MONASH University

New tools for visualising and explaining multivariate spatio-temporal data

H. Sherry Zhang

B.Comm. (Hons), Monash University

A thesis submitted for the degree of Doctor of Philosophy at Monash University

in 2023

Department of Econometrics & Business Statistics

Contents

Copyright notice

Abstract

A vast amount of data is generated in our society, encompassing an array of measurements, accompanied by location and time information. This multivariate spatio-temporal data is used to study important problems challenging today's world including climate change and disease spread. This research introduces new methods and frameworks with accompanying open-source software to help in the analysis of multivariate spatio-temporal data.

The first contribution (Chapter 2) delivers diagnostic plots designed to understand the optimisers for high-dimensional projection pursuit. It provides computational tools to track the optimisation progress and coverage of the parameter space. The second contribution (Chapter 3) develops a new data structure, called cubble, for organising spatio-temporal data. Spatio-temporal data are often split into multiple tables, each with different observation units, or organised into a memory-inefficient single table combining all the data. The new data structure organises the spatial and temporal components of the data into a single object, efficiently, and allows pivoting separately into the spatial and temporal tables for different analyses. The third contribution (Chapter 4) introduces a data pipeline for constructing indexes from multivariate spatio-temporal data. While indexes from various domains use similar statistical methodologies to summarize data into an index, a standardised workflow to systematically assemble indexes is absent. This work bridges this gap by offering a unified framework and infrastructure to modularise the steps in constructing an index.

Declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes one original paper published in a peer reviewed journal and one submitted publication. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the Department of Econometrics & Business Statistics under the supervision of Professor Dianne Cook, Dr. Patricia Menéndez, Dr. Ursula Laa, and Dr. Nicolas Langrené.

The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into team-based research. In the case of Chapter 2 to Chapter 4, my contribution to the work involved the following:

Chapter	**Publication title**	**Status**	**Student contribution**	**Co-authors contribution**	**Monash Student Yes/No'**
2	Visual Diagnostics for Constrained Optimisation with Application to Guided Tours	Published in the R Journal; The R package ferrn on CRAN	80% Concept, Analysis, Software, Writing	D. Cook 10%, P. Menéndez 5%, U. Laa 2.5%, N. Langrené 2.5%	No

(continued)

Chapter	Publication title	Status	Student contribution	Co-authors contribution	Monash Student Yes/No'
3	Cubble: An R Package for Organizing and Wrangling Multivariate Spatio-temporal Data	In revision for the Journal of Statistical Software; The R package cubble on CRAN	80% Concept, Analysis, Software, Writing	D. Cook 10%, P. Menéndez 5%, U. Laa 2.5%, N. Langrené 2.5%	No

Chapter 4, *A Tidy Framework and Infrastructure to Systematically Assemble Spatio-temporal Indexes from Multivariate Data* is to be submitted for the Journal of Computational and Graphical Statistics and the associated R package `tidyindex` is to be submitted to CRAN.

The thesis is written in Australian spelling, except for Chapters 3 and 4, which use American spelling as specified by the publication venue.

I have not renumbered sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student name: H. Sherry Zhang

Student signature:

Date: 23rd August 2023

I hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author I have consulted with the responsible author to agree on the respective contributions of the authors.

Main Supervisor name: Dianne Cook

Main Supervisor signature:

Date: 23rd August 2023

Acknowledgements

I would like to express my gratitude to my supervisors, Prof. Dianne Cook, Dr. Patricia Menéndez, Dr. Ursula Laa, and Dr. Nicolas Langrené. Thanks, Di, for first taking me as a summer research student, then as an honours student, and now as a PhD candidate. Your consistent optimism and unwavering support have made my journey a delightful experience, especially during challenging times. Looking back, I am amazed at how much you have influenced me academically and as a person. Thank you, Patricia, Ursula, and Nicolas, for always being there to help with mathematical notations and writing and it is my fortune to have had such a great supervision team!

I would also like to thank Weihao, Ze Yu, Cynthia, and recently Janith and Krisanat, for being my office mates all the way from Menzies to the Education building. I will miss complaining about coursework units, poking fun at our ugly plots, and going to lunch and snack breaks together. Thank you, Dr. Emi Tanaka, Dr. Jessica Leung, and Prof. Daniel Simpson for providing valuable feedback as my PhD panel members. Thank you, Prof. Catherine Forbes and Prof. Gael Martin, our PhD coordinator, for organising the book club, PhD meetings, and milestone events. Special thanks to the Monash NUMBATs group for organising the Hackathon trips to Abbotsford Convent and San Remo. Lastly, thank you, Dr. Ursula Laa, Prof. Catherine Hurley, and Prof. Dr. Edzer Pebesma for hosting me during my visit to Europe in 2023.

I would also like to thank my training buddies in the swimming squad, gym, climbing gym, and recently, squash team. Thank you Cynthia and Susanna for introducing me to climbing and always cheering me to conquer more challenging climbs on the wall – just as Hans does too.

Special thank goes to Mike for being a great companion. From grocery trips to holidays in Gold Coast, Brisbane, Perth, and Sydney, I enjoyed the beaches, quokkas, wildlife, and sea life with you—even if it meant sharing half of my room with you over the past months.

Thank you mum and dad for always being there for me at all times. Even though I have been a great distance away from home for over seven years (approximately 8000 km!), knowing that you have maintained my room just the way I left it fills me with appreciation and a warm feeling of connection.

This research was possible by financial support from a CSIRO Data61 scholarship and the Department of Econometrics and Business Statistics.

Chapter 1

Introduction

Multivariate spatio-temporal data are ubiquitous in our society. In finance and economics, stock prices and economic indicators are tracked over time; in logistics, supermarkets collect product level data to decide the optimal stock levels for different stores; in meteorology, weather stations record climate variables to monitor climate change and its impact on agriculture, human health, and natural disasters. Spatio-temporal data are observations recorded with time and geographic location information. Multivariate means that multiple variables (e.g. temperature, precipitation, wind speed and direction) are recorded. It is common for analytical methods to focus separately on the aspects – time series analysis addresses temporal trends, spatial analysis examines geographic patterns, and multivariate analysis models the relationship between the multiple measured variables. However, all the three aspects are ideally considered together to tackle contemporary problems, such as monitoring droughts which requires historical time-stamped data to understand "normal" conditions for any spatial neighborhood, and the interactions of precipitation, temperature and other variables. From multivariate spatio-temporal data, decision-makers compute indexes constructed from these components to inform the public and to determine when or if to take remedial action.

