, and the connections between nodes are called edges. An edge is solely illustrated if the two connecting observation points fall within the separating distance under consideration. To visualize the squared difference associated with each edge, a force-directed graph is used, which will adjust the node position using the Fruchterman-Reingold algorithm (Fruchterman and Reingold 1991). This algorithm defines various emulated *forces*, which drive the movement of nodes until their position does not change anymore, or a maximum number of simulation iterations has been reached. These forces can relate to actual physical forces, like attraction between particles, but also implement any functional constraint that mutates the movement of nodes. Here, a rule is implemented to maintain a predetermined length for each node, which in turn accelerates the nodes to comply with this constraint.

Force-directed graphs a widely used to visualize and analyze social network datasets (Bannister et al. 2013; Itoh et al. 2009; Rahman and Karim 2016). All three named studies focus on methodological aspects of simulating and drawing force-directed graphs. Other applications have been found in biology (Danaci et al. 2018; Genc and Dogrusoz 2003), which used force-directed graphs to visualize biological signaling pathways, or enzyme pathways respectively. A combination of a biological application and a spatio-temporal interpretation of meteorological parameters was reported by Damos (2016). The authors used force-directed graphs to link observations in a time-series of air temperature and humidity to the population size of moles. The edges are weighted by the statistical significance of meteorological parameters for the population size, which means, these edges associated with the most significant parameters will dominate the others for the layout.
For the proposed method presented in this chapter, we weigh the edges by the

residual value of the respective point pair. To assess the usefulness of this approach the following hypotheses are tested:

- The force-directed graph of the two or more datasets are suitable to visualize the underlying co-variance and distinctly classify the respective sample.
- Modularization of research tools through containerization enhances the scientific re-usability and scales the application of research tools.

## B.3 Methods and Implementation

### B.3.1 Implementation

As of this writing, the Open Geospatial Consortium (OGC) is actively developing, verifying, and approving a new family of APIs for geospatial data distribution and processing, called OGC API 3.0 (Consortium 2019). These APIs build on the legacy of the OGC Web service standards (ie. WMS or WPS), which are well-established and accepted by all GIS systems. The new APIs will all be OpenAPI 3.0 compliant, which drastically increases the interoperability of implementations, even outside the GIS domain. In the context of this work, there are a number of relevant APIs, which can be used to implement geostatistical workflows.

The OGC API is furthermore quite appealing, as core parts of the API, including Features, Coverages and Processes are built on the legacy of their respective predecessors, the OGC Web service standard. These are well-established and accepted by all GIS systems and the corresponding OGC APIs will be downward compatible. Important is the web mapping service (WMS) (Michaelis and Ames 2008) for raster data, the web coverage service (WCS) for coverages, the web feature service (WFS) (Michaelis and Ames 2008) for vector data and the web processing service (WPS) (Michaelis and Ames 2008) for geospatial processing of data. To deploy these web services, a multitude of open-source and proprietary solutions exist. The Java software GeoServer and C software mapserver are the two most common open-source projects, while ArcGIS Server is the most common proprietary solution.
Due to the novel character of the implementation framework at the time of this writing, not all APIs are yet verified and/or the API code is not yet fully supporting all use cases stable enough for production. Thus, to exemplify possible implementations, data distribution can be realized using GeoServer due to the maturity of the software. In future implementations, either GeoServer might extend to OGC API itself, or the other frameworks will be ready for production to migrate the API.

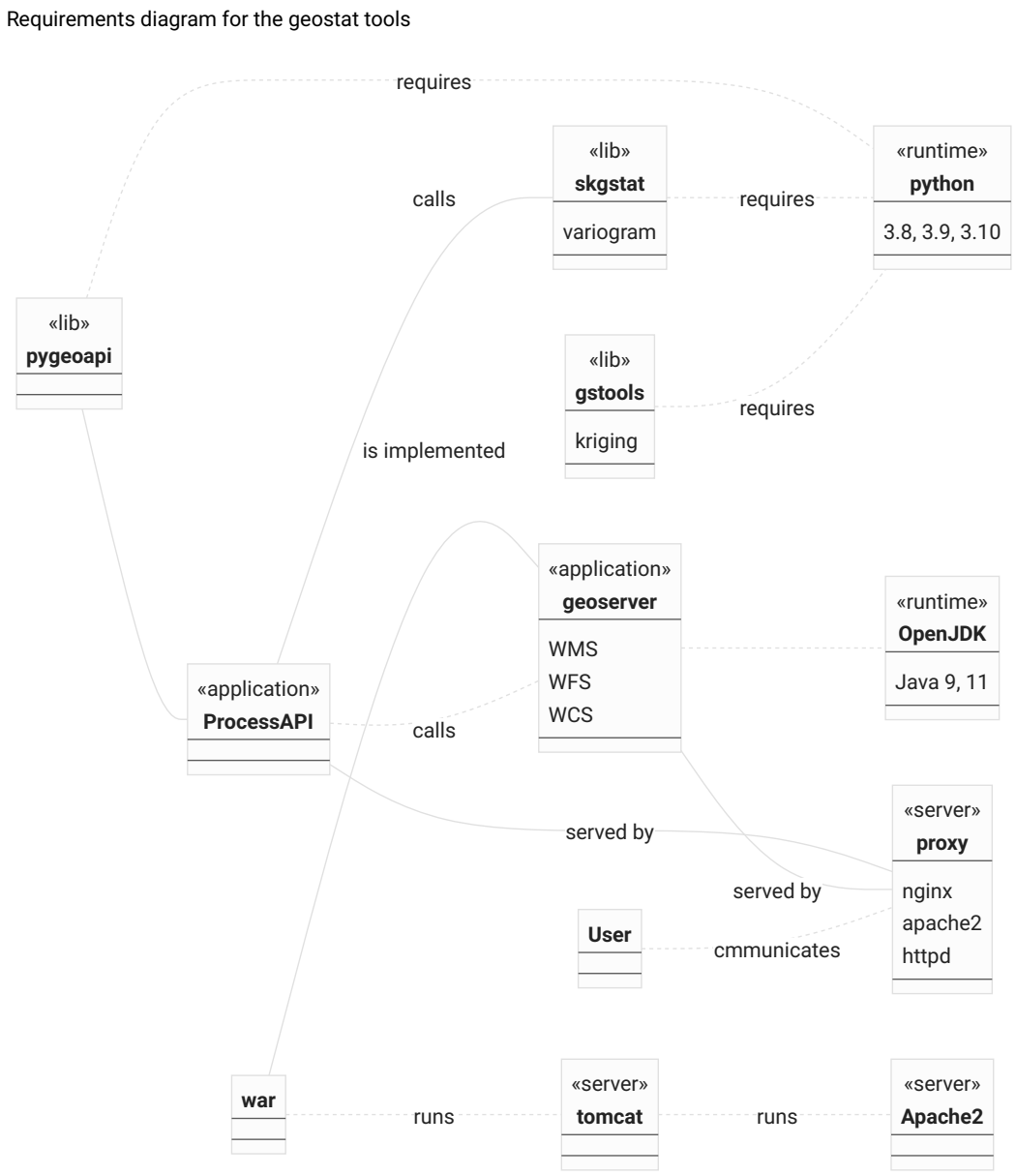



**Figure B.1:** Dependency graph for the OGC Processes API for system-wide implementation.

To implement geostatistical methods, the OGC Processes API can be used. A process can be any runnable computer program that consumes standardized input and returns standardized output. Each of these processes can describe its inputs using Swagger and OpenAPI, which offers interoperability to literally any client program. Data can be supplied directly by passing it in text-based formats like GeoJSON or CoverageJSON, or preferably by providing the URI of an OGC Features or Coverages API compatible endpoint, or legacy web services like WCS and WFS. This way the API methods can be applied to the accompanying datasets, or to any other geospatial dataset available through a web API.

A major downside of the infrastructure specified up to this point is the dependencies. An API like this needs different types of applications and runtimes present on the operating system in order to run (fig. B.1). The simplified dependency graph does not include system dependencies of the actual geostatistical libraries, like GEOS or PROJ4. The entities (boxes) on the graph represent different kinds of system architecture. The lines represent two different kinds of interactions between entities: the solid line represents implementations where one entity interacts with the other within the same environment, while the dashed lines represent requirements, which are the prerequisite for applications and libraries to be implemented. The graph involves two runtimes, Python to run the OGC API implementation and the tools, and OpenJDK to run the Geoserver application. As Geoserver is packed into a web archive, a server application running this archive is needed. This server is called Tomcat, which in turn needs an Apache2 web server to communicate with remote clients. The OGC Processes API can be implemented using the Python library `pygeoapi`. This is the backbone, which on the one hand operates the geostatistical Python libraries `GSTools` and `SciKit-GStat` and on the other hand requests data for the processes run from the Geoserver, based on the parameterization given by the user. An optional, but usually important, last building block is a Proxy server that operates as the single gate for information entering and leaving the stack, which can be served under a single domain. This is important as any communication between the server and the proxy should be encrypted by an SSL certificate. The Apache2 and proxy entity can be realized by physically the same server, but would still run two separate tasks.

There is no way around the described dependencies, in case all elements of the system are required. The approach and originality for this work, however, is to split the above architecture into modular, containerized units which are only dependent on a container-running service on the host system. In addition, a simple yet flexible specification of how these container modules communicate with the host system relaxes most of the issues arising from the aforementioned dependency

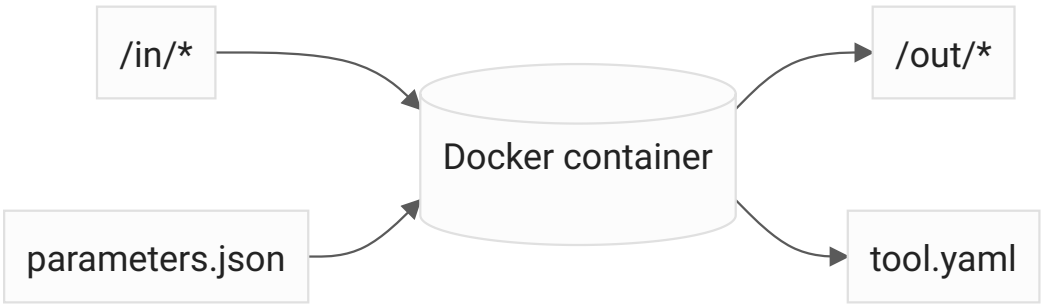



**Figure B.2:** Simplified information flow into a docker container, which implements the tool specification used for Geostat API.

diagram. The other main advantage is, that containers can be split by runtimes so that no part of the implementation interferes with the host system and multiple and even incompatible runtimes can be operated at the same time. Furthermore, the dependencies relevant to a single tool are packaged along with the tool itself and not with all dependencies of all entities that just arbitrarily happen to use the same runtime. This should make developments and maintenance straightforward. The most common software for building and running these kinds of containers is called docker. It is available for all major operating systems, widely accepted and containers built by docker can be run with other solutions as well. In order to run arbitrary tools inside a docker container, a specification was formulated, which will turn a container into a recognizable tool. This specification is created in a generic, implementation language-agnostic way and does not dependent on the presented framework implementation in any way. The specification is hosted publicly on Github, allowing community contributions (Mirko Mälicke and Dolich 2023b). The main requirements for building such a container are a set of files and the entry point of the container (fig. B.2). The container entry point has to be any runnable file without any runtime arguments. Otherwise, the state of the container would be dependent on the way it was called. Next, a YAML file has to be present at a specific location inside the container, which contains metadata like the tool description and the parameters, which the tool will accept. The metadata also specifies if more than one tool can be found in the container. Parameterizations for running a tool are collected in the parameterization file (parameters.json), which will be included in a specific input data folder, that is mounted from the host system to the container on runtime. This way, the parameterization, and the input data can be created by any kind of framework, editor, or script at runtime. For parameters that cannot be serialized into JSON or are simply too large, the parameterization file can link any file-based data source inside the input folder. To ease the loading

of these parameterizations into data structures of the respective implementation language, a number of software packages have been created. These can be installed into the tool container and read the parameterization, including standard data file formats like JSON, CSV, and netCDF. Libraries exist for Python, R, NodeJS, Octave and Matlab.

On the output side of a tool container, the scripts are required to store all results in a specific output location. This folder is again mounted to the host file system and will therefore persist results after the container has finished. The reason for this is, that it is common practice to remove containers by docker after they have finished. Even if the container is kept on the system, any user of the tool would be required to commit changes into a new docker layer, as the container does not store changes on exit automatically. Additionally to a result folder, it is recommended to bind standard output and standard error stream of the container to the host's streams. This way, one can capture any error and console output by the tool inside the container.

The aforementioned specifications for docker containers to recognize them as tool containers are framework agnostic. As long as a service executing containers, like `docker`[13], `singularity`[14] or `containerd`[15] is present, the tools can be used. However, the specification itself should be seen as standardization across platform-dependent client applications and thus the intended audience is developers. For the programming languages Python and NodeJS, I provide a client application to run the tool containers. First, the Python library `toolbox-runner` (Mirko Mälicke and Dolich 2023a) that consumes parameters and Python objects natively and translates everything to container parameterizations is implemented. This should make the tools way more usable for Python users. It also adds control flows like asynchronous and event-based execution options and implements a number of convenience methods to archive tool results.
The NodeJS library `@hydrocode/tool-runner` (Mirko Mälicke 2023a) provides similar functionality for a NodeJS environment. It focuses more on providing an HTTP-based API for building cloud and web applications for tool specification-enabled containers. An exemplary project[16], which makes use of this API is also developed, that translates tool specifications into a graphical user interface and can

**13** Docker. (n.d.). What is a Container? Retrieved March 30, 2023, from https://www.docker.com/resources/what-container

**14** Singularity. (n.d.). Singularity. Retrieved March 30, 2023, from https://sylabs.io/singularity/

**15** containerd. (n.d.). containerd. Retrieved March 30, 2023, from https://containerd.io/

**16** hydrocode-de. (n.d.). tool-runner-frontend. Retrieved March 30, 2023, from https://github.com/hydrocode-de/tool-runner-frontend

be compiled to a native desktop application for Windows, macOS, and Linux, or used as a basis for mobile applications for iOS and Android.

Another advantage of using frameworks instead of interacting with the docker containers directly is, that frameworks can collect error messages, log output, results, and runtime metadata in a unified way for all tools. In case the user decides to pull semantic versions of tools (ie. `1.3` instead of the docker-specific version `latest`, which always points to the newest version), the provided frameworks persist all inputs, all outputs, a checksum for the used container, and the exact version of the underlying image into a local archive. This enables the user in principle to pull the exact same image locally, mount the archived input data and parameterization, and run the container. After finishing, the newly created results and their checksum can be compared to the checksum in the archive as an example of repeatable tools. The same container can be used with substituted input data to run a replicable tool, as it was just verified that exactly the same tool was applied in exactly the same way, but to other data. To this end, this is an example of technical replicability.