This research addresses the challenge of investigating multivariate spatio-temporal data together and also independently, by providing new tools for organising, visualising and explaining relationships. The illustration in Figure 1.1 shows how the three research topics are related and provide solutions. The solutions are to provide easy ways to pivot between the three components, to allow focusing on multivariate, or spatio-temporal components of the data

and a new data pipelines for constructing indexes for monitoring different aspects of our world using multiple variables. When fixing the time, the data are reduced to its spatial and multivariate elements. When the spatial component represents observations, it can be analysed using multivariate methods such as dimension reduction and the particular dimension reduction technique investigated in this thesis is called projection pursuit. When the data are collected at different locations in space, software from geo-informatics can be useful to analyse the spatial aspect of the data. However, existing spatial and temporal data analysis software are built upon different data formats. In order to combine spatial and temporal data for spatio-temporal analysis, the spatial data need to duplicate observations at each time point. However, these duplicates can lead to inefficiency for spatial analysis. This introduces the constant need to combine and separate the two components to align with the existing software, creating frictions in the data analysis. Multivariate spatio-temporal variables are often combined into a single series for each location to produce an index series which can be used for decision making or communicating conditions. But index definition and construction is vastly different in different fields and different researchers making it difficult to understand how they might perform with slight changes in the formula, or be affected by data quality issues and how competing indexes compare.

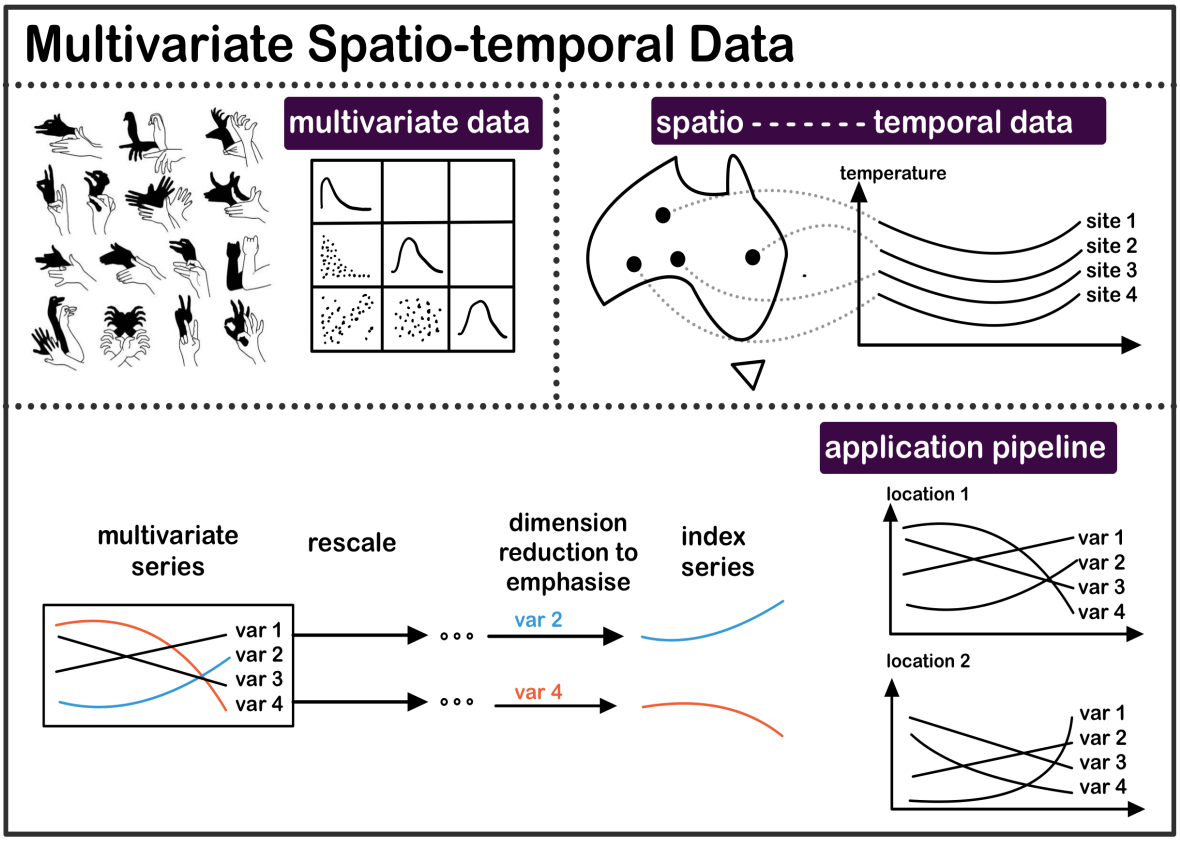

Figure 1.1: *A graphical summary of three directions to analyse multivariate spatio-temporal data in this thesis: 1) explore the multivariate relationship between variables with the time being fixed, 2) explore spatial and temporal characteristics simultaneously when the observations are collected at different locations, and 3) construct indexes with different designs to summarize multivariate information.*

1.1 Visual diagnostics for projection pursuit optimization

Many data, despite having different multivariate, spatial, and temporal features, can all be categorised as multivariate spatio-temporal data. When there are only a few time snaps in the data, we may treat each time snap as independent and apply multivariate methods to analyze the variable relationship. Bivariate relationships, either linear or non-linear, can be represented in a scatterplot matrix, however, it becomes complex when a certain relationship is attributed to three or more variables. A dimension reduction technique called "projection pursuit" can be used to find interesting structures in multivariate data by linear projection. Combined with a visualisation technique called guided tour, it can show the data points smoothly transiting from randomness to some interesting structures found by the algorithm. In projection pursuit, multivariate data is transformed into a low dimensional space, typically 1D or 2D, using a projection matrix. Each projection matrix corresponds to a projection of the data, on which

statistics, also called the index functions, can be computed. The projection pursuit algorithm optimises the statistics on the set of orthonormal matrices to detect interesting patterns in the data, such as clusters or outlines. In practice, however, the optimiser sometimes does not always work as desired: it may fail unexpectedly, gets stuck at a local maximum, or approach the maximum without reaching it. In this work, four diagnostic plots are proposed to track the optimization algorithms in projection pursuit.

1.2 Cubble: A new spatio-temporal data structure

When multivariate spatio-temporal data contain only a collection of variables, the spatial and temporal dimensions remain to be explored. Weather station data is one such example, where the number of variables recorded depends on the instruments installed, while stations are widely distributed spatially and daily data are available over years. Spatial and temporal data analysis each provide tools for examining one dimension of the data, however, when working with spatio-temporal data, researchers may switch their data among different forms (pure spatial, pure temporal, or a combined table) to analyse the data. This presents a unique task of coordinating the data and results from different formats, which is not the case when the data all have a single observational unit. While the actual process to reshape the data may not be difficult for a given audience, this repeated requirement to reorganise the data is disruptive from a workflow perspective, forcing researchers to pause on the actual data analysis and turn to transforming among different data formats. This research addresses this problem by proposing a new data structure to organise spatio-temporal data in **R** so that different spatial and temporal information can be easily accessed for exploratory data analysis.

1.3 A tidy framework for indexes

Multivariate spatio-temporal data can also be analysed as multivariate time series with fixed observations. To visualize and explain this collection of multivariate time series, indexes can be constructed to monitor the joint effect of multiple variables over time or to compare information from different observations. Examples of such indexes can be found in monitoring the environment (i.e. drought indexes and water quality indexes), measuring social development (i.e. human development index and gender equality index), and making decisions on resource allocation. While research institutes and government publish index values calculated according

to standard practice, information should be made available to understand how the index may behave under different data conditions and its implication for decision making. In this work, we develop a general data pipeline to construct spatio-temporal indexes from multivariate data. This provides researchers with a standard framework for constructing and analysing indexes, including experimenting different parameter choices, adjusting steps in the index definition, calculating uncertainty, and assessing index robustness. The design of the pipeline framework is aligned with the tidy framework adopted by the tidyverse and tidymodel, allowing the construction and analysis of indexes in a unified syntax, regardless of their application domains.

1.4 Thesis overview

The rest of the thesis is organized as follows: Chapter 2 presents the proposed visual diagnostics plots designed to assess the optimisation in projection pursuit guided tour, along with the R implementation, `ferrn` (Zhang et al. 2021). In Chapter 3, a novel data structure and the R package, `cubble` (Zhang et al. 2023a), is introduced to organise spatio-temporal data, with examples given to demonstrate its use in analysing weather station data. Chapter 4 proposes a framework for constructing spatio-temporal indexes from data and the resulting data pipeline is implemented in the package `tidyindex` (Zhang et al. 2023b). Chapter 5 concludes the thesis and discusses potential future directions.

Chapter 2

Visual Diagnostics for Constrained Optimisation with Application to Guided Tours

A guided tour helps to visualise high-dimensional data by showing low-dimensional projections along a projection pursuit optimisation path. Projection pursuit is a generalisation of principal component analysis in the sense that different indexes are used to define the interestingness of the projected data. While much work has been done in developing new indexes in the literature, less has been done on understanding the optimisation. Index functions can be noisy, might have multiple local maxima as well as an optimal maximum, and are constrained to generate orthonormal projection frames, which complicates the optimization. In addition, projection pursuit is primarily used for exploratory data analysis, and finding the local maxima is also useful. The guided tour is especially useful for exploration because it conducts geodesic interpolation connecting steps in the optimisation and shows how the projected data changes as a maxima is approached. This work provides new visual diagnostics for examining a choice of optimisation procedure based on the provision of a new data object which collects information throughout the optimisation. It has helped to diagnose and fix several problems with projection pursuit guided tour. This work might be useful more broadly for diagnosing optimisers and comparing their performance. The diagnostics are implemented in the R package `ferrn`.

2.1 Introduction

Visualisation is widely used in exploratory data analysis (Tukey 1977; Unwin 2015; Healy 2018; Wilke 2019). Presenting information in graphics often unveils insights that would otherwise not be discovered and provides a more comprehensive understanding of the problem at hand. Task-specific tools such as Li, Zhao, and Scheidegger (2020) show how visualisation can be used to understand, for instance, the behaviour of the optimisation for the example of neural network classification models. However, no general visualisation tool is available for diagnosing optimisation procedures. The work presented in this paper brings visualization tools into optimisation problems with the aim to better understand the performance of optimisers in practice.

The focus of this paper is on the optimisation problem arising in the projection pursuit guided tour (Buja et al. 2005), an exploratory data analysis technique used for detecting interesting structures in high-dimensional data through a set of lower-dimensional projections (Cook et al. 2008). The goal of the optimisation is to identify the projection, represented by the projection matrix, that gives the most interesting low-dimensional view. A view is said to be interesting if it can show some structures of the data that depart from normality, such as bimodality, clustering, or outliers.

The optimization challenges encountered in the projection pursuit guided tour problem are common to those of optimization in general. Examples include the existence of multiple optima (local and global), the trade-off between computational burden and proximity to the optima, or dealing with noisy objective functions that might be non-smooth and non-differentiable (D. Jones, Schonlau, and Welch 1998). The visualization tools, optimization methods, and conceptual framework presented in this paper can therefore be applied to other optimization problems.

The remainder of the paper is organised as follows. The next section provides an overview of optimisation methods, specifically random search and line search methods. A review of the projection pursuit guided tour, an overview of the optimisation problem and, outlines of three existing algorithms follows. The third section presents the new visual diagnostics, including the design of a data structure to capture information during the optimisation, from which several diagnostic plots are created. An illustration of how the diagnostic plots can be used to examine the performance of different optimisers and guide improvements to existing algorithms is shown

using simulated data. Finally, an explanation of the implementation in the R package, `ferrn` (Zhang et al. 2021), is provided.

2.2 Optimisation methods

The type of optimisation problem considered in this paper is constrained optimization (Bertsekas 2014), assuming it is not possible to find a solution to the problem in the way of a closed-form. That is, the problem consists in finding the minimum or maximum of a function $f \in L^p$ in the constrained space $\mathcal{A}$, where L^p defines the vector space of function f, whose pth power is integrable.