To link back to the beginning and the exemplary OGC API implementation, which is finally built around the framework, not the tools. This separates the architecture from tool implementation while utilizing the interoperability of OGC APIs. Finally, it also leaves the implementation of a tool to a containerized and therefore isolated environment, that, once parameterization files are loaded, focuses only on the tool algorithm through a single executable script in literally any programming language. The specification abstracted all pre- and postprocessing away. This can turn especially useful in case user-defined scripts are used from third parties, like other scientists, that are not trained software engineers and prioitize algorithm implementation over technical implementation and workflow details.

Coming from chapter 2, the presented framework is used to connect directly to the discussion of that chapter. To assess the validity of the chosen approach, the analysis workflow can be run several times changing only one parameter in question at a time, similar to sensitivity analysis in modeling. To exemplify this, the main limitation (as I see it today) of the aforementioned work is focused.

The clustering of spatial dispersion functions was based on the distance between two empirical variograms. The idea of a variogram is to represent the spatial covariance structure of a sample by a suitable model and its parameterization. Thus, in principle, the results of chapter 2 should be repeatable by using the variogram parameters as input for the clustering tool, instead of the empirical variogram itself.

Another big advantage of the proposed framework is the ability to combine tools and software. A force-directed network graph (Fruchterman and Reingold 1991) tool is implemented using the aforementioned tool specification. As a novel approach to visualize the co-variance of a geospatial dataset, this tool is combined with the clustering results replicated from chapter 2, as described above. The force-directed network graph is used to visualize the distance matrix of all observation points and the associated squared observation differences at the same time. In general terms, force-directed graphs can be used to group entities, ie. in social networks to weigh connections between nodes by the relationship of the persons that are represented by the node (Bannister et al. 2013; Itoh et al. 2009; Rahman and Karim 2016). For the presented application, each node represents an observation location of a geostatistical dataset and the edges connect point pairs that are within the separating distance of the considered separating distance lag. This is a natural way to visualize, how the observation network influences the emergence of a covariance structure. Force-directed graphs introduce an additional dimension when compared to other mathematical network graphs, as an iterative simulation moves nodes like particles through a two- or three-dimensional canvas, following different rules that accelerate particles or constrain their movement. These rules are called *forces*, hence force-directed graph. Hence, the movement of the nodes and their final position, after the simulation terminated are related to the spatial co-variance of the dataset. The squared differences associated with each edge are used to dimension them, such that the nodes are moved to match the pre-determined length.

Here, I use this visualization tool to implement a *force* that seeks to keep the length of an edge at the squared observation difference of the respective point pair. This way, point pairs pull and push each other depending on the joint differences in observation. The final setting the nodes end up in when the simulation converges can be interpreted as the expected value of the joint distribution of the connected edges, which is simply the local covariance.

I hypothesize that a force-directed graph is a suitable tool to visualize the joint effect of the observation network and spatial covariance in a single graph. Further, visual distinctions between the network graphs of the average variogram for each cluster, or the emergence thereof, are expected.

### B.3.2 Data and Methods

The current state of the Python framework is used to demonstrate a geostatistical workflow and implement an analysis. Besides the scientific insight into how co-variance structures develop and are sustained over time in soil moisture samples,

another objective is to showcase the ease of usage and the analytical depth of the workflow.

The dataset used is the same as described in 2.3.1. No changes have been applied to the described data preprocessing, yet only the soil moisture time series recorded at an installation depth of 30cm are used. Following the description of 2.3.1, we end up with 57 profiles at 19 cluster locations within the Colpach catchment (fig. 2.1).

The first tool implemented into this workflow is a generalization of the dispersion functions (Mirko Mälicke 2022b) used to describe spatial statistical dispersion over time (see section 2.3.2). The implementation used in the original publication defined dispersion functions as a generalization of variogram functions to relax the intrinsic hypothesis. To implement a tool function, `SciKit-GStat` is used to calculate dispersion functions and is then applied to a moving window, which can be parameterized in terms of window size and shift. This way, any kind of variogram function supported by `SciKit-GStat` can be applied as a moving window aggregation function to any kind of moving window setup supported by `NumPy` (van der Walt et al. 2011). The user is encouraged to check the geostatistical validity of the chosen settings, but it is noted that `SciKit-GStat` also implements spatio-temporal variogram functions for windows that may need to respect temporal auto-correlation within the window itself. In the given example, the parameterization described in section 2.3.2 is reproduced, which used 30 days window sizes to estimate dispersion functions based on the Cressie-Hawkins estimator (Cressie and Hawkins 1980). A notable difference between the original analysis and the parameterization chosen for the workflow is the binning function. Here, the standard function of even lag class widths is used, instead of the variable bin widths to create uniform sample sizes within each lag class as described and motivated in section 2.3.2. The original publication repeated the analysis not only for different sensor installation depths but also for three different years. For this chapter, only the 2013 data is used.

To identify windows of similar spatial covariance structure between the observation locations, the analysis described in chapter 2 conducted a Mean shift clustering algorithm (see section 2.7.1). The tool implemented is generalized as a generic clustering tool built on `SciKit-Learn`, the most sophisticated Python implementation for clustering (Mirko Mälicke 2023b). The implemented tool can handle N-dimensional samples, including a transformation for 1D samples, which the underlying software is not capable of. Besides Mean Shift a number of other clustering algorithms are available: K-Means, Affinity propagation (Frey and Dueck 2007), and Agglomerative clustering (Pedregosa et al. 2011b), with four different methods to link sets into a cluster. For consistency, the Mean shift was used, which identified 5 clusters for 2013, of which two only contained very few members and

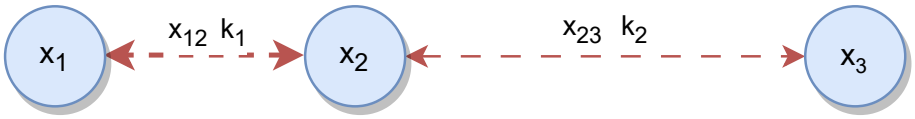



**Figure B.3:** Conceptual idea of representing spatial co-variance by a force-directed graph. The graph shows three nodes connected by two edges. The sketch illustrates how the squared observation differences between ie. $x_1$ and $x_2$ if set as the distance $x_{12}$ indirectly control the value of the spring constant $k1$, if interpreted as a physical spring.

are therefore neglected. For comparison, the tool was parameterized to identify 3 clusters using K-Means, which ended up in the identification of broadly the same clusters.

Up to this point, the main data processing from chapter 2 is repeated, using a more robust and consistent software implementation as described in chapter A. In the following, a new perspective on the spatial structure is introduced, which can be compared to the findings presented in chapter 2. Other than in the original analysis, the parameterization of the clustering tool was changed to cluster the dispersion function parameters over the empirical values. Interestingly, still, the same clusters are identified. For each of the clusters, the mean soil moisture was used to calculate a variogram. Beyond chapter 2 and its re-implementation, the separating distances and associated squared observation differences for all point pairs within the effective range of the respective variogram are further analyzed. The final main tool for this workflow is intended to visualize the covariance structure based on this data directly.

To visualize the covariance of any given variogram instance, a network graph is used. Each observation location is represented by a network node, however, the position is not derived from the spatial coordinates (the measurement location), but simulated by a verlet integration of a position and velocity matrix representing the network nodes. The implementation is taken from an existing Python library (Hagberg et al. 2008). The implemented numerical integrator is simplified by a number of assumptions to make the simulation of large networks feasible. A constant unit time step for the simulation is assumed, which does not impact the proposed analysis, as only the network after the simulation converged into stable positions is used. Second, all nodes in the network are represented by particles of the same mass. The simulation implements an abstract concept of forces, which in this context are any kind of transformation function, changing the particles' velocities, or locations directly. Thus, a simulated force can apply a physical force like gravity, but can also apply constraints, like preventing particles from leaving a certain

area. When referring to a *force* in this context, the implementation function of the simulation is meant, not necessarily the physical concept. The used simulation activated various forces. First, all particles move towards the shared center of weight. The main reason is to keep nodes together and can be thought of as a joint center of mass. The second kind of force is applied along the network edges. If they are supplied with a value, the force function will accelerate the particle along its edges to reach the desired edge lengths. These values default to 1 if no data is supplied. The implementation of the tool uses pairwise observation differences to connect all pairs within a given range of lag classes. Effectively, this turns the pairs within a lag class into springs connecting the observation locations and their observation differences into the spring constant.
Consider three exemplary nodes $x_1, x_2, x_3$ (fig. B.3). The respective distances $x_{12}, x_{23}$ are set to the squared observation distances, and the simulation is run until the velocities of the nodes relative to each other have converged. Using Hook's law, the ratio of the spring constants can be related to the two distances, as long as the nodes do not move to relative to each other:

$$\begin{aligned} F &= k\Delta x \\ k_1 x_{12} &= k_2 x_{23} \\ \frac{k_1}{k_2} &= \frac{x_{23}}{x_{12}} \end{aligned} \tag{B.1}$$

Where $F$ is the force applied, $k$ is the spring constant and $x$ is the distance between two nodes. Given that $x_{12}$ is half the distance $x_{23}$ (fig. B.3), the associated spring constant $k_1$ needs to be $2k_2$. Thus, observation point pairs with small squared differences are not pulled closer together, one would also need a larger force to push them apart. This is a very intuitive illustration of co-variance in a geostatistical dataset. Given an interactive version of a force-directed graph, it is even possible to actively *push* nodes away from their stable position and investigate on the effect on the network graph and the movement velocity, when the simulation accelerates the nodes to match the edge length constraints again. The simulation will stop as soon as the particle velocities are smaller than a specified threshold or a maximum number of iterations is reached. This threshold is set to 5000 here. The resulting state of the network is interpreted as a visualization of the sample's covariance structure.

To better assess the network graphs of a variogram, different template variograms are calculated to visualize their covariance structure as a benchmark of how these graphs can look like. Then, the actual variogram can be located on the spectrum of possible network manifestations. The first variogram is a pure nugget effect

variogram, as calculated on a sample of random coordinates and purely random observations. With a sample large enough, no spatial correlation is expected, thus the network nodes are not expected to exhibit any specific order and the driving force for layout is the movement towards the center coordinate of the sample. The final layout is then expected to be a fairly 'small' circle, as within the circle the network can arrange to contain all, normally distributed, pairwise squared differences in a compact way. As the variograms in question have rather small sample sizes, a random sample of the same sample size is created. The deviation of its covariance network graph to a circle is interpreted as inherent uncertainty to the method, which is attributed to the sample size, rather than the spatial configuration of each cluster variogram.
The second benchmark variogram is calculated for a highly structured field. Let $M$ be a 2D regular grid defined as:

$$M_{i,j} = i + j \tag{B.2}$$

Where $i$ is the matrix row index and $j$ is the matrix column index. This field is highly structured, as the field values are ordered. It is also deterministic, as for any location on the grid, the whole field can be derived, solely from the single value at that location. The variance of the field is increasing with the size of the grid, in principle unbounded.
Finally, a second, highly structured field is created. Other than the grid described in equation (B.2), the second field is defined as:

$$S_{i,j} = 1 \tag{B.3}$$

Where $i$ is the matrix row index and $j$ is the matrix column index. Similar to $M$, the field is structured, as it is ordered and deterministic. In contrast to $M$, the variance is always 0. A variogram sampled from this field is also a pure nugget effect variogram, but in contrast to the random sample, the variance is 0 and the nugget effect variogram will be located at the x-axis. These two cases help to differentiate the effect of non-spatial variability on the overall network layout.
With these benchmark variograms calculated for the sample size of the spatial data in question, the actual covariance graph of the sample can be assessed in a field structure versus field variance spectrum.

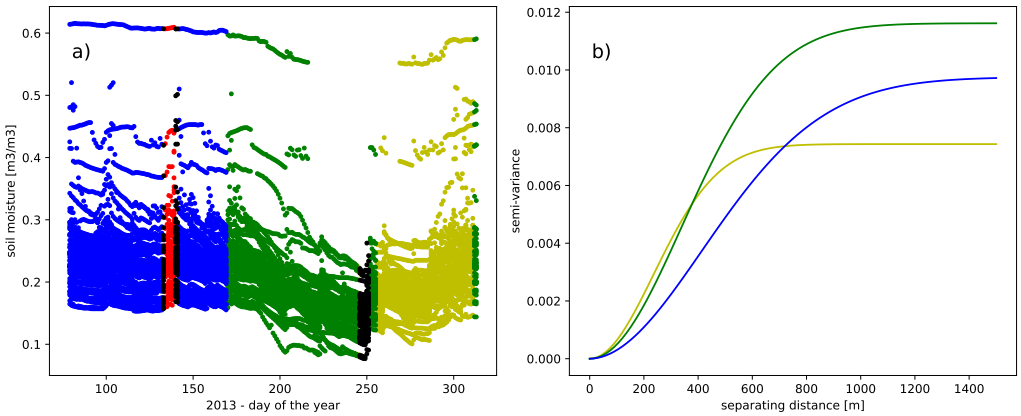



**Figure B.4:** Result for reproducing the results of (M. Mälicke et al. 2020), with variogram models fitted to the aggregated, clustered variogram parameters (in contrast to clustered empirical variograms in the original publication). **a)** Soil moisture in the Colpach catchment in 2013, the time series are colored according to the clustering result for variogram parameters. **b)**: The fitted variogram models for each of the clusters. Each of these models is the aggregation of the dispersion functions within each cluster and not fitted to empirical variograms of the observations.

# B.4 Results for reproducing a published workflow

## B.4.1 Reproducible workflows

Using containerized, reproducible workflows put one in the position to run extensive analysis systematically. This has an impact on vastly decreased processing times and also clearly communicates parameterizations and data used in the workflow in standalone files. Large parts of the analysis presented in chapter 2 were reproduced (fig. B.4 a) in two containerized tools, first to apply dispersion functions in a moving window, then a generic clustering tool, which includes the Mean shift algorithm used in the original publication (fig. B.4 b). In the second step, parameterizations have been changed in an iterative process and the actual analysis result could be used as an objective function, to explore how sensitive the result is on the parameterization. This application already is way beyond the scope of the original application, in which only two window sizes could be tested due to technical limitations.

The soil moisture time series observed in the Colpach catchment from 2013 is shown in figure B.4 a). In line with the methodology of chapter 2, each observation is colored according to the clustering result. The identified clusters exhibit temporal

coherence, with similar clusters occurring consecutively in time. Unlike the original chapter, the clusters were generated based on variogram parameters computed for each moving window. Despite this modification, the proposed method yielded highly similar results. In order to facilitate a more comprehensive comparison of the three clusters, an illustrative variogram based on the mean parameters of each cluster is presented in Figure B.4b).

The exemplary hypothesis that can be answered at this point is: The results from chapter 2 can be reproduced by clustering the variogram parameters, instead of the empirical variogram (see fig. B.4 a).
As the parameters describe a theoretical variogram model that represents the spatial properties of the sample captured by the empirical variogram, differences are expected to be imposed on the variogram parameters as well. While the clustering itself worked as expected, as a second step the variogram parameters for each of the three main clusters, blue, green, and yellow, are aggregated in cluster means (fig. B.4 b). These variogram models are provided to visualize the differences in spatial covariance structure. The cluster variogram models differ in sill and effective range parameter, which fits the results and conclusions of M. Mälicke et al. (2020) for the dispersion functions. The remaining chapter will use the underlying data (for both variogram models and dispersion functions) to introduce a novel way to visualize the described spatial co-variance.

### B.4.2 Visualization beyond variograms

The presented workflow loosened the restriction of implementing a scientific analysis workflow in a single programming language. The interfaces of the two tools are clearly defined and the actual code is containerized. For the presented workflow, one can easily switch from Python, which was used so far, to other languages. A language particularly interesting at this point is Javascript. Due to its tight relation to web browsers, it is a highly interactive language, which is designed to react to user input asynchronously and mutate existing HTML elements as functions return. This is uncommon for (geo-)scientific workflows, which usually run code in a static way that transforms inputs into outputs in a transparent manner. For data exploration, especially without the need to generate results, a more interactive yet in-transparent language like Javascript has advantages.

With the containerized tools in place, combining both approaches is straightforward. A lightweight Javascript application was written (Mirko Mälicke 2023c), that reads generic network graph definitions and outputs them as force-directed diagrams in a browser window. By binding most parameters of the underlying

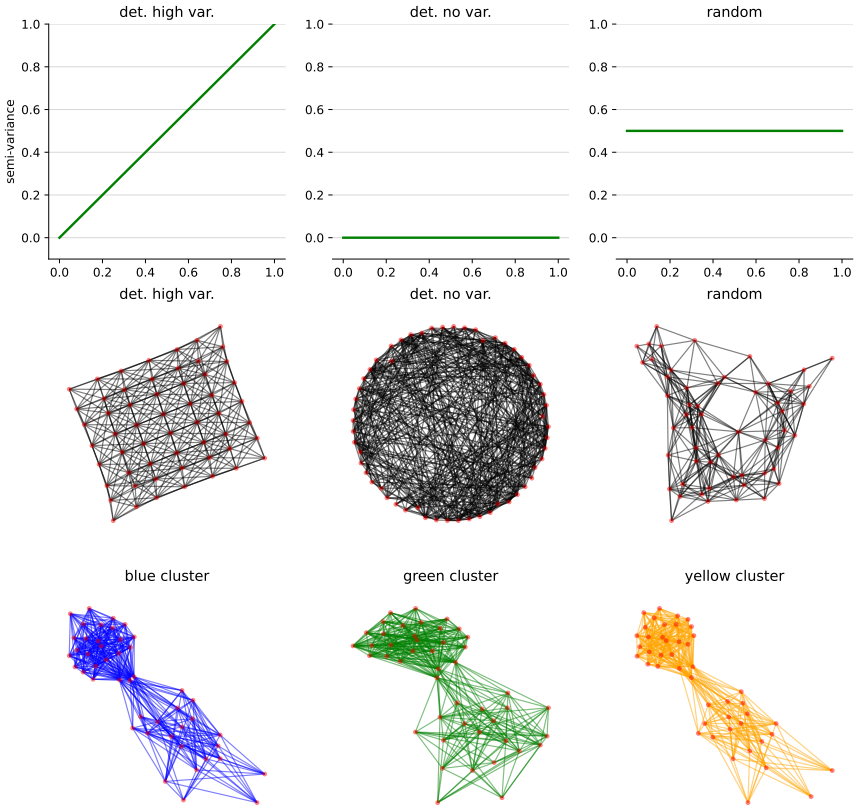



**Figure B.5:** The upper row shows conceptualized variograms for the three benchmark datasets, where *random* is based on a random variogram, and the other two on a deterministic field with high variance (*de. high var.*), and the no variance (*det. no var.*). The center row illustrates the force-directed diagram of the three benchmark datasets. The bottom row shows force-directed diagrams for the three spatial dispersion parameter clusters, denoted blue, green, and yellow cluster. For all network graphs, only the first three distance lag classes of the respective empirical variograms were taken into account. This roughly reflects the effective range of the variograms.

library to HTML elements, the graph can be explored in an extremely interactive way. This step allowed exploring force-directed graphs and their suitability to visualize covariance structures of an empirical variogram instance. One could very rapidly re-run the full workflow to generate new graph visualizations, but also inject network graph data originating from different sources. Three *benchmark* variograms were generated following B.3.2, a) a purely random variogram and two sampled from a regular mesh grid with b) large and c) no variance. The three benchmark datasets present three distinctly different force-directed graphs (fig. B.5, center row). For reference, a conceptual variogram model representing the spatial covariance of each benchmark dataset is shown as well (fig. B.5 top row). Both illustrations depict the spatial covariance from different perspectives: the variogram in a functional condensed form, and the force-directed graph to reveal *structure* as visually as possible. The graph of the deterministic mesh grid with high variance and trend presents itself as a simulacrum of the mesh grid itself. As the high variance of the field is generated by adding the matrix indices, the squared difference of any point pair is reflecting their distance exactly. Thus, the native way of organizing these differences is the mesh grid itself.
One insight the force-directed graphs revealed for the clusters (fig B.5 bottom row) is the observation network design. The network is split into two parts, connected only by three nodes. Each node represents an observation location. These locations are well-connected to all other points, indicating a central location in the network, but all three nodes also keep the two sub-networks at fairly medium distances, meaning they are the most representative observation locations for both sub-networks simultaneously. In case one needs to dismantle a dense observation network, but wants to keep a reduced amount of sensors active, the force-directed graph is most helpful to identify suitable candidates.

Note, that the Javascript-based version was interactive, thus dragging network nodes or watching the initial arrangement of the nodes cannot be depicted in figure B.5. But the interactive nature of the graph unearthed different *behavior* of the networks. First, the time until the graph converged and node positions did not change substantially anymore differed for the force-directed graphs (fig. B.5). The particle velocity for a force-directed diagram is only driven by the forces applied, which in turn only depends on the required length of an edge that is set to be the squared observation difference of the respective point pair. In case the force-directed diagram converges slowly, there are significant interfering forces, that draw a particle in different directions at the same time. This can be thought of as high friction within the network.
Secondly, the interactive nature of the Javascript driven web-page allowed dragging

particles interactively. That basically stretches or bends the edges and effectively temporarily changes the squared difference of the respective point pair, by changing the position of a node. The force-directed graph will run another simulation to get the nodes and edges back into their equilibrium position. Here, it is worth mentioning, that these positions are not necessarily the same as the positions the nodes were dragged from. The simulation runs again and can end up in a different state. This is an illustration of equifinality, as in these cases it is possible to arrange the nodes in more than one way. Directly connected to the first observation, this *dragging* resulted in faster and slower responses of the network, which directly relates to the aforementioned *friction*. The random graph could easily be dragged away and almost instantly moved back into position. The same-value mesh grid, in which all edges seek to be of the same length, not only results in a very structured graph, but the graph reacted slowly to changes, hence there is a lot of *friction* within the network. More random variations in the covariance structure allow for more flexible positioning of nodes and faster convergence, as there is more variety in edge lengths. In turn, this implies that the time until convergence and the final stability of the network can be used as a proxy for the degree of organization in the spatial covariance. The stability of the network is here defined to be the average particle movement that still occurs after the graph converged or the simulation was ended due to a parameterized time limit. In general terms, I hypothesize that convergence time is negatively correlated with randomness, while network stability is correlated to equifinality in the network arrangement. The latter has been observed in the limited use cases presented especially for highly structured, deterministic fields.

These findings resulted in the development of a Python integration of force-directed networks as described in section B.3.2. Python is preferred here, as the reproducibility of the workflow is focused on instead of explorative interactivity. The Python-driven containerized tool can be run in parallel with the Javascript tool.
The simulations were run again and in general terms, the final, converged networks looked the same as for the Javascript tool. In addition to simply reproducing the results, the Python tool built on the implementation in NetworkX (Hagberg et al. 2008) to allow for intercepting the simulation on each iteration, which would be fairly difficult to do in the Javascript version. By tracking the positions of the nodes on every second iteration of the simulation, relative velocities for each node were calculated (fig. B.6). The thin lines describe the movement of single nodes, while the thick line is the network average. Note that all calculated velocities have been smoothed by a moving average of window size 2, mainly for visual reasons.

Figure B.6 is used to visually describe characteristic *behavior* of the networks

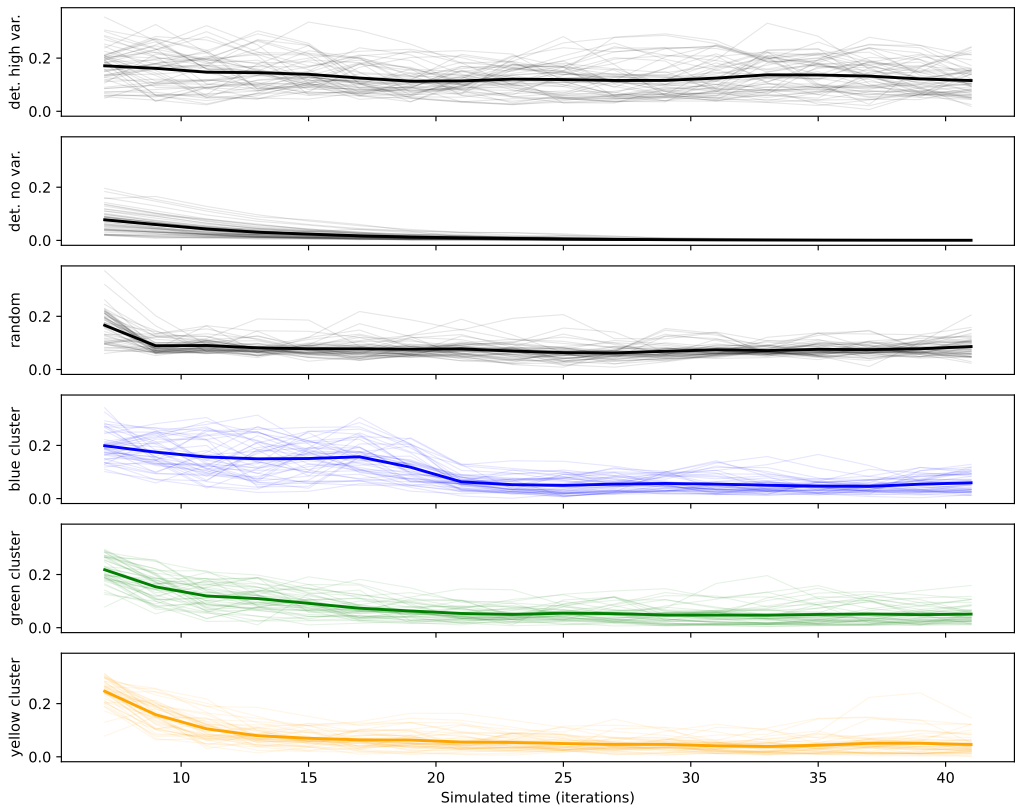



**Figure B.6:** Network convergence of the force-directed graph simulation for the networks presented in figure B.5. The thin lines show the relative velocity per iteration for all nodes of the graph and the thick line is the average velocity of all nodes combined. The names on the y-axis identify the same networks as shown in figure B.5.

| Name | Initial velocity | Convergence at it. | Velocity at con. | Stddev at con. |
|---|---|---|---|---|
| det. high var. | 0.171 | 16 | 0.132 | 0.066 |
| det. no var. | 0.077 | 34 | 0.000 | 0.000 |
| random | 0.166 | 6 | 0.078 | 0.035 |
| blue cluster | 0.199 | 22 | 0.043 | 0.031 |
| green cluster | 0.218 | 20 | 0.049 | 0.029 |
| yellow cluster | 0.247 | 28 | 0.045 | 0.037 |

**Table B.1:** Convergence signatures for the force-directed graphs shown in fig. B.5.

during the simulation. Mainly the initial velocity, the iterations until convergence, the remaining speed after convergence (base level), and the rate of decrease in velocity until the velocity converges are focused. The combination of these characteristics are summarized as *convergence signatures*.
The three benchmark networks show distinct convergence signatures (fig. B.6). The force-directed graph for the deterministic field with high variance (1. row), shows high initial velocity, which does not drop significantly. It moves into position at a constant speed and then continuously keep changing positions. When turning the simulation frames into a video, this movement looks like flickering resulting from nodes switching positions. While the network keeps its mesh grid structure, the nodes are interchangeable, because the graph is based on (repeating) squared observation differences, not geographic coordinates. This is an interpretation of equifinality, manifesting as a base level of movement, that is almost as high as the initial velocity. From figure B.6, a drop in initial velocity cannot easily be captured. Thus, the simulation time was forced to the 10-fold of the first run, preventing the simulation to stop. The results for the deterministic field with high variance (1. row) better show the drop from initial velocity and the high base level after convergence, which is reached after 16 iterations.

The other deterministic field shows a distinctly different convergence signature. The mean velocity is practically zero. With 34 iterations, it also took the longest time to converge. The nodes all show a monotonic decline in velocity and also keep their relative rank compared to the other nodes while converging. The two other benchmark datasets show a constant change in velocities for all particles for the whole simulation. After convergence, the node velocities and their standard deviation are negligible, meaning there is no movement in the network anymore. The force-directed graph for the random sample converges faster than any other graph calculated. This also holds for other realizations of random graphs. The average node velocity after convergence is larger but does not reach the level of the high variance network. In addition, the rate at which the velocities change is changing as well. While the deterministic, high variance graph showed a general, persistent high movement, the random sample network has periods of comparatively high movement, but also periods of smaller average movement. This can better be seen in the longer simulation (fig. B.7).