Gradient-based methods are commonly used to optimise an objective function, with the most notable one being the gradient ascent (descent) method. Although these methods are popular, they rely on the availability of the objective function derivatives. As will be shown in the next section, the independent variables in our optimisation problem are the entries of a projection matrix, and the computational time required to perform differentiation on a matrix could impede the rendering of tour animation. In addition, some objective functions rely on the empirical distribution of the data, which makes it in general not possible to get the gradient. Hence, gradient-based methods are not the focus of this paper, and consideration will be given to derivative-free methods.

Derivative-free methods (Conn, Scheinberg, and Vicente 2009; Rios and Sahinidis 2013), which do not rely on the knowledge of the gradient, are more generally applicable. Derivative-free methods have been developed over the years, where the emphasis is on finding, in most cases, a near-optimal solution. Here we consider three derivative-free methods, two of which are random search methods: creeping random search and simulated annealing, and the other one is pseudo-derivative search.

Random search methods (Romeijn 2009; Zabinsky 2013; Andradóttir 2015) have a random sampling component as part of their algorithms and have been shown to have the ability to optimise non-convex and non-smooth functions. The initial random search algorithm, pure random search (Brooks 1958), draws candidate points from the entire space without using any information of the current position and updates the current position when an improvement on the objective function is made. As the dimension of the space becomes larger, sufficient sampling from the entire space would require a long time for convergence to occur, despite a

guaranteed global convergence (Spall 2005). Various algorithms have thus been developed to improve pure random search by either concentrating on a narrower sampling space or using a different updating mechanism. Creeping random search (White 1971) is such a variation, where a candidate point is generated within a neighbourhood of the current point. This makes creeping random search faster to compute but global convergence is no longer guaranteed. On the other hand, simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983; Bertsimas and Tsitsiklis 1993), introduces a different updating mechanism. Rather than only updating the current point when an improvement is made, it uses a Metropolis acceptance criterion, where worse candidates still have a chance to be accepted. The convergence of simulated annealing algorithms has been widely researched (Mitra, Romeo, and Sangiovanni-Vincentelli 1986; Granville, Krivánek, and Rasson 1994) and the global optimum can be attained under mild regularity conditions.

The pseudo-derivative search uses a common search scheme in optimisation: line search. In line search methods, users are required to provide an initial estimate x_1 and, at each iteration, a search direction S_k and a step size α_k are generated. Then one moves on to the next point following $x_{k+1} = x_k + \alpha_k S_k$ and the process is repeated until the desired convergence is reached. In derivative-free methods, local information of the objective function is used to determine the search direction. The choice of step size also needs consideration, as inadequate step sizes might prevent the optimisation method from converging to an optimum. An ideal step size can be chosen by finding the value of $\alpha_k \in \mathbb{R}$ that maximises $f(x_k + \alpha_k S_k)$ with respect to α_k at each iteration.

2.3 Projection pursuit guided tour

A projection pursuit guided tour combines two different methods (projection pursuit and guided tour) to explore interesting features in a high-dimensional space. Projection pursuit, coined by Friedman and Tukey (1974), detects interesting structures (e.g., clustering, outliers, and skewness) in multivariate data via low-dimensional projections. Guided tour (Cook et al. 1995) is one variation of a broader class of data visualisation methods, tour (Buja et al. 2005), which displays high-dimensional data through a series of animated projections.

Let $\mathbf{X}_{n\times p}$ be the data matrix with n observations in p dimensions. A d-dimensional projection is a linear transformation from $\mathbb{R}^p$ into $\mathbb{R}^d$ defined as $\mathbf{Y} = \mathbf{X} \cdot \mathbf{A}$, where $\mathbf{Y}_{n\times d}$ is the projected data and $\mathbf{A}_{p\times d}$ is the projection matrix. We define $f : \mathbb{R}^{n\times d} \mapsto \mathbb{R}$ to be an index function that maps

the projected data $\mathbf{Y}$ onto a scalar value. This is commonly known as the projection pursuit index function, or just index function, and is used to measure the "interestingness" of a given projection. An interesting projection shows structures that are non-normal since theoretical proofs from Diaconis and Freedman (1984) have shown that projections tend to be normal as n and p approach infinity under certain conditions. There have been many index functions proposed in the literature, here are a few examples: early indexes that can be categorised as measuring the L^2 distance between the projection and a normal distribution: Legendre index (Friedman and Tukey 1974); Hermite index (Hall 1989); natural Hermite index (Cook, Buja, and Cabrera 1993); chi-square index (Posse 1995) for detecting spiral structure; LDA index (Lee et al. 2005) and PDA (Lee and Cook 2010) index for supervised classification; kurtosis index (Loperfido 2020) and skewness index (Loperfido 2018) for detecting outliers in financial time series; and most recently, scagnostic indexes (Laa and Cook 2020) for summarising structures in scatterplot matrices based on eight scagnostic measures (Wilkinson, Anand, and Grossman 2005; Wilkinson and Wills 2008).

As a general visualisation method, tour produces animations of high-dimensional data via rotations of low-dimensional planes. There are different versions depending on how the high-dimensional space is investigated: grand tour (Cook et al. 2008) selects the planes randomly to provide a general overview; manual tour (Cook and Buja 1997) gradually phases in and out one variable to understand the contribution of that variable in the projection. Guided tour, the main interest of this paper, chooses the planes with the aid of projection pursuit to gradually reveal the most interesting projection. Given a random start, projection pursuit iteratively finds bases with higher index values, and the guided tour constructs a geodesic interpolation between these planes to form a tour path. Figure 2.1 shows a sketch of the tour path where the blue squares represent planes (targets) selected by the projection pursuit optimisation, and the white squares represent planes in the geodesic interpolation between targets. Mathematical details of the geodesic interpolation can be found in Buja et al. (2005). (Note that the term *frame* used in Buja's paper refers to a particular set of orthonormal vectors defining a plane. This is also conventionally referred to as a basis, which is used in this paper and the associated software.) The aforementioned tour method has been implemented in the R package `tourr` (Wickham et al. 2011).