The three force-directed network graphs built from the residual matrix of the soil moisture cluster variograms exhibit only slight differences (fig.B.5). Their convergence signature, however, is distinct. A similar network graph is expected as all three cluster variograms share the exact same spatial positions and therefore also the same distance matrix. The distance matrix is directly related to the point

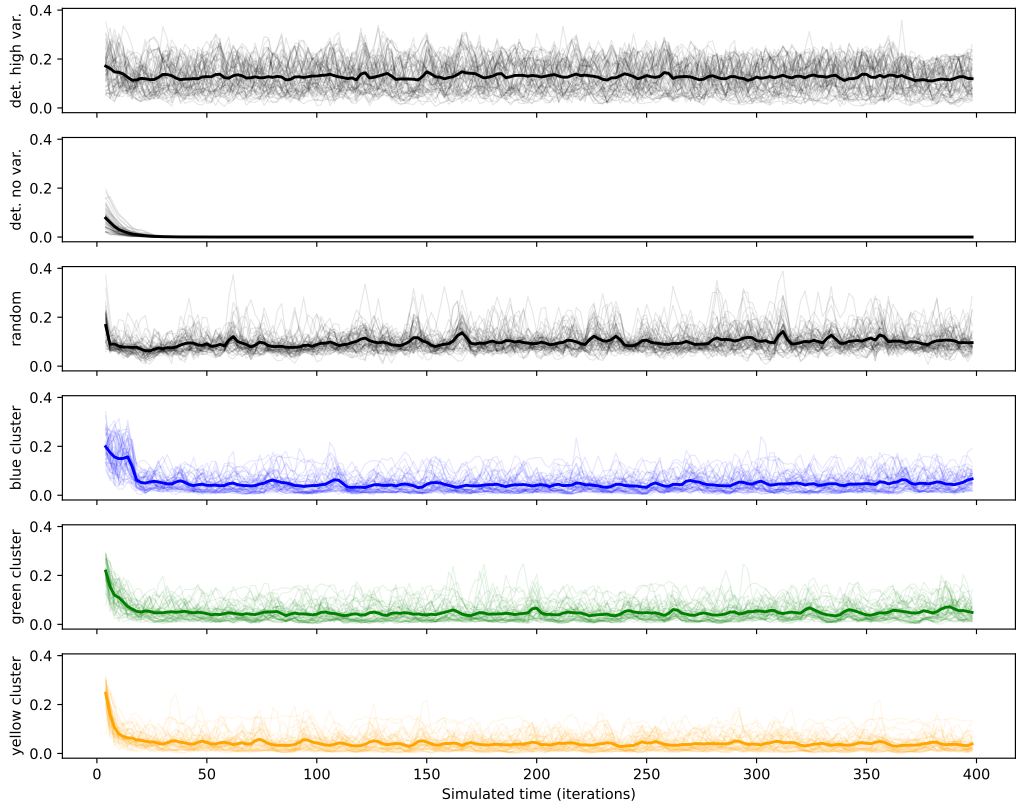



**Figure B.7:** Network convergence of the force-directed graph simulation for the networks presented in figure B.5 for **200 iterations**. The thin lines show the relative velocity per iteration for all nodes of the graph and the thick line is the average velocity of all nodes combined. The names on the y-axis identify the same networks as shown in figure B.5.

pairs that will be used as a basis for the force-directed graph. Thus, the networks share the same nodes and edges and only differ in the desired edge lengths.

The blue cluster force-directed graph shows the signatures of two different benchmark datasets. The early phase of the simulation shows a deterministic, high-variance signature, but the graph then converges after 22 iterations. After convergence, there is a fair amount of the random signature, as one would expect in a real geospatial sample. From the visualization of the force-directed graph after convergence (fig. B.5, blue graph) one can see two sub-graphs that are only connected by three points. This setup results from the observation network used. The similarity to the deterministic high variance graph can also be seen for the upper sub-graph. The blue cluster formed at the end of a rainy spring and ended with the beginning of a long dry summer.
Then, the green cluster formed for the dispersion functions of drying and then dry soil. The convergence signature of the green cluster shows way less similarity to the deterministic, high variance signature, compared to the blue one (fig. B.6). The signature visually presents as a union of a random and deterministic, no-variance signature. The soil moisture observations got closer as the soil was drying towards residual soil moisture. This signature can be seen in the convergence of the force-directed graph, although the observation network was not arranged on a regular grid (like the mesh grid for the benchmark graph).
The yellow force-directed graph exhibits less similarity to the deterministic, high-variance graph (fig. B.5). The upper sub-graph has a more rounded shape, although allowing for more nodes to position inside the blob. This shape cannot be directly linked to one of the benchmark graphs, yet it might be a combination of the no variance and random graphs. The yellow cluster formed with the first heavy rainfall reaching the subsurface after a long dry summer and lasted way shorter than the other two clusters (in terms of time). The graph could still reflect the observation of near residual soil moisture, overlayed by increasing random components, as rainfall sets it. In terms of convergence signatures, the yellow signature presents like an overlay of the random and the deterministic, no-variance signatures.

## B.5 Discussion

### B.5.1 Tool Framework

I presented a specification for replicable software workflows and implementations for the Python and NodeJS programming languages. To exemplify the usefulness of the approach, an existing methodology published a couple of years ago was used to

rerun the original analysis and replicate the approach in a different setting. Finally, it is used to develop a new visualization and analysis tool for spatial covariance described through empirical variograms.

The main abstraction of the approach is containerization. This entails two main improvements over 'classic' scientific analysis workflows and research software implementations equally.

Tools are containerized and thus, also modular. That means, an entity is created for each tool, which represents a specific step in an analysis workflow. The container content is defined to be the context of this processing step. From a software perspective, the container contains all the necessary dependencies a tool needs to run. Besides rather high-level dependencies like other software libraries the tool imports in the implementation language, this also includes dependencies and prerequisites like the implementation language itself down to the operating system level. The advantage is, that the state of the container is not dependent on the host system. Hence, all software relevant to the tool is contained in the container, but also not more. In more 'classic' scientific workflows, all dependencies for all tools needed in a specific workflow, and usually way more dependencies are present in the host system. That can make it particularly complicated for the user to identify the exact processing context of a tool. One might miss dependencies, run non-supported or conflicting versions, run code, which was mutated due to yet another workflow conducted on the host system or might even rely on data, that is not related to the research at all.

A prime example is the Python programming language, which is widely used in geoscience. It is common practice to host more than one Python interpreter on the system and have software libraries available, which are shared between workflows. Further, many libraries used in a scientific context are actually not implemented in Python, but in FORTRAN or C/C++. These libraries usually have system dependencies, like compilers, to compile sources either at installation or even runtime. A FORTRAN compiler is shared across resources, and thus an update of any system component can be cascaded to a newer compiler version, which might then affect unrelated scientific code.
Another example are geospatial libraries like GDAL or GEOS, which include data about coordinate reference systems. These libraries are shared system-wide and updates, ie. in a GIS system might change the data (about reference systems) in an unrelated Python workflow that depends on a library that wraps the system-wide libraries. This can turn into a non-trivial dependency conflict, especially in case the spatial functions of that Python library is not used in the workflow at all. A

Docker container bypasses these problems, as each tool container would ship with its own compiler and its own copy of system-wide libraries.

The other main improvement of containerization is contextualization, which is directly bound to the software perspective on containerization given above. The geoscience community does neither have a standard programming language, nor standard analysis tools. For almost any processing step, at least two solutions exist. Containers are interchangeable. If one moves from one software to another solution, another interface is used, as both software solutions are unrelated and may not be standardized or rely on interfaces. The tool specification takes on this aspect. It provides a clear, human, and machine-readable description of the container's interface. Communicating an interface is already a huge step forward, despite different implementations would still be necessary. Thus, the tool specification is also a building block for generating tool-related interfaces. As an example, it could be used to generate a common interface for using analysis tools for digital elevation models. Then, a tool containing GRASS GIS could easily be interchanged with another container using whitebox GIS and the parameterization of the tool would stay the same. At the current state, the tool specification does not intend to solve this but is aiming at providing a common ground to implement interfaces like that.

Through the aforementioned contextualization, the proposed tool specification and implementation frameworks can be used to let scientists work more effectively together. As every tool ships with its own context, the actual processes that transform input parameterization and data into the tool output are not exposed and do not interfere with other system components. Thus, two tools can be implemented in two different contexts. An example can be the chosen programming language. Rainfall data for a given catchment could be downloaded and processed using the statistical language R, which is widely used among hydrologists. The subsequent tool, which may use a neural network to predict discharge from the processed rainfall data can be implemented in Python, which has become the de-facto industry standard for machine learning.
Another example of differing contexts is programming paradigms. In cases where monolithic software stacks need to be created to run a processing toolbox for scientific workflows, many different tools need to be unified into a single code base. Ensuring the maintainability of such software would imply transforming code into a single coding style. In a scientific context, that implies that many tools are effectively re-implemented in cases where the final software is ie. following an object-oriented approach, but tools are delivered following imperative, procedural, or functional approaches. This is an issue as it is obviously a lot of unnecessary

work. More important is, that a specific paradigm might have been chosen for a good reason and a transformation might entail issues that are only caused by the transformation.
Finally, (geo-)scientists are not trained in software development. Thus, collaborative scientific workflows combine wildly different code in terms of style and paradigm, but also code that has been written at almost any level of professionalism. The proposed tool specification and implementation frameworks abstract all these issues away into the container and the context it represents. The scientist just has to define the parameterization of the analysis and the data needed in a clear and transparent standardized file.

A scientific processing workflow like describes also implies, that the proposed specification and frameworks do not necessarily improve the quality of the code of individual processing steps. It improves the interoperability of individual steps by increasing technical repeatability. It helps to use all the code at once, without the need to teach scientists proper software engineering or rewriting all the code in a single language. The underlying algorithms used can still be wrong or used in the wrong way. Both issues cannot be solved by the proposed specification. The tool specification is an attempt to unify wildly different software codes in order to be able to build frameworks that might point out *scientific* errors in the workflow. Another possible application is a framework that could estimate propagated uncertainties for all steps of a workflow. The tool container can easily be run in a Monte Carlo approach.

### B.5.2 Force-directed graphs

Force-directed graphs are to my knowledge an innovative and original approach to visualizing the covariance of a variogram function. The visualization can be parameterized to include the full distance- and squared difference matrix, filtered by a maximum separating distance, or include a single lag class only. This complements the 'classic' visualization of variogram functions quite well as it gains insight into the development of the covariance beyond the aggregated information of a semi-variance value for a particular lag class.

This chapter re-created the result of chapter 2, which linked the clustering of empirical dispersion functions to meteorological processes driving the wetting and drying of the observed soil. The hypothesis here was that a force-directed graph of each cluster is distinct, which was found to be true. While the converged graphs presented similarly, the properties of the converging graphs were different. The yellow cluster converged slower and showed overall higher 'friction' in the

network, which refers to the ability of the network simulation to find a stable configuration for the nodes, that satisfies the specified desired edge sizes. Following the argumentation of chapter 2, that can be explained by the same processes taking place, as the soil is already dry during that time and the pairwise squared observation differences are expected to be small.

The three benchmark graphs have proven helpful to classify any given force-directed network graph on the spectrum of possible graphs. In this work, randomness, sample variance, and field determinism were focused on. There might be more statistical field properties of interest, which can be translated into a benchmark force-directed graph to assess variogram co-variances. The three analyzed properties were chosen, as differences are expected in these regards. A necessary extension of the proposed method is to apply the analysis to other datasets and compare the graphs across datasets. The three clusters focused describe all the same field, but during different seasons. I expect all three network graphs to be distinctly different from other graphs representing other datasets or processes. To better assess the limits of the proposed method, a systematic comparative study has to be conducted. I hypothesize that the force-directed network graph of a rainfall field during the spring season in the Attert catchment at the time of study is different from the presented three clusters. At the same time, I expect it to be more similar to the cluster graphs than ie. the force-directed graph of a pancake.

## B.6 Conclusions

My main conclusions for this chapter are as follows:

1. Containerization of individual scientific analysis tools solved implementation and dependency issues, that otherwise impose an overly complex implementation for generic tools.

2. A contextualization of research tools clearly separates individual tools and transparently communicates interfaces of tools. Scientifically, these interfaces represent the context and limits at which a specific parameterization of a tool is valid.

3. The implementation of the proposed containerization and tool specification allowed for rapid reproduction of research, an extension of the workload by magnitudes, and a straightforward combination of tools, which are technically challenging to combine.

4. Force-directed network graphs presented as a useful visualization and analysis tool for co-variance structures described by a variogram. The potential to base variogram classification algorithms on these graphs was clearly illustrated.

# C Conclusions & Outlook

This thesis covered various distinct but interconnected areas of research. I contributed to the advancement of geostatistics by combining clustering approaches to the emergence of spatial dispersion functions. For the Attert catchment, evidence for two fundamentally different states within the catchment was presented and could be linked back to dominating processes. By combining information theory with geostatistics, we were able to link the clustering directly to the uncertainty of the measurements. The observation uncertainty was analytically propagated into dispersion functions and thus, uncertainty bounds for their distance turned into a valuable measure for clustering. Clusters are distinct because the soil moisture patterns are distinct with respect to the observation uncertainty.

The Python software SciKit-GStat was developed in the scope of this thesis. The Python package is highly flexible, generalized, well-documented, and well-tested. It has been used in a number of other research articles and has been used by engineering companies, which is evidence of the maturity to be used in practical applications. Especially interfaces to other software for geostatistical interpolation and simulation make SciKit-GStat an indispensable part of the geostatistical analysis environment in Python.

With SciKit-GStat available, further developments became possible. The SciKit-GStat uncertainty expansion is a well-written, graphical interface for SciKit-GStat and related tools. Beside the easier educational usage, a major extension to handle (observation) uncertainties in variograms have been presented. We embed the concept of replacing empirical variograms through uncertainty bounds. This way, the choice of an interpretive model and its parameters is subject to uncertainty itself, which needs to be considered. This links back to dispersion functions calculated for the soil moisture measurements in the Attert catchment, which were clustered with respect to the same source of uncertainty. While the respective chapter does not perform the step of choosing an interpretive model, SciKit-GStat Uncertainty now finally presents a framework in which such a choice can be made relying on a processing module that implements a suite of methods for the quantification of uncertainty associated with empirical variograms.

Finally, the flexibility of SciKit-GStat is embedded into the way the broader context of advancing reusable research tools. I presented a simple, additive specification

for research tools. Due to its additive character in combination with containerization, it is possible to implement the specification for a multitude of different existing research software, which is not standardized. This is not only convenient as it speeds up workflows and makes them replicable, but the concept also provides context to each research tool individually, as input and output data are clearly communicated and parameters are separated from data and clearly described.
This was exemplified by combining a broader application of dispersion functions to the same data used in the first place and finally combining the variogram functions with force-directed diagrams, which are not part of the geostatistical domain. I provided evidence that force-directed diagrams are a valuable tool to visualize and also analyze the covariance structure of spatial samples, which is captured by an empirical variogram.

As a personal statement, I want to highlight that the advances in building research software focused on reliable and easy-to-use research code, while the research itself guided the technical developments. In my opinion, this is the main benefit and purpose of good research software. The thesis illustrates how technical developments to produce more reliable research code, measured to the standards of software engineering, can be fed back, to fuel new research. As SciKit-GStat and SciKit-GStat Uncertainty was not developed as an end in themselves, it was possible for me to keep on building on the same basis. As findings from the first studies became more easily reproducible it was possible to rigorously weave the open ends discussed. Finally, an in-depth framework for analyzing and working with observation uncertainties in geostatistics is made available.

# Bibliography


Akaike, H (1973). **Information theory and an extension of the likelihood ratio principle**. In: *Proceedings of the Second International Symposium of Information Theory, Edited by BN Petrov and F. Csaki*. Vol. 257, 281 (see page 68).

Albertson, John D and Nicola Montaldo (2003). **Temporal dynamics of soil moisture variability: 1. Theoretical basis**. *Water Resources Research* 39:10 (see page 39).

Arthur, A. Mac and I. Robinson (2015). **A critique of field spectroscopy and the challenges and opportunities it presents for remote sensing for agriculture, ecosystems, and hydrology**. In: *Remote Sensing for Agriculture, Ecosystems, and Hydrology XVII*. Ed. by Christopher M. U. Neale and Antonino Maltese. Vol. 9637. International Society for Optics and Photonics. SPIE, 29–39. DOI: 10.1117/12.2201046. URL: https://doi.org/10.1117/12.2201046 (see page 54).

Atkinson, Peter M. and Nicholas J. Tate (2000). **Spatial Scale Problems and Geostatistical Solutions: A Review**. *Professional Geographer* 52:4, 607–623. ISSN: 14679272. DOI: 10.1111/0033-0124.00250 (see page 96).

Baker, Monya (2016). **1,500 scientists lift the lid on reproducibility**. *Nature* 533:7604 (see page 7).

Bannister, Michael J, David Eppstein, Michael T Goodrich, and Lowell Trott (2013). **Force-directed graph drawing using social gravity and scaling**. In: *Graph Drawing: 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers 20*. Springer, 414–425 (see pages 155, 162).

Bárdossy, András (2006). **Copula-based geostatistical models for groundwater quality parameters**. *Water Resour. Res* 42. DOI: 10.1029/2005WR004754 (see page 97).

Bárdossy, András and Zbigniew W. Kundzewicz (July 1990). **Geostatistical methods for detection of outliers in groundwater quality spatial fields**. *Journal of Hydrology* 115:1-4, 343–359. ISSN: 0022-1694. DOI: 10.1016/0022-1694(90)90213-H. URL: https://www.sciencedirect.com/science/article/pii/002216949090213H (see page 22).

Bárdossy, András and W. Lehmann (1998). **Spatial distribution of soil moisture in a small catchment. Part 1: Geostatistical analysis**. *Journal of Hydrology* 206:1-2, 1–15. ISSN: 00221694. DOI: 10.1016/S0022-1694(97)00152-2 (see pages 16, 99, 112).

Bárdossy, András and Jing Li (2008). **Geostatistical interpolation using copulas**. *Water Resources Research* 44:7. ISSN: 00431397. DOI: 10.1029/2007WR006115 (see page 97).

Barlow, Richard E., D.J. Bartholomew, J.M. Bremner, and H.D. Brunk (1972). **Statistical Inference Under Order Restrictions: Theory and Application of Isotonic Regression**. Tech. rep. MISSOURI UNIV COLUMBIA DEPT OF STATISTICS, 400 (see page 23).

Bayer, Michael (2012). "SQLAlchemy." In: *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. Ed. by Amy Brown and Greg Wilson. aosabook.org. URL: http://aosabook.org/en/sqlalchemy.html (see page 57).

Beven, K. J. (2000). **Uniqueness of place and process representations in hydrological modelling**. *Hydrology and Earth System Sciences* 4:2, 203–213. DOI: 10.5194/hess-4-203-2000. URL: https://hess.copernicus.org/articles/4/203/2000/ (see page 154).

Beven, Keith and Andrew Binley (1992). **The future of distributed models: Model calibration and uncertainty prediction**. *Hydrological Processes* 6:3, 279–298. DOI: https://doi.org/10.1002/hyp.3360060305. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hyp.3360060305. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/hyp.3360060305 (see page 81).

Beyer, Kevin, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft (1999). **When Is "Nearest Neighbor" Meaningful?** In: *International conference on database theory*. Springer, Berlin, Heidelberg, 217–235. DOI: 10.1007/3-540-49257-7_15. URL: http://link.springer.com/10.1007/3-540-49257-7%7B%5C_%7D15 (see page 41).

Bivand, Roger S, Edzer J Pebesma, Virgilio Gómez-Rubio, and Edzer Jan Pebesma (2008). **Applied spatial data analysis with R**. Vol. 747248717. Springer (see pages 101, 145, 146).

Blume, T., E. Zehe, and A. Bronstert (June 2009). **Use of soil moisture dynamics and patterns at different spatio-temporal scales for the investigation of subsurface flow processes**. *Hydrology and Earth System Sciences* 13:7, 1215–1233. ISSN: 1607-7938. DOI: 10.5194/hess-13-1215-2009 (see page 16).

Boisvert, J. B., J. G. Manchuk, and C. V. Deutsch (2009). **Kriging in the presence of locally varying anisotropy using non-euclidean distances**. *Mathematical Geosciences* 41:5, 585–601. ISSN: 18748961. DOI: 10.1007/s11004-009-9229-1 (see page 97).

Boisvert, Jeff B. and Clayton V. Deutsch (2011). **Programs for kriging and sequential Gaussian simulation with locally varying anisotropy using non-Euclidean distances**. *Computers and Geosciences* 37:4, 495–510. ISSN: 00983004. DOI: 10.1016/j.cageo.2010.03.021. URL: http://dx.doi.org/10.1016/j.cageo.2010.03.021 (see page 97).

Branch, Mary Ann, Thomas F Coleman, and Yuying Li (1999). **A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems**. *SIAM Journal on Scientific Computing* 21:1, 1–23 (see page 132).

Bras, Rafael L. (Aug. 2015). *Complexity and organization in hydrology: A personal view*. DOI: 10.1002/2015WR016958. URL: http://doi.wiley.com/10.1002/2015WR016958 (see page 17).

Brocca, L, F Melone, T Moramarco, and R Morbidelli (Jan. 2009). **Soil moisture temporal stability over experimental areas in Central Italy**. *Geoderma* 148:3-4, 364–374. ISSN: 00167061. DOI: 10.1016/j.geoderma.2008.11.004 (see page 16).

Brocca, L., R. Morbidelli, F. Melone, and T. Moramarco (Feb. 2007). **Soil moisture spatial variability in experimental areas of central Italy**. *Journal of Hydrology* 333:2-4, 356–373. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2006.09.004. URL: https://www.sciencedirect.com/science/article/pii/S0022169406004781 (see page 16).

Brocca, L., T. Tullo, F. Melone, T. Moramarco, and R. Morbidelli (Feb. 2012). **Catchment scale soil moisture spatial-temporal variability**. *Journal of Hydrology* 422-423, 63–75. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2011.12.039. URL: https://www.sciencedirect.com/science/article/pii/S0022169411009437 (see page 16).

Bronstert, Axel, Benjamin Creutzfeldt, Thomas Graeff, Irena Hajnsek, Maik Heistermann, Sibylle Itzerott, Thomas Jagdhuber, David Kneis, Erika Lück, Dominik Reusser, and Erwin Zehe (Feb. 2012). **Potentials and constraints of different types of soil moisture observations for flood simulations in headwater catchments**. *Natural Hazards* 60:3, 879–914. ISSN: 0921030X. DOI: 10.1007/s11069-011-9874-9. URL: http://link.springer.com/10.1007/s11069-011-9874-9 (see page 16).

Burgess, T. M. and R. Webster (1980). **Optimal interpolation and isarithmic mapping of soil properties. I. The semi-variogram and punctual kriging**. *Journal of Soil and Science* 31:2, 315–331, 7 figs, 1 table, 27 refs. ISSN: 00224588. DOI: 10.1111/j.1365-2389.1980.tb02084.x (see pages 17, 66, 99, 106, 112, 128).

Chakravarti, Nilotpal (1989). **Isotonic median regression: a linear programming approach**. *Mathematics of operations research* 14:2, 303–308 (see page 130).

Chapman, Lee, Cassandra Bell, and Simon Bell (2017). **Can the crowdsourcing data paradigm take atmospheric science to a new level? A case study of the urban heat island of London quantified using Netatmo weather stations**. *International Journal of Climatology* 37:9, 3597–3605. DOI: https://doi.org/10.1002/joc.4940. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/joc.4940. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/joc.4940 (see page 54).

Chen, Min, Alexey Voinov, Daniel P. Ames, Albert J. Kettner, Jonathan L. Goodall, Anthony J. Jakeman, Michael C. Barton, Quillon Harpham, Susan M. Cuddy, Cecelia DeLuca, Songshan Yue, Jin Wang, Fengyuan Zhang, Yongning Wen, and Guonian Lü (2020). **Position paper: Open web-distributed integrated geographic modelling and simulation to enable broader participation and applications**. *Earth-Science Reviews* 207, 103223. ISSN: 0012-8252. DOI: https://doi.org/10.1016/j.earscirev.2020.103223. URL: https://www.sciencedirect.com/science/article/pii/S0012825220302695 (see page 7).

Choi, Woojae and Ronald L Jacobs (2011). **Influences of formal learning, personal learning orientation, and supportive learning environment on informal learning**. *Human Resource Development Quarterly* 22:3, 239–257 (see page 39).

Choi, Young-Don, Jonathan L. Goodall, Jeffrey M. Sadler, Anthony M. Castronova, Andrew Bennett, Zhiyu Li, Bart Nijssen, Shaowen Wang, Martyn P. Clark, Daniel P. Ames, Jeffery S. Horsburgh, Hong Yi, Christina Bandaragoda, Martin Seul, Richard Hooper, and David G. Tarboton (2021). **Toward open and reproducible environmental modeling by integrating online data repositories, computational environments, and model Application Programming Interfaces**. *Environmental Modelling& Software* 135, 104888. ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2020.104888. URL: https://www.sciencedirect.com/science/article/pii/S1364815220309452 (see pages 7, 154).

Christakos, George (2000). **Modern spatiotem poral Geostatistics**. Ed. by Jo Anne DeGraffenreid. 1st. New York City, USA: Oxford University Press (OUP), 272. ISBN: 0195138953 (see page 97).

Clark, Martyn P, Andrew G Slater, David E Rupp, Ross A Woods, Jasper A Vrugt, Hoshin V Gupta, Thorsten Wagener, and Lauren E Hay (2008). **Framework for Understanding Structural Errors (FUSE): A modular framework to diagnose differences between hydrological models**. *Water Resources Research* 44:12 (see page 153).

Comaniciu, Dorin and Peter Meer (2002). **Mean shift: A robust approach toward feature space analysis**. *IEEE Transactions on pattern analysis and machine intelligence* 24:5, 603–619 (see page 23).

Consortium, Open Geospatial (2019). *OGC API - Features - Part 1: Core*. http://docs.opengeospatial.org/is/17-069r3/17-069r3.html. Accessed: 2023-03-30 (see page 156).

Cressie, Noel and Douglas M. Hawkins (1980). **Robust estimation of the variogram: I**. *Journal of the International Association for Mathematical Geology* 12:2, 115–125. ISSN: 00205958. DOI: 10.1007/BF01035243 (see pages 22, 63, 99, 105, 126, 163).

Curriero, Frank C (2005). **on the Use of Non-Euclidean Isotropy in Geostatistics on the Use of Non-Euclidean Isotropy in Geostatistics**. *Environmental Health*: December 2005 (see page 97).

Daly, Edorado and Amilcare Porporato (2005). **A review of soil moisture dynamics: from rainfall infiltration to ecosystem response**. *Environmental engineering science* 22:1, 9–24 (see page 16).

Damos, Petros (2016). **Using multivariate cross correlations, Granger causality and graphical models to quantify spatiotemporal synchronization and causality between pest populations**. *BMC ecology* 16, 1–17 (see page 155).

Danaci, Hasan Fehmi, Rengul Cetin-Atalay, and Volkan Atalay (2018). **EClerize: A customized force-directed graph drawing algorithm for biological graphs with EC attributes**. *Journal of Bioinformatics and Computational Biology* 16:04, 1850007 (see page 155).

De Cesare, L, D E Myers, and D Posa (2002). **FORTRAN programs for space-time modeling**. *Computers & Geosciences* 28:2, 205–212 (see pages 17, 97, 98, 118, 139).

Delbari, Masoomeh, Peyman Afrasiab, and Willibald Loiskandl (2009). **Using sequential Gaussian simulation to assess the field-scale spatial uncertainty of soil water content**. *Catena* 79:2, 163–169 (see page 51).

Dell'Oca, A., A. Guadagnini, and M. Riva (2020). **Interpretation of multi-scale permeability data through an information theory perspective**. *Hydrology and Earth System Sciences* 24:6, 3097–3109. DOI: 10.5194/hess-24-3097-2020. URL: https://hess.copernicus.org/articles/24/3097/2020/ (see page 59).

Der Kiureghian, Armen and Ove Ditlevsen (2009). **Aleatory or epistemic? Does it matter?** *Structural safety* 31:2, 105–112 (see page 53).

Deutsch, Clayton V and André G Journel (1998). **GSLib**. *Geostatistical software library and user's guide* 369 (see page 98).

Dimitrakopoulos, Roussos and Xiaochun Luo (1994). "Spatiotemporal modelling: covariances and ordinary kriging systems." In: *Geostatistics for the next century*. Springer, 88–93 (see page 139).

Doane, David P (1976). **Aesthetic frequency classifications**. *The American Statistician* 30:4, 181–183 (see page 123).