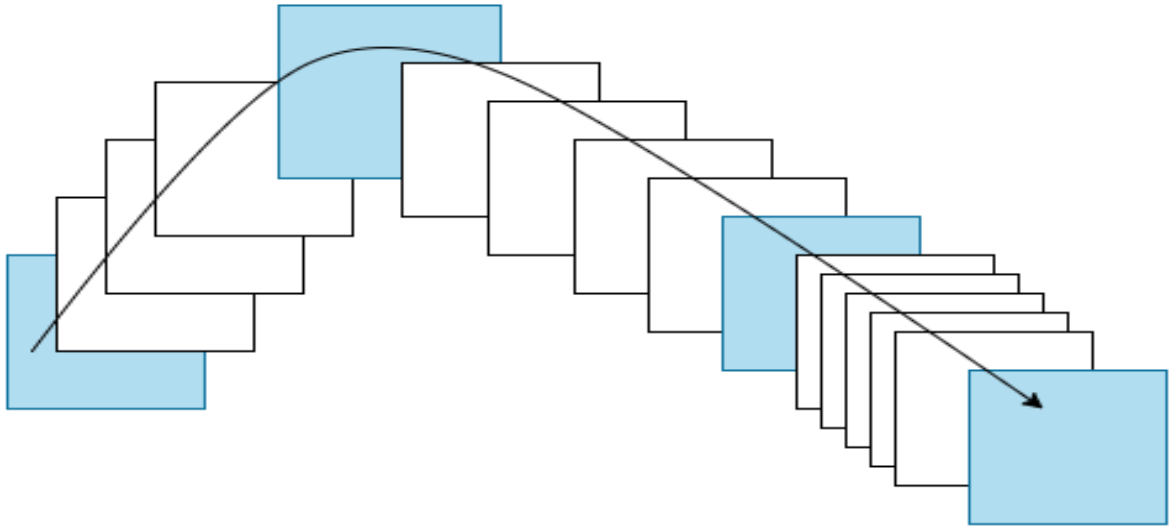

Figure 2.1: *An illustration for demonstrating the frames in a tour path. Each square (frame) represents the projected data with a corresponding basis. Blue frames are returned by the projection pursuit optimisation and white frames are constructed between two blue frames by geodesic interpolation.*

2.3.1 Optimisation in the tour

In projection pursuit, the optimisation aims at finding the global and local maxima that give interesting projections according to an index function. That is, it starts with a given randomly selected basis $\mathbf{A}_1$ and aims at finding an optimal final projection basis $\mathbf{A}_T$ that satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{X}\cdot\mathbf{A}) \quad s.t. \quad \mathbf{A}'\mathbf{A} = I_d\,, \tag{2.1}$$

where f and $\mathbf{X}$ are defined as in the previous section, $\mathcal{A}$ is the set of all p-dimensional projection bases, I_d is the d-dimensional identity matrix, and the constraint ensures the projection bases, $\mathbf{A}$, to be orthonormal. It is worth noticing the following: 1) The optimisation is constrained, and the orthonormality constraint imposes a geometrical structure on the bases space: it forms a Stiefel manifold. 2) There may be index functions for which the objective function might not be differentiable. 3) While finding the global optimum is the goal of the optimisation problem, interesting projections may also appear in the local optimum. 4) The optimisation should be fast to compute since the tour animation is viewed by the users during the optimisation.

2.3.2 Existing algorithms

Three optimisers have been implemented in the `tourr` (Wickham et al. 2011) package: creeping random search (CRS), simulated annealing (SA), and pseudo-derivative (PD). Creeping random search (CRS) is a random search optimiser that samples a candidate basis $\mathbf{A}_l$ in the neighbourhood of the current basis $\mathbf{A}_{\text{cur}}$ by $\mathbf{A}_l = (1-\alpha)\mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{\text{rand}}$ where $\alpha \in [0,1]$ controls the radius of the sampling neighbourhood and $\mathbf{A}_{\text{rand}}$ is generated randomly. $\mathbf{A}_l$ is then orthonormalised to fulfil the basis constraint. If $\mathbf{A}_l$ has an index value higher than the current basis $\mathbf{A}_{\text{cur}}$, the optimiser outputs $\mathbf{A}_l$ for a guided tour to construct an interpolation path. The neighbourhood parameter α is adjusted by a cooling parameter: $\alpha_{j+1} = \alpha_j * \text{cooling}$ before the next iteration starts. The optimiser terminates when the maximum number of iteration $l_{\max}$ is reached before a better basis can be found. The algorithm of CRS can be found in the appendix. Posse (1995) has proposed a slightly different cooling scheme by introducing a halving parameter c. In his proposal, α is only adjusted if the last iteration takes more than c times to find a better basis.

Simulated annealing (SA) uses the same sampling process as CRS but allows a probabilistic acceptance of a basis with lower index value than the current one. Given an initial value of $T_0 \in \mathbb{R}^+$, the "temperature" at iteration l is defined as $T(l) = \frac{T_0}{\log(l+1)}$. When a candidate basis fails to have an index value larger than the current basis, SA gives it a second chance to be accepted with probability

$$P = \min\left\{\exp\left[-\frac{|\, I_{\text{cur}} - I_l \,|}{T(l)}\right], 1\right\},$$

where $I_{(\cdot)} \in \mathbb{R}$ denotes the index value of a given basis. This implementation allows the optimiser to make a move and explore the basis space even if the candidate basis does not have a higher index value. Hence it enables the optimiser to jump out of a local optimum. The second algorithm in the appendix highlights how SA differs from CRS in the inner loop.

Pseudo-derivative (PD) search uses a different strategy than CRS and SA. Rather than randomly sample the basis space, PD first computes a search direction by evaluating bases close to the current basis. The step size is then chosen along the corresponding geodesic by another optimisation over a 90 degree angle from $-\pi/4$ to $\pi/4$. The resulting candidate basis $\mathbf{A}_{**}$ is returned for the current iteration if it has a higher index value than the current one. The third algorithm in the appendix summarises the inner loop of the PD.

2.4 Visual diagnostics

A data structure for diagnosing optimisers in projection pursuit guided tour is first defined. With this data structure, four types of diagnostic plots are presented.