Dooge, James CI (1986). **Looking for hydrologic laws**. *Water Resources Research* 22:9S (see page 17).

Dowd, P. A. (1991). **A review of recent developments in geostatistics**. *Computers and Geosciences* 17:10, 1481–1500. ISSN: 00983004. DOI: 10.1016/0098-3004(91)90009-3 (see page 96).

– (1984). "The variogram and kriging: robust and resistant estimators." In: *Geostatistics for natural resources characterization*. Springer, 91–106 (see pages 63, 105, 126).

Emery, Xavier and María Peláez (2011). **Assessing the accuracy of sequential Gaussian simulation and cosimulation**. *Computational Geosciences* 15:4, 673–689 (see page 51).

Entekhabi, Dara, Ignacio Rodriguez-Iturbe, and Rafael L Bras (1992). **Variability in large-scale water balance with land surface-atmosphere interaction**. *Journal of Climate* 5:8, 798–813 (see page 36).

Essawy, Bakinam T., Jonathan L. Goodall, Daniel Voce, Mohamed M. Morsy, Jeffrey M. Sadler, Young Don Choi, David G. Tarboton, and Tanu Malik (2020). **A taxonomy for reproducible and replicable research in environmental modelling**. *Environmental Modelling& Software* 134, 104753. ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2020.104753. URL: https://www.sciencedirect.com/science/article/pii/S1364815219311612 (see pages 5, 7).

Euser, Tanja, HC Winsemius, Markus Hrachowitz, Fabrizio Fenicia, Stefan Uhlenbrook, and HHG Savenije (2013). **A framework to assess the realism of model structures using hydrological signatures**. *Hydrology and Earth System Sciences* 17:5, 1893–1912 (see page 153).

Fersch, B., T. Francke, M. Heistermann, M. Schrön, V. Döpper, J. Jakobi, G. Baroni, T. Blume, H. Bogena, C. Budach, T. Gränzig, M. Förster, A. Güntner, H.-J. Hendricks Franssen, M. Kasner, M. Köhli, B. Kleinschmit, H. Kunstmann, A. Patil, D. Rasche, L. Scheiffele, U. Schmidt, S. Szulc-Seyfried, J. Weimar, S. Zacharias, M. Zreda, B. Heber, R. Kiese, V. Mares, H. Mollenhauer, I. Völksch, and S. Oswald (2020). **A dense network of cosmic-ray neutron sensors for soil moisture observation in a highly instrumented pre-Alpine headwater catchment in Germany**. *Earth System Science Data* 12:3, 2289–2309. DOI: 10.5194/essd-12-2289-2020. URL: https://essd.copernicus.org/articles/12/2289/2020/ (see pages 102, 117).

Freedman, David and Persi Diaconis (1981). **On the histogram as a density estimator: L 2 theory**. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 57:4, 453–476 (see page 123).

Frey, Brendan J and Delbert Dueck (2007). **Clustering by passing messages between data points**. *science* 315:5814, 972–976 (see page 163).

Fruchterman, Thomas MJ and Edward M Reingold (1991). **Graph drawing by force-directed placement**. *Software: Practice and experience* 21:11, 1129–1164 (see pages 155, 162).

Fukunaga, Keinosuke and Larry Hostetler (1975). **The estimation of the gradient of a density function, with applications in pattern recognition**. *IEEE Transactions on information theory* 21:1, 32–40 (see page 23).

Gao, Fuchang and Lixing Han (2012). **Implementing the Nelder-Mead simplex algorithm with adaptive parameters**. *Computational Optimization and Applications* 51:1, 259–277 (see pages 125, 133).

Gelman, Andrew, John B Carlin, Hal S Stern, David B Dunson, and Donald B Vehtari Aki Rubin (2014). **Bayesian data analysis - Third edition**. Chapman and Hall/CRC (see page 69).

Genc, Burkay and Ugur Dogrusoz (2003). **A constrained, force-directed layout algorithm for biological pathways**. In: *Graph Drawing*. Vol. 2003. Springer, 314–319 (see page 155).

Genton, Marc G (1998). **Highly robust variogram estimation**. *Mathematical Geology* 30:2, 213–221 (see pages 63, 105, 126).

Gómez-Plaza, A, M Martınez-Mena, J Albaladejo, and VM Castillo (2001). **Factors regulating spatial distribution of soil water content in small semiarid catchments**. *Journal of hydrology* 253:1-4, 211–226 (see page 39).

Goovaerts, P. (Feb. 2000). **Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall**. *Journal of Hydrology* 228:1-2, 113–129. ISSN: 00221694. DOI: 10.1016/S0022-1694(00)00144-X (see page 96).

Gräler, Benedikt, Edzer Pebesma, and Gerard Heuvelink (2016). **Spatio-Temporal Interpolation using gstat**. *The R Journal* 8 (1), 204–218. URL: https://journal.r-project.org/archive/2016/RJ-2016-014/index.html (see page 98).

Grayson, Rodger B., Andrew W. Western, Francis H. S. Chiew, and Günter Blöschl (1997). **Preferred states in spatial soil moisture patterns: Local and nonlocal controls**. *Water Resources Research* 33:12, 2897–2908. ISSN: 0043-1397. DOI: 10.1029/97WR02174. URL: http://onlinelibrary.wiley.com/doi/10.1029/97WR02174/full (see pages 16, 38, 40).

Guadagnini, Alberto, Monica Riva, and Shlomo P. Neuman (July 2018). *Recent advances in scalable non-Gaussian geostatistics: The generalized sub-Gaussian model*. DOI: 10.1016/j.jhydrol.2018.05.001. URL: https://www.sciencedirect.com/science/article/pii/S0022169418303287 (see page 97).

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart (2008). **Exploring Network Structure, Dynamics, and Function using NetworkX**. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 11–15 (see pages 164, 171).

Handcock, Mark S and Michael L Stein (1993). **A Bayesian analysis of kriging**. *Technometrics* 35:4, 403–410 (see page 51).

Heathman, Gary C, Myriam Larose, Michael H Cosh, and Rajat Bindlish (2009). **Surface and profile soil moisture spatio-temporal analysis during an excessive rainfall period in the Southern Great Plains, USA**. *Catena* 78:2, 159–169 (see page 16).

Hinterding, Angela (2003). **Entwicklung hybrider Interpolationsverfahren für den automatisierten Betrieb am Beispiel meteorologischer Größen**. IfGi (see pages 23, 130).

Höge, Marvin, Thomas Wöhling, and Wolfgang Nowak (2018). **A primer for model selection: The decisive role of model complexity**. *Water Resources Research* 54:3, 1688–1715 (see page 68).

Hora, Stephen C (1996). **Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management**. *Reliability Engineering & System Safety* 54:2-3, 217–223 (see page 53).

Hu, L. Y. and T. Chugunova (Nov. 2008). **Multiple-point geostatistics for modeling subsurface heterogeneity: A comprehensive review**. *Water Resources Research* 44:11. ISSN: 00431397. DOI: 10.1029/2008WR006993. URL: http://doi.wiley.com/10.1029/2008WR006993 (see page 97).

Hüllermeier, Eyke and Willem Waegeman (2021). **Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods**. *Machine Learning* 110:3, 457–506 (see page 53).

Hunter, J. D. (May 2007a). **Matplotlib: A 2D Graphics Environment**. *Computing in Science Engineering* 9:3, 90–95. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.55 (see page 104).

– (2007b). **Matplotlib: A 2D graphics environment**. *Computing in Science & Engineering* 9:3, 90–95. DOI: 10.1109/MCSE.2007.55 (see page 14).

Hurvich, Clifford M and Chih-Ling Tsai (1989). **Regression and time series model selection in small samples**. *Biometrika* 76:2, 297–307 (see page 68).

Inc., Plotly Technologies (2015). *Collaborative data science*. URL: https://plot.ly (see page 104).

Itoh, Takayuki, Chris Muelder, Kwan-Liu Ma, and Jun Sese (2009). **A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs**. In: *2009 IEEE Pacific Visualization Symposium*. IEEE, 121–128 (see pages 155, 162).

IUSS Working Group (2006). **WRB, 2006**. *World reference base for soil resources* (see page 20).

Jackisch, C., K. Germer, T. Graeff, I. Andrä, K. Schulz, M. Schiedung, J. Haller-Jans, J. Schneider, J. Jaquemotte, P. Helmer, L. Lotz, A. Bauer, I. Hahn, M. Šanda, M. Kumpan, J. Dorner, G. de Rooij, S. Wessel-Bothe, L. Kottmann, S. Schittenhelm, and W. Durner (2020). **Soil moisture and matric potential – an open field comparison of sensor systems**. *Earth System Science Data* 12:1, 683–697. DOI: 10.5194/essd-12-683-2020. URL: https://essd.copernicus.org/articles/12/683/2020/ (see page 54).

Jewell, Sharon A and Nicolas Gaussiat (2015). **An assessment of kriging-based rain-gauge–radar merging techniques**. *Quarterly Journal of the Royal Meteorological Society* 141:691, 2300–2313 (see page 96).

Johnson, Stephen C (1967). **Hierarchical clustering schemes**. *Psychometrika* 32:3, 241–254 (see page 124).

Jost, G, GBM Heuvelink, and A Papritz (2005). **Analysing the space–time distribution of soil water storage of a forest ecosystem using spatio-temporal kriging**. *Geoderma* 128:3-4, 258–273 (see page 17).
Journel, A G and C J Huijbregts (Jan. 1976). *Mining geostatistics* (see pages 66, 106, 129).
Journel, André G (1994). "Modeling uncertainty: some conceptual thoughts." In: *Geostatistics for the next century*. Springer, 30–43 (see page 51).
Kiese, R, B Fersch, C Baessler, C Brosy, K Butterbach-Bahl, Christian Chwala, M Dannenmann, J Fu, R Gasche, R Grote, et al. (2018). **The TERENO Pre-Alpine Observatory: Integrating meteorological, hydrological, and biogeochemical measurements and modeling**. *Vadose Zone Journal* 17:1, 1–17 (see page 102).
Kitanidis, Peter K. and Efstratios G. Vomvoris (June 1983). **A geostatistical approach to the inverse problem in groundwater modeling (steady state) and one-dimensional simulations**. *Water Resources Research* 19:3, 677–690. ISSN: 19447973. DOI: 10.1029/WR019i003p00677 (see page 17).
Kleidon, Axel (2012). **How does the Earth system generate and maintain thermodynamic disequilibrium and what does it imply for the future of the planet?** *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 370:1962, 1012–1040 (see page 26).
Kondepudi, Dilip and Ilya Prigogine (1998). "From heat engines to dissipative structures." In: *Modern Thermodynamics*. John Wiley & Sons Chichester (see page 26).
Kraft, Dieter et al. (1988). **A software package for sequential quadratic programming** (see page 133).
Krige, D. G. (1951). **Journal of the Chemical Metallu , Rgical & Mining Society of South Africa**. *Journal of the Chemical Metallurgical & Society of South Mining Africa* 52:6, 119–139. URL: http://journals.co.za/content/saimm/52/6/AJA0038223X%7B%5C_%7D4792 (see pages 4, 96).
Kullback, Solomon and Richard A Leibler (1951). **On information and sufficiency**. *The annals of mathematical statistics* 22:1, 79–86 (see page 26).
Lark, R. M. (2000). **Estimating variograms of soil properties by the method-of-moments and maximum likelihood**. *European Journal of Soil Science* 51:4, 717–728. DOI: https://doi.org/10.1046/j.1365-2389.2000.00345.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1365-2389.2000.00345.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-2389.2000.00345.x (see pages 68, 98, 133, 148, 149).
– (2012). **Towards soil geostatistics**. *Spatial Statistics* 1, 92–99. ISSN: 22116753. DOI: 10.1016/j.spasta.2012.02.001 (see page 17).
Leavesley, GH, LG Stannard, et al. (1995). **The precipitation-runoff modeling system-PRMS.** *Computer models of watershed hydrology.*, 281–310 (see page 153).
Lloyd, CD and Peter M Atkinson (2001). **Assessing uncertainty in estimates with ordinary and indicator kriging**. *Computers & Geosciences* 27:8, 929–937 (see page 51).
Loritz, R., H. Gupta, C. Jackisch, M. Westhoff, A. Kleidon, U. Ehret, and E. Zehe (2018). **On the dynamic nature of hydrological similarity**. *Hydrology and Earth System Sciences*

22:7, 3663–3684. DOI: 10.5194/hess-22-3663-2018. URL: https://hess.copernicus.org/articles/22/3663/2018/ (see pages 18, 23, 24, 71).

Loritz, Ralf, Sibylle K. Hassler, Conrad Jackisch, Niklas Allroggen, Loes van Schaik, Jan Wienhöfer, and Erwin Zehe (Mar. 2017). **Picturing and modeling catchments by representative hillslopes**. *Hydrology and Earth System Sciences* 21:2, 1225–1249. ISSN: 1607-7938. DOI: 10.5194/hess-21-1225-2017. URL: https://www.hydrol-earth-syst-sci.net/21/1225/2017/ (see pages 19, 20).

Ly, S., C. Charles, and A. Degré (June 2011). **Geostatistical interpolation of daily rainfall at catchment scale: The use of several variogram models in the Ourthe and Ambleve catchments, Belgium**. *Hydrology and Earth System Sciences* 15:7, 2259–2274. ISSN: 10275606. DOI: 10.5194/hess-15-2259-2011 (see page 17).

Ly, Sarann, Catherine Charles, and Aurore Degré (2015). **Different methods for spatial interpolation of rainfall data for operational hydrology and hydrological modeling at watershed scale: a review**. *Biotechnologie, Agronomie, Société et Environnement* 17:2013, 1–10 (see page 97).

Ma, Chunsheng (2002). **Spatio-temporal covariance functions generated by mixtures**. *Mathematical geology* 34:8, 965–975 (see pages 17, 97).

– (2003). **Spatio-temporal stationary covariance models**. *Journal of Multivariate Analysis* 86:1, 97–107 (see page 17).

– (2005). **SPATIO-TEMPORAL VARIOGRAMS AND COVARIANCE MODELS**. *Advances in Applied Probability* 725:April, 706–725 (see page 97).

MacQueen, James et al. (1967). **Some methods for classification and analysis of multivariate observations**. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, 281–297 (see page 124).

Mälicke, M., S. K. Hassler, T. Blume, M. Weiler, and E. Zehe (2020). **Soil moisture: variable in space but redundant in time**. *Hydrology and Earth System Sciences* 24:5, 2633–2653. DOI: 10.5194/hess-24-2633-2020. URL: https://hess.copernicus.org/articles/24/2633/2020/ (see pages 71, 97, 167, 168).

Mälicke, Mirko (Nov. 2019). *Companion Code for: Soil moisture: variable in space but redundant in time. (10.5194/hess-2019-574)*. Version discussion. DOI: 10.5281/zenodo.3773110 (see page 14).

– (May 2021). *KIT-HYD/scikit_gstat_companion_code: Version 0.1.0*. Version 0.1.0. DOI: 10.5281/zenodo.4817675. URL: https://doi.org/10.5281/zenodo.4817675 (see page 94).

– (May 2022a). *hydrocode-de/skgstat_uncertainty: Version 1.3*. Version v1.3.0. DOI: 10.5281/zenodo.6545079. URL: https://doi.org/10.5281/zenodo.6545079 (see page 56).

– (Nov. 2022b). *VForWaTer/tool_moving_dispersion: Version 0.2*. Version v0.2.0. DOI: 10.5281/zenodo.7361231. URL: https://doi.org/10.5281/zenodo.7361231 (see page 163).

– (Mar. 2023a). *hydrocode-de/tool-runner-js: Version 0.5*. Version v0.5.0. DOI: 10.5281/zenodo.7784089. URL: https://doi.org/10.5281/zenodo.7784089 (see page 160).

– (Mar. 2023b). *VForWaTer/tool_clustering: Preview Release*. Version v0.2.0. DOI: 10.5281/zenodo.7784770. URL: https://doi.org/10.5281/zenodo.7784770 (see page 163).

Mälicke, Mirko (Mar. 2023c). *VForWaTer/tool_data_studio: Preview Release*. Version v0.1.0. DOI: 10.5281/zenodo.7788150. URL: https://doi.org/10.5281/zenodo.7788150 (see page 168).
Mälicke, Mirko and Alexander Dolich (Mar. 2023a). *hydrocode-de/tool-runner: Version 0.9.2*. Version v0.9.2. DOI: 10.5281/zenodo.7701279. URL: https://doi.org/10.5281/zenodo.7701279 (see page 160).
– (Mar. 2023b). *VForWaTer/tool-specs: Preview Release*. Version v0.1.0. DOI: 10.5281/zenodo.7784026. URL: https://doi.org/10.5281/zenodo.7784026 (see page 159).
Mälicke, Mirko, Egil Möller, Helge David Schneider, and Sebastian Müller (May 2021a). *mmaelicke/scikit-gstat: A scipy flavoured geostatistical variogram analysis toolbox*. Version v0.6.0. DOI: 10.5281/zenodo.4835779. URL: https://doi.org/10.5281/zenodo.4835779 (see pages 14, 50, 53, 54, 56, 59, 61–63, 66, 80, 94, 102, 107, 146).
– (May 2021b). *mmaelicke/scikit-gstat: A scipy flavoured geostatistical variogram analysis toolbox*. Version v0.6.0. DOI: 10.5281/zenodo.4835779. URL: https://doi.org/10.5281/zenodo.4835779 (see page 62).
Marshall, JS, RC Langille, and W Mc K Palmer (1947). **Measurement of rainfall by radar**. *Journal of Meteorology* 4:6, 186–192 (see page 96).
Martinez-Carreras, Núria, Andreas Krein, Francesc Gallart, Jean-François Iffly, Christophe Hissler, Laurent Pfister, Lucien Hoffmann, and Philip N Owens (2012). **The influence of sediment sources and hydrologic events on the nutrient and metal content of fine-grained sediments (attert river basin, Luxembourg)**. *Water, Air, & Soil Pollution* 223:9, 5685–5705 (see page 20).
Martínez-Fernández, José and Antonio Ceballos (2003). **Temporal stability of soil moisture in a large-field experiment in Spain**. *Soil Science Society of America Journal* 67:6, 1647–1656 (see page 16).
Matheron, Georges (1963). **Principles of geostatistics**. *Economic geology* 58:8, 1246–1266 (see pages 61, 63, 96, 105, 110, 126).
McDonnell, J. J., M. L. Roderick, J. Selker, K. Vaché, C. Hinz, R. Hooper, G. Grant, M. Sivapalan, J. Kirchner, M. Weiler, S. Dunn, and R. Haggerty (June 2007). **Moving beyond heterogeneity and process complexity: A new vision for watershed hydrology**. *Water Resources Research* 43:7. ISSN: 00431397. DOI: 10.1029/2006wr005467. URL: http://doi.wiley.com/10.1029/2006WR005467 (see page 17).
McNamara, James P, David Chandler, Mark Seyfried, and Shiva Achet (2005). **Soil moisture states, lateral flow, and streamflow generation in a semi-arid, snowmelt-driven catchment**. *Hydrological Processes: An International Journal* 19:20, 4023–4038 (see page 38).
Meyles, Erik, Andrew Williams, Les Ternan, and John Dowd (Feb. 2003). **Runoff generation in relation to soil moisture patterns in a small Dartmoor catchment, Southwest England**. *Hydrological Processes* 17:2, 251–264. ISSN: 08856087. DOI: 10.1002/hyp.1122. URL: http://doi.wiley.com/10.1002/hyp.1122 (see page 38).

Michaelis, Christopher D. and Daniel P. Ames (2008). "Web Feature Service (WFS) and Web Map Service (WMS)." In: *Encyclopedia of GIS*. Ed. by Shashi Shekhar and Hui Xiong. Boston, MA: Springer, 1259–1261. DOI: 10.1007/978-0-387-35973-1_1480 (see page 156).

Mittelbach, Heidi and Sonia I Seneviratne (2012). **A new perspective on the spatio-temporal variability of soil moisture: Temporal dynamics versus time invariant contributions**. *Hydrology and Earth System Sciences* 16:7, 2169–2179 (see pages 35–37).

Montero, José-María, Gema Fernández-Avilés, and Jorge Mateu (2015). **Spatial and spatio-temporal geostatistical modeling and kriging**. John Wiley & Sons (see pages 66, 99, 106, 110, 112, 114, 129, 130, 136, 137, 140).

Moré, Jorge J (1978). "The Levenberg-Marquardt algorithm: implementation and theory." In: *Numerical analysis*. Springer, 105–116 (see page 132).

Mowrer, H Todd (1997). **Propagating uncertainty through spatial estimation processes for old-growth subalpine forests using sequential Gaussian simulation in GIS**. *Ecological Modelling* 98:1, 73–86 (see page 51).

Müller, S., L. Schüler, A. Zech, and F. Heße (2021a). **GSTools v1.3: A toolbox for geostatistical modelling in Python**. *Geoscientific Model Development Discussions* 2021, 1–33. DOI: 10.5194/gmd-2021-301. URL: https://gmd.copernicus.org/preprints/gmd-2021-301/ (see pages 54, 56, 70).

– (2021b). **GSTools v1.3: A toolbox for geostatistical modelling in Python**. *Geoscientific Model Development Discussions* 2021, 1–33. DOI: 10.5194/gmd-2021-301. URL: https://gmd.copernicus.org/preprints/gmd-2021-301/ (see pages 99, 141).

Murphy, Benjamin, Roman Yurchak, and Sebastian Müller (Apr. 2021). *GeoStat-Framework/PyKrige: v1.6.0*. Version v1.6.0. DOI: 10.5281/zenodo.4661732. URL: https://doi.org/10.5281/zenodo.4661732 (see page 98).

Myers, Donald E and Andre Journel (1990). **Variograms with zonal anisotropies and noninvertible kriging systems**. *Mathematical Geology* 22:7, 779–785 (see page 139).

Nearing, Grey S, Frederik Kratzert, Alden Keefe Sampson, Craig S Pelissier, Daniel Klotz, Jonathan M Frame, Cristina Prieto, and Hoshin V Gupta (2021). **What role does hydrological science play in the age of machine learning?** *Water Resources Research* 57:3, e2020WR028091 (see page 154).

Neuper, M. and U. Ehret (2019). **Quantitative precipitation estimation with weather radar using a data- and information-based approach**. *Hydrology and Earth System Sciences* 23:9, 3711–3733. DOI: 10.5194/hess-23-3711-2019. URL: https://hess.copernicus.org/articles/23/3711/2019/ (see page 53).

Nowak, Marek and Georges Verly (2005). "The practice of sequential Gaussian simulation." In: *Geostatistics Banff 2004*. Springer, 387–398 (see page 51).

Oliphant, Travis E (2006). **A guide to NumPy**. Vol. 1. Trelgol Publishing USA (see page 104).

Pachepsky, Ya A, AK Guber, and Diederik Jacques (2005). **Temporal persistence in vertical distributions of soil moisture contents**. *Soil Science Society of America Journal* 69:2, 347–352 (see page 39).

Pardo-Igúzquiza, Eulogio and Peter Dowd (2001). **Variance–covariance matrix of the experimental variogram: assessing variogram uncertainty**. *Mathematical Geology* 33:4, 397–419 (see page 53).

Patil, S and M Stieglitz (2012). **Controls on hydrologic similarity: Role of nearby gauged catchments for prediction at an ungauged catchment**. *Hydrology and Earth System Sciences* 16:2, 551–562. ISSN: 10275606. DOI: 10.5194/hess-16-551-2012. URL: www.hydrol-earth-syst-sci.net/16/551/2012/ (see page 18).

Pebesma, Edzer and Roger S Bivand (2005). **S classes and methods for spatial data: the sp package**. *R news* 5:2, 9–13 (see pages 101, 145, 146).

Pebesma, Edzer J. (2004). **Multivariable geostatistics in S: the gstat package**. *Computers & Geosciences* 30, 683–691 (see page 98).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011a). **Scikit-learn: Machine Learning in Python**. *Journal of Machine Learning Research* 12, 2825–2830 (see pages 14, 23, 62, 106–108, 124, 125, 130).

– (2011b). **Scikit-learn: Machine Learning in Python**. *Journal of Machine Learning Research* 12, 2825–2830 (see page 163).

Pfister, Laurent, Joel Humbert, and Lucien Hoffmann (2000). **Recent trends in rainfall-runoff characteristics in the Alzette river basin, Luxembourg**. *Climatic Change* 45:2, 323–337 (see page 20).

Pool, M., J. Carrera, A. Alcolea, and E. M. Bocanegra (Dec. 2015). **A comparison of deterministic and stochastic approaches for regional scale inverse modeling on the Mar del Plata aquifer**. *Journal of Hydrology* 531, 214–229. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2015.09.064 (see page 17).

Rahman, Marufur and Rezaul Karim (2016). **Comparative study of different methods of social network analysis and visualization**. In: *2016 International Conference on Networking Systems and Security (NSysS)*. IEEE, 1–7 (see pages 155, 162).

Reichstein, Markus, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, et al. (2019). **Deep learning and process understanding for data-driven Earth system science**. *Nature* 566:7743, 195–204 (see page 154).

Riva, Monica, Shlomo P. Neuman, Alberto Guadagnini, and Martina Siena (2013). **Anisotropic Scaling of Berea Sandstone Log Air Permeability Statistics**. *Vadose Zone Journal* 12:3, vzj2012.0153. DOI: https://doi.org/10.2136/vzj2012.0153. eprint: https://acsess.onlinelibrary.wiley.com/doi/pdf/10.2136/vzj2012.0153. URL: https://acsess.onlinelibrary.wiley.com/doi/abs/10.2136/vzj2012.0153 (see page 59).

Riva, Monica, Marco Panzeri, Alberto Guadagnini, and Shlomo P Neuman (2011). **Role of model selection criteria in geostatistical inverse estimation of statistical data- and model-parameters**. *Water Resources Research* 47:7 (see page 68).

Rodríguez-Iturbe, Ignacio, Gustavo Devoto, and Juan B. Valdés (Dec. 1979). **Discharge response analysis and hydrologic similarity: The interrelation between the ge-**

**omorphologic IUH and the storm characteristics**. *Water Resources Research* 15:6, 1435–1444. ISSN: 19447973. DOI: 10.1029/WR015i006p01435. URL: http://doi.wiley.com/10.1029/WR015i006p01435 (see page 18).

Rolston, D. E., J. W. Biggar, and H. I. Nightingale (Dec. 1991). **Temporal persistence of spatial soil-water patterns under trickle irrigation**. *Irrigation Science* 12:4, 181–186. ISSN: 1432-1319. DOI: 10.1007/BF00190521. URL: https://doi.org/10.1007/BF00190521 (see page 16).

Rosenbaum, U., H. R. Bogena, M. Herbst, J. A. Huisman, T. J. Peterson, A. Weuthen, A. W. Western, and H. Vereecken (Oct. 2012). **Seasonal and event dynamics of spatial soil moisture patterns at the small catchment scale**. *Water Resources Research* 48:10. ISSN: 00431397. DOI: 10.1029/2011WR011518. URL: http://doi.wiley.com/10.1029/2011WR011518 (see page 36).

Russian, Anna, Marco Dentz, and Philippe Gouze (Aug. 2017). **Self-averaging and weak ergodicity breaking of diffusion in heterogeneous media**. *Phys. Rev. E* 96 (2), 022156. DOI: 10.1103/PhysRevE.96.022156. URL: https://link.aps.org/doi/10.1103/PhysRevE.96.022156 (see page 70).

Sampson, Paul D. and Peter Guttorp (1992). **Nonparametric Estimation of Nonstationary Spatial Covariance Structure**. *Journal of the American Statistical Association* 87:417, 108–119. DOI: 10.1080/01621459.1992.10475181. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1992.10475181. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1992.10475181 (see page 17).

Schiavo, Massimiliano, Monica Riva, Laura Guadagnini, Erwin Zehe, and Alberto Guadagnini (2022). **Probabilistic identification of Preferential Groundwater Networks**. *Journal of Hydrology* 610, 127906. ISSN: 0022-1694. DOI: https://doi.org/10.1016/j.jhydrol.2022.127906. URL: https://www.sciencedirect.com/science/article/pii/S0022169422004814 (see page 51).

Schume, Helmut, Georg Jost, and Klaus Katzensteiner (2003). **Spatio-temporal analysis of the soil water content in a mixed Norway spruce (Picea abies (L.) Karst.)-European beech (Fagus sylvatica L.) stand**. *Geoderma* 112:3-4, 273–287. ISSN: 00167061. DOI: 10.1016/S0016-7061(02)00311-7 (see page 17).

Scott, David W. (2009). **Sturges' rule**. *WIREs Computational Statistics* 1:3, 303–306. DOI: https://doi.org/10.1002/wics.35. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.35. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.35 (see page 122).

– (2010). **Scott's rule**. *WIREs Computational Statistics* 2:4, 497–502. DOI: https://doi.org/10.1002/wics.103. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.103. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.103 (see page 123).