2.4.1 Data structure for diagnostics

Three main pieces of information are recorded during the projection pursuit optimisation: 1) projection bases $\mathbf{A}$, 2) index values I, and 3) state S. For CRS and SA, possible states include `random_search`, `new_basis`, and `interpolation`. Pseudo-derivative (PD) has a wider variety of states, including `new_basis`, `direction_search`, `best_direction_search`, `best_line_search`, and `interpolation`. Multiple iterators index the information collected at different levels: t is a unique identifier prescribing the natural ordering of each observation; j and l are the counter of the outer and inner loop, respectively. Other parameters of interest recorded, V, include `method` that tags the name of the optimiser, and `alpha` that indicates the sampling neighbourhood size for searching observations. A matrix notation of the data structure is presented as follows:

t	$\mathbf{A}$	I	S	j	l	V_1	V_2	Description
1	$\mathbf{A}_1$	I_1	S_1	1	1	V_{11}	V_{12}	start basis
2	$\mathbf{A}_2$	I_2	S_2	2	1	V_{21}	V_{22}	search
…	…	…	…	…	…	…	…	…
…	…	…	…	2	l_2	…	…	search (accepted)
…	…	…	…	2	1	…	…	interpolation
…	…	…	…	…	…	…	…	…
…	…	…	…	2	k_2	…	…	interpolation
…	…	…	…	…	…	…	…	…
…	…	…	…	J	1	…	…	search
…	…	…	…	…	…	…	…	…
T	$\mathbf{A}_T$	I_T	S_T	J	l_J	V_{T1}	V_{T2}	search (final)
…	…	…	…	J	1	…	…	interpolation
…	…	…	…	…	…	…	…	…
…	…	…	…	J	k_J	…	…	interpolation
…	…	…	…	$J+1$	1	…	…	search (last round)
…	…	…	…	…	…	…	…	…
T'	$\mathbf{A}_{T'}$	$I_{T'}$	$S_{T'}$	$J+1$	l_{J+1}	$V_{T'1}$	$V_{T'2}$	search (last round)

where $T' = T + k_J + l_{J+1}$. Note that there is no output in iteration $J + 1$ since the optimiser does not find a better basis in the last iteration and terminates. The final basis found is A_T with index value I_T.

The data structure constructed above meets the tidy data principle (Wickham 2014) that requires each observation to form a row and each variable to form a column. With tidy data structure, data wrangling and visualisation can be significantly simplified by well-developed packages such as `dplyr` (Wickham et al. 2022) and 'ggplot2" (Wickham 2016).

2.4.2 Diagnostic 1: Checking how hard the optimiser is working

A starting point of diagnosing an optimiser is to understand how many searches it has conducted, i.e., we want to summarise how the index is increasing over iterations and how many basis need to be sampled at each iteration. This is achieved using the function `explore_trace_search()`: a boxplot shows the distribution of index values for each try, where the accepted basis (corresponding to the highest index value) is always shown as a point. When there are only few tries at a given iteration, showing the data points directly is preferred over the boxplot and this is controlled via the `cutoff` argument. Additional annotations are added to facilitate better reading of the plot, and these include 1) the number of points searched in each iteration can be added as text label at the bottom of each iteration; 2) the anchor bases to interpolate are connected and highlighted in a larger size; 3) the colour of the last iteration is in greyscale to indicate no better basis found in this iteration.

Figure 2.2 shows an example of the search plot for CRS (left) and SA (right). Both optimisers quickly find better bases in the first few iterations and then take longer to find one in the later iterations. The anchor bases, the ones found with the highest index value in each iteration, always have an increased index value in the optimiser CRS while this is not the case for SA. This feature gives CRS an advantage in this simple example to quickly find the optimum.

2.4.3 Diagnostic 2: Examining the optimisation progress

Another interesting feature to examine is the changes in the index value between interpolating bases since the projection on these bases is shown in the tour animation. Trace plots are created by plotting the index value against time. Figure 2.3 presents the trace plot of the same optimisations as Figure 2.2, and one can observe that the trace is smooth in both cases. It may seem bizarre

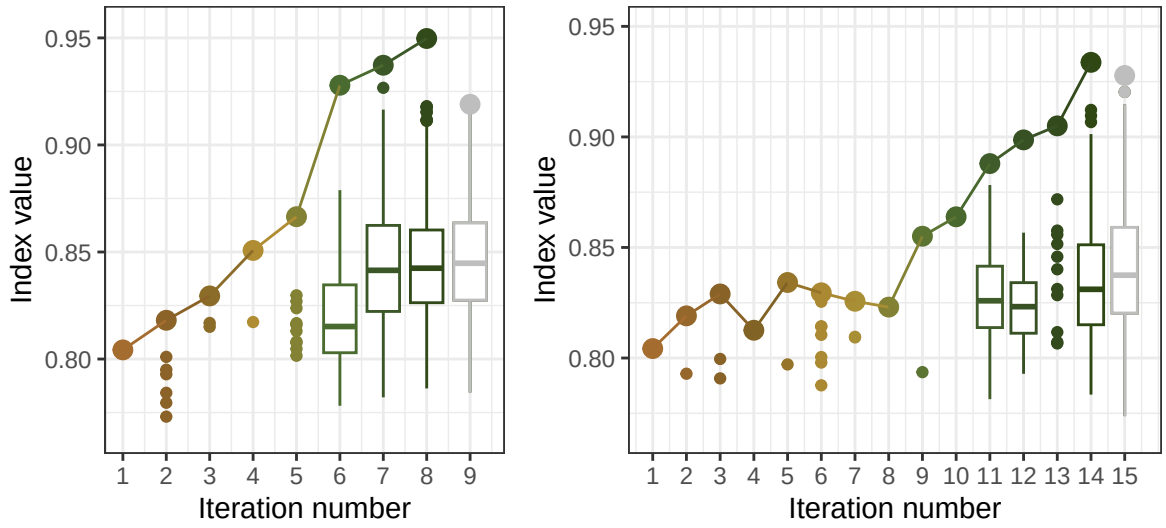

Figure 2.2: *A comparison of the searches by two optimisers: CRS (left) and SA (right) on a 2D projection problem of a six-variable dataset,* `boa6` *using the holes index. Both optimisers reach the final basis with a similar index value, while it takes SA longer to find the final basis. In the earlier iterations, optimisers quickly find a better basis to proceed, while in the later iterations, most sampled bases fail to make an improvement on the index value, and a boxplot is used to summarise the distribution of the index values. There is no better basis found in the last iteration, 9 (left) and 15 (right), before reaching the maximum number of try and hence it is colored grey. The color scale is from the customised botanical palette in the* `ferrn` *package.*

at first sight that the interpolation sometimes passes bases with higher index values before it decreases to a lower target. This happens because, on the one hand, the probabilistic acceptance in SA implies that some worse bases will be accepted by the optimiser. In addition, the guided tour interpolates between the current and target basis to provide a smooth transition between projections, and sometimes a higher index value will be observed along the interpolation path. This indicates that a non-monotonic interpolation cannot be avoided, even for CRS. Later, in Section *A problem of non-monotonicity*, there will be a discussion on improving the non-monotonic interpolation for CRS.