Shannon, C. E. (1948). **A Mathematical Theory of Communication**. *Bell System Technical Journal* 27:3, 379–423. DOI: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1948.tb01338.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x (see pages 18, 25, 63, 71, 105, 121, 127, 136).

Simm, W. A., F. Samreen, R. Bassett, M. A. Ferrario, G. Blair, J. Whittle, and P. J. Young (2018). **SE in ES: Opportunities for Software Engineering and Cloud Computing in Environmental Science**. In: *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society*. ICSE-SEIS '18. Gothenburg, Sweden: Association for Computing Machinery, 61–70. ISBN: 9781450356619. DOI: 10.1145/3183428.3183430. URL: https://doi.org/10.1145/3183428.3183430 (see pages 6, 7).

Sivapalan, Murugesu (Apr. 2003). **Process complexity at hillslope scale, process simplicity at the watershed scale: is there a connection?** *Hydrological Processes* 17:5, 1037–1041. ISSN: 0885-6087. DOI: 10.1002/hyp.5109. URL: http://doi.wiley.com/10.1002/hyp.5109 (see page 17).

Sivapalan, Murugesu, Mary A Yaeger, Ciaran J Harman, Xiangyu Xu, and Peter A Troch (2011). **Functional model of water balance variability at the catchment scale: 1. Evidence of hydrologic similarity and space-time symmetry**. *Water Resources Research* 47:2. ISSN: 00431397. DOI: 10.1029/2010WR009568. URL: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2010WR009568 (see page 18).

Skamarock, William C and Joseph B Klemp (2008). **A time-split nonhydrostatic atmospheric model for weather research and forecasting applications**. *Journal of computational physics* 227:7, 3465–3485 (see page 153).

Snepvangers, JJJC, GBM Heuvelink, and JA Huisman (2003). **Soil water content interpolation using spatio-temporal kriging with external drift**. *Geoderma* 112:3-4, 253–271 (see page 17).

Spiegelhalter, David J, Nicola G Best, Bradley P Carlin, and Angelika Van Der Linde (2002). **Bayesian measures of model complexity and fit**. *Journal of the royal statistical society: Series b (statistical methodology)* 64:4, 583–639 (see pages 68, 69).

Spiegelhalter, David J, Nicola G Best, Bradley P Carlin, and Angelika Van der Linde (2014). **The deviance information criterion: 12 years on**. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76:3, 485–493 (see page 68).

Sprenger, Matthias, Stefan Seeger, Theresa Blume, and Markus Weiler (2016). **Travel times in the vadose zone: Variability in space and time**. *Water Resources Research* 52:8, 5727–5754 (see page 20).

Starr, JL, DJ Timlin, RE Cady, and TJ Nicholson (2002). **Evaluating Uncertainties in Ground-Water Recharge Estimates Through Advanced Monitoring**. In: *AGU Spring Meeting Abstracts* (see page 16).

Takagi, K. and H. S. Lin (Mar. 2012). **Changing controls of soil moisture spatial organization in the Shale Hills Catchment**. *Geoderma* 173-174, 289–302. ISSN: 00167061. DOI: 10.1016/j.geoderma.2011.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0016706111003132%7B%5C#%7Df0015 (see page 36).

Teuling, A. J., R. Uijlenhoet, F. Hupet, E. E. van Loon, and P. A. Troch (2006). **Estimating spatial mean root-zone soil moisture from point-scale observations**. *Hydrology and Earth System Sciences* 10:5, 755–767. DOI: 10.5194/hess-10-755-2006. URL: https://www.hydrol-earth-syst-sci.net/10/755/2006/ (see pages 16, 35, 37).

Teuling, Adriaan J and Peter A Troch (2005). **Improved understanding of soil moisture variability dynamics**. *Geophysical Research Letters* 32:5 (see page 39).

Thiesen, Stephanie, Diego M Vieira, Mirko Mälicke, Ralf Loritz, J Florian Wellmann, and Uwe Ehret (2020). **Histogram via entropy reduction (HER): an information-theoretic alternative for geostatistics**. *Hydrology and Earth System Sciences* 24:9, 4523–4540 (see page 127).

Tidwell, Vincent C and John L Wilson (1999). **Permeability upscaling measured on a block of Berea Sandstone: Results and interpretation**. *Mathematical Geology* 31:7, 749–769 (see pages 55, 59, 60).

– (1997). **Laboratory method for investigating permeability upscaling**. *Water Resources Research* 33:7, 1607–1616. DOI: https://doi.org/10.1029/97WR00804. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/97WR00804. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/97WR00804 (see pages 50, 55, 58–60).

– (2002). **Visual attributes of a rock and their relationship to permeability: A comparison of digital image and minipermeameter data**. *Water Resources Research* 38:11, 43-1-43–13. DOI: https://doi.org/10.1029/2001WR000932. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2001WR000932. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2001WR000932 (see pages 55, 59).

Todini, Ezio (2001). **Influence of parameter estimation uncertainty in Kriging: Part 1-Theoretical Development**. *Hydrology and Earth System Sciences* 5:2, 215–223 (see page 51).

Topp, GC, JL Davis, and AP Annan (1982). **Electromagnetic Determination of Soil Water Content Using TDR: I. Applications to Wetting Fronts and Steep Gradients 1**. *Soil Science Society of America Journal* 46:4, 672–678 (see page 16).

Topp, GC, WD Zebchuk, JL Davis, and WG Bailey (1984). **The measurement of soil water content using a portable TDR hand probe**. *Canadian Journal of Soil Science* 64:3, 313–321 (see page 16).

van der Walt, S., S. C. Colbert, and G. Varoquaux (Mar. 2011). **The NumPy Array: A Structure for Efficient Numerical Computation**. *Computing in Science Engineering* 13:2, 22–30. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37 (see pages 99, 104, 123, 163).

Vanderlinden, Karl, Harry Vereecken, Horst Hardelauf, Michael Herbst, Gonzalo Martínez, Michael H Cosh, and Yakov A Pachepsky (2012). **Temporal stability of soil water contents: A review of data and analyses**. *Vadose Zone Journal* 11:4, vzj2011–0178 (see pages 16, 39, 97).

Vapnik, VN and A Ya Chervonenkis (1974). **The method of ordered risk minimization, I**. *Avtomatika i Telemekhanika* 8, 21–30 (see pages 69, 88).

Vereecken, H, JA Huisman, H Bogena, Jan Vanderborght, JA Vrugt, and JW Hopmans (2008). **On the value of soil moisture measurements in vadose zone hydrology: A review**. *Water resources research* 44:4 (see pages 16, 97).

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Sté-

fan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors (2020). **SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python**. *Nature Methods* 17, 261–272. DOI: 10.1038/s41592-019-0686-2 (see pages 14, 62, 99, 104, 107, 125, 132, 133, 149).

Wackernagel, Hans (1998), 60–63. In: *Multivariate Geostatistics: An Introduction with Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-662-03550-4. DOI: 10.1007/978-3-662-03550-4_9. URL: https://doi.org/10.1007/978-3-662-03550-4_9 (see page 114).

Wagener, Thorsten, Murugesu Sivapalan, Peter Troch, and Ross Woods (July 2007). **Catchment Classification and Hydrologic Similarity**. *Geography Compass* 1:4, 901–931. ISSN: 1749-8198. DOI: 10.1111/j.1749-8198.2007.00039.x. URL: http://doi.wiley.com/10.1111/j.1749-8198.2007.00039.x (see page 18).

Ward Jr, Joe H and Marion E Hook (1963). **Application of an hierarchical grouping procedure to a problem of grouping profiles**. *Educational and Psychological Measurement* 23:1, 69–81 (see page 124).

Webster, R and MA Oliver (1993). "How large a sample is needed to estimate the regional variogram adequately?" In: *Geostatistics Tróia'92*. Springer, 155–166 (see page 53).

Weijs, S V, N Van De Giesen, and M B Parlange (2013). **Data compression to define information content of hydrological time series**. *Hydrology and Earth System Sciences* 17:8, 3171–3187. ISSN: 16077938. DOI: 10.5194/hess-17-3171-2013. URL: www.hydrol-earth-syst-sci.net/17/3171/2013/ (see page 18).

Weiler and Beven (2015). **Do we need a community hydrological model?** *Water Resources Research* 51:9, 7777–7784 (see pages 153, 154).

Wendi, Dadiyorto and Norbert Marwan (Aug. 2018). **Extended recurrence plot and quantification for noisy continuous dynamical systems**. *Chaos* 28:8, 085722. ISSN: 10541500. DOI: 10.1063/1.5025485. URL: http://aip.scitation.org/doi/10.1063/1.5025485 (see page 18).

Wendi, Dadiyorto, Norbert Marwan, and Bruno Merz (Jan. 2018). **In Search of Determinism-Sensitive Region to Avoid Artefacts in Recurrence Plots**. *International Journal of Bifurcation and Chaos* 28:01, 1850007. ISSN: 0218-1274. DOI: 10.1142/s0218127418500074. URL: https://www.worldscientific.com/doi/abs/10.1142/S0218127418500074 (see page 18).

Western, Andrew W and Rodger B Grayson (1998). **The Tarrawarra data set: Soil moisture patterns, soil characteristics, and hydrological flux measurements**. *Water Resources Research* 34:10, 2765–2768 (see pages 38, 40).

Western, Andrew W., Rodger B. Grayson, Günter Blöschl, Garry R. Willgoose, and Thomas A. McMahon (Mar. 1999). **Observed spatial organization of soil moisture and its relation to terrain indices**. *Water Resources Research* 35:3, 797–810. ISSN: 00431397.

DOI: 10.1029/1998WR900065. URL: http://doi.wiley.com/10.1029/1998WR900065 (see pages 38, 40).

Western, Andrew W., Sen Lin Zhou, Rodger B. Grayson, Thomas A. McMahon, Günter Blöschl, and David J. Wilson (2004). **Spatial correlation of soil moisture in small catchments and its relationship to dominant spatial hydrological processes**. *Journal of Hydrology* 286:1-4, 113–134. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2003.09.014 (see pages 16, 17, 97).

Wu, Wanru, Marvin A Geller, and Robert E Dickinson (2002). **The response of soil moisture to long-term variability of precipitation**. *Journal of Hydrometeorology* 3:5, 604–613 (see page 36).

Zehe, E., T. Graeff, M. Morgner, A. Bauer, and A. Bronstert (2010). **Plot and field scale soil moisture dynamics and subsurface wetness control on runoff generation in a headwater in the Ore Mountains**. *Hydrology and Earth System Sciences* 14:6, 873–889. ISSN: 10275606. DOI: 10.5194/hess-14-873-2010 (see pages 16, 17).

Zehe, E., R. Loritz, Y. Edery, and B. Berkowitz (2021). **Preferential pathways for fluid and solutes in heterogeneous groundwater systems: self-organization, entropy, work**. *Hydrology and Earth System Sciences* 25:10, 5337–5353. DOI: 10.5194/hess-25-5337-2021. URL: https://hess.copernicus.org/articles/25/5337/2021/ (see page 51).

Zehe, Erwin, Rolf Becker, András Bárdossy, and Erich Plate (Dec. 2005). **Uncertainty of simulated catchment runoff response in the presence of threshold processes: Role of initial soil moisture and precipitation**. *Journal of Hydrology* 315:1-4, 183–202. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2005.03.038 (see page 51).

Zehe, Erwin and Günter Blöschl (2004). **Predictability of hydrologic response at the plot and catchment scales: Role of initial conditions**. *Water Resources Research* 40:10 (see page 54).

Zehe, Erwin, U Ehret, L Pfister, Theresa Blume, Boris Schroeder, M Westhoff, C Jackisch, Stanislauv J Schymanski, M Weiler, K Schulz, et al. (2014). **HESS Opinions: From response units to functional units: a thermodynamic reinterpretation of the HRU concept to link spatial organization and functioning of intermediate scale catchments**. *Hydrology and Earth System Sciences* 18:11, 4635–4655 (see pages 17–19).

Zimmermann, B., E. Zehe, N. K. Hartmann, and H. Elsenbeer (2008). **Analyzing spatial data: An assessment of assumptions, new methods, and uncertainty using soil hydraulic data**. *Water Resources Research* 44:10, 1–18. ISSN: 00431397. DOI: 10.1029/2007WR006604 (see pages 66, 106, 130).

# List of Publications

## Articles in Refereed Journals

[1] **SciKit-GStat 1.0: a SciPy-flavored geostatistical variogram estimation toolbox written in Python**. *Geoscientific Model Development* 15:6, 2505–2532. DOI: 10.5194/gmd-15-2505-2022. URL: https://gmd.copernicus.org/articles/15/2505/2022/. Joint work with Mälicke, M..

[2] **Soil moisture: variable in space but redundant in time**. *Hydrology and Earth System Sciences* 24:5, 2633–2653. DOI: 10.5194/hess-24-2633-2020. URL: https://hess.copernicus.org/articles/24/2633/2020/. Joint work with Mälicke, M., S. K. Hassler, T. Blume, M. Weiler, and E. Zehe.

[3] **SciKit-GStat Uncertainty: A software extension to cope with uncertain geostatistical estimates**. *Spatial Statistics* 54, 100737. ISSN: 2211-6753. DOI: https://doi.org/10.1016/j.spasta.2023.100737. URL: https://www.sciencedirect.com/science/article/pii/S221167532300012X. Joint work with Mälicke, Mirko, Alberto Guadagnini, and Erwin Zehe.

[4] **Agroforestry: An Appropriate and Sustainable Response to a Changing Climate in Southern Africa?** *Sustainability* 12:17. ISSN: 2071-1050. DOI: 10.3390/su12176796. URL: https://www.mdpi.com/2071-1050/12/17/6796. Joint work with Sheppard, Jonathan P., Rafael Bohn Reckziegel, Lars Borrass, Paxie W. Chirwa, Claudio J. Cuaranhua, Sibylle K Hassler, Svenja Hoffmeister, Florian Kestel, Rebekka Maier, Mirko Mälicke, Christopher Morhart, Nicholas P. Ndlovu, Maik Veste, Roger Funk, Friederike Lang, Thomas Seifert, Ben du Toit, and Hans-Peter Kahle.

[5] **Histogram via entropy reduction (HER): an information-theoretic alternative for geostatistics**. *Hydrology and Earth System Sciences* 24:9, 4523–4540. Joint work with Thiesen, Stephanie, Diego M Vieira, Mirko Mälicke, Ralf Loritz, J Florian Wellmann, and Uwe Ehret.