2.4.4 Diagnostic 3a: Understanding the optimiser's coverage of the search space

Apart from checking the search and progression of an optimiser, looking at where the bases are positioned in the basis space is also of interest. Given the orthonormality constraint, the space of projection bases $\mathbf{A}_{p\times d}$ is a Stiefel manifold. For one-dimensional projections, this forms a p-dimensional sphere. A dimensionality reduction method, e.g., principal component analysis, is applied to first project all the bases onto a 2D space. In a projection pursuit guided tour

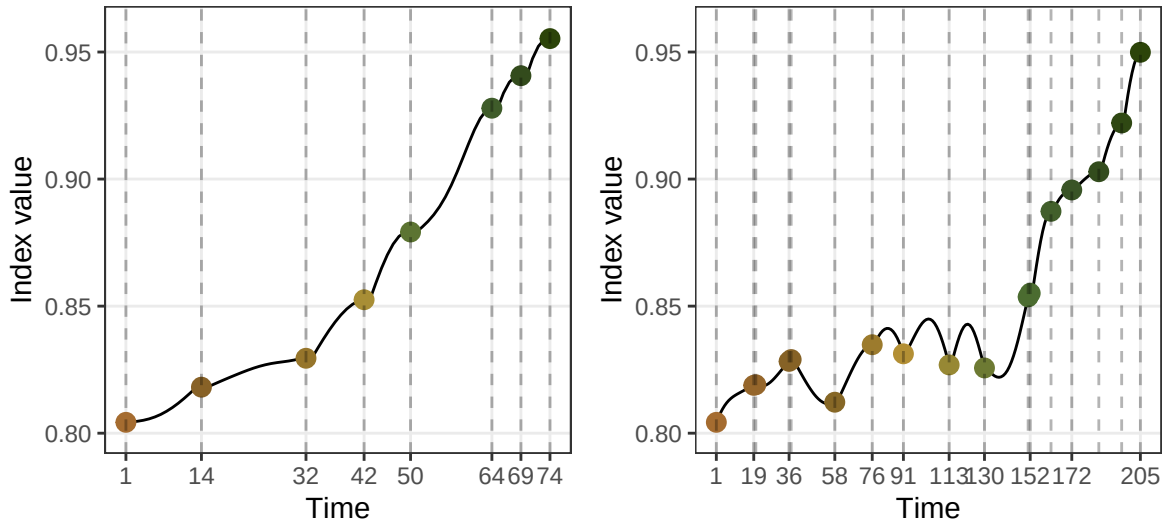

Figure 2.3: *An inspection of the index values as the optimisation progress for two optimisers: CRS (left) and SA (right). The holes index is optimised for a 2D projection problem on the six-variable dataset* `boa6`*. Lines indicate the interpolation, and dots indicate new target bases generated by the optimisers. Interpolation in both optimisation is smooth, while SA is observed to first pass by some bases with higher index values before reaching the target bases in time 76-130.*

optimisation, there are various types of bases involved: 1) The starting basis; 2) The search bases that the optimiser evaluated to produce the anchor bases; 3) The anchor bases that have the highest index value in each iteration; 4) The interpolating bases on the interpolation path; and finally, 5) the end basis. The importance of these bases differs but the most important ones are the starting, interpolating, and end bases. Sometimes, two optimisers can start with the same basis but finish with bases of opposite signs. This happens because the projection is invariant to the orientation of the basis, and so is the index value. However, this creates difficulties for comparing the optimisers since the end bases will be symmetric to the origin. A sign flipping step is conducted to flip the signs of all the bases in one routine if different optimisations finish at opposite places.

Several annotations have been made to help understand this plot. As mentioned previously, the original basis space is a high-dimensional sphere, and random bases on the sphere can be generated via the `geozoo` (Schloerke 2016) package. We use PCA to project and visualize the parameters/ bases in 2D. The centre of the 2D view is the first two PCs of the data matrix. It theoretically should be a circle but may have some irregular edges due to finite sampling. Thus the edge is smoothed by using a radius estimated as the largest distance from the centre to any basis. In the simulation, the theoretical best basis is known and can be labelled to compare

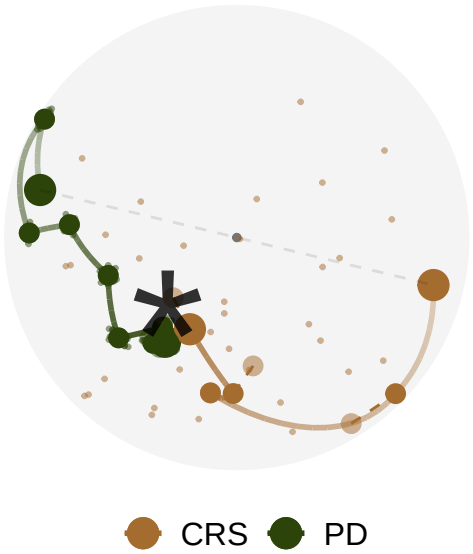

Figure 2.4: *Search paths of CRS (brown) and PD (green) in the PCA-reduced basis space for 1D projection problem on the five-variable dataset,* `boa5` *using holes index. The basis space, a 5D unit sphere, is projected onto a 2D circle by PCA. The black star represents the theoretical best basis the optimisers are aiming to find. All the bases in PD have been flipped for easier comparison of the final bases, and a grey dashed line has been annotated to indicate the symmetry of the two start bases.*

how close to this that the optimisers stopped. Various aesthetics, i.e., size, alpha (transparency), and colour, are applicable to emphasize critical elements and adjust for the presentation. For example, anchor points and search points are less important, and hence a smaller size and alpha are used. Alpha can also be applied on the interpolation paths to show start to finish from transparent to opaque.

Figure 2.4 shows the PCA plot of CRS and PD for a 1D projection problem. Both optimisers find the optimum, but PD gets closer. With the PCA plot, one can visually appreciate the nature of these two optimisers: PD first evaluates points in a small neighbourhood for a promising direction, while CRS evaluates points randomly in the search space to search for the next target. There are dashed lines annotated for CRS, and it describes the interruption of the interpolation, which will be discussed in detail in Section *A problem of non-monotonicity*.

2.4.5 Diagnostic 3b: Animating the diagnostic plots

Animation is another type of display to show how the search progresses from start to finish in the space. Figure 2.5 shows the animated version (six frames from the animation if viewed in pdf) of the PCA plot in Figure 2.4. An additional piece of information one can learn from this

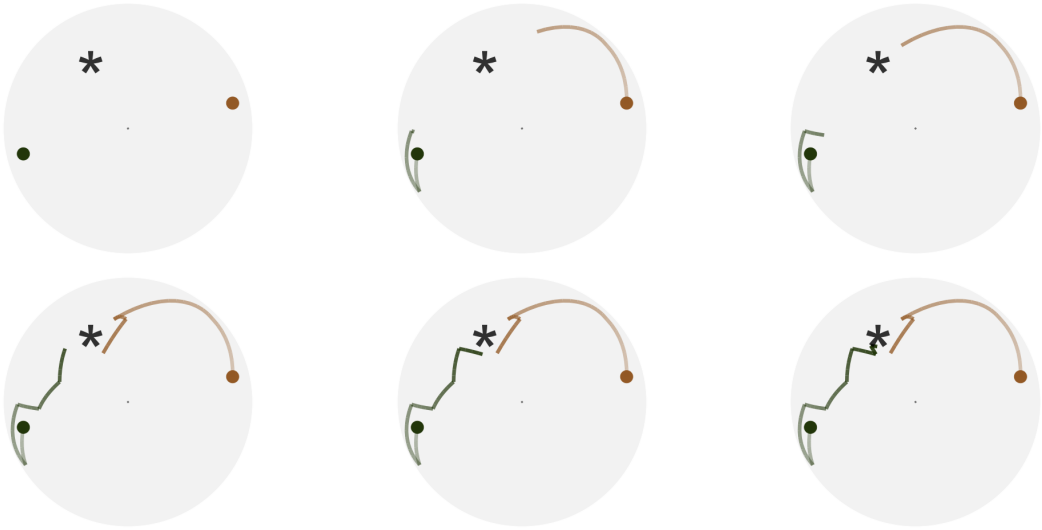

Figure 2.5: *Six frames selected from the animated version of the previous plot. With animation, the progression of the search paths from start to finish is better identified. CRS (brown) finishes the optimisation quicker than PD (green) since there is no further movement for CRS in the sixth frame. The full video of the animation can be found in the html version of the paper.*

animation is that CRS finds its end basis quicker than PD since CRS finishes its search in the 5th frame while PD is still making more progress.

2.4.6 Diagnostic 4a: The tour looking at itself

As mentioned previously, the original $p \times d$ dimension space can be simulated via randomly generated bases in the `geozoo` (Schloerke 2016) package. While the PCA plot projects the bases from the direction that maximises the variance, the tour plot displays the original high-dimensional space from various directions using animation. Figure 2.6 shows some frames from the tour plot of the same two optimisations in its original space.

2.4.7 Diagnostic 4b: Forming a torus

While the previous few examples have looked at the space of 1D basis in a unit sphere, this section visualises the space of 2D basis. Recall that the columns in a 2D basis are orthogonal to each other, so the space of $p \times 2$ bases is a torus in the p-D space (Buja and Asimov 1986). For $p = 3$ one would see a classical 3D torus shape as shown by the grey points in Figure 2.7. The two circles of the torus can be observed to be perpendicular to each other and this can be linked back to the orthogonality condition. Two paths from CRS and PD are plotted on top of

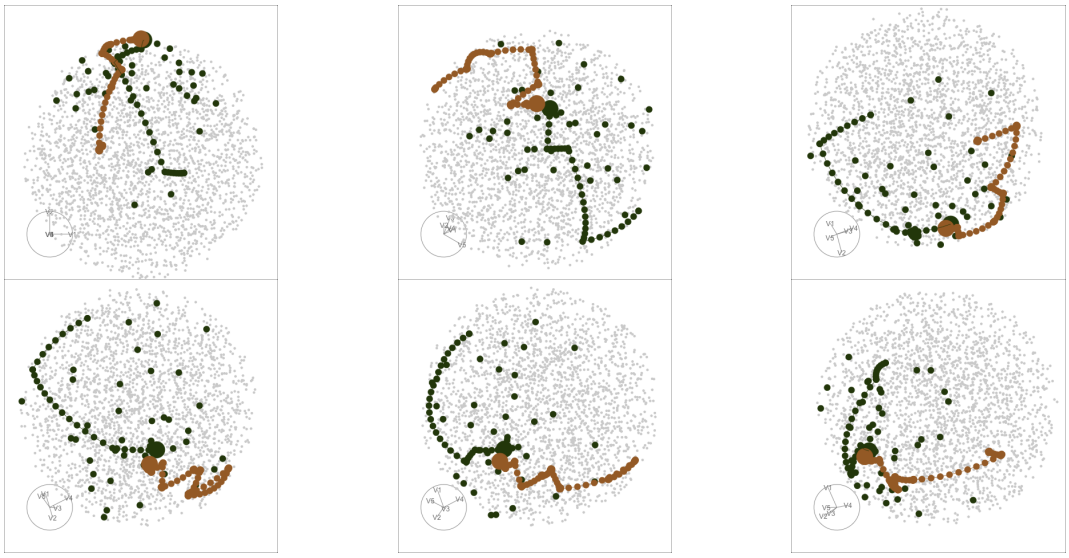

Figure 2.6: *Six frames selected from rotating the high-dimensional basis space, along with the same two search paths from Figure 2.4 and Figure 2.5. The basis space in this example is a 5D unit sphere, on which points (grey) are randomly generated via the CRAN package geozoo. The full animation can be seen in the html version of the paper.*

the torus and coloured in green and brown, respectively, to match the previous plots. The final basis found by PD and CRS are shown in a larger shape and printed below, respectively:

```
             [,1]        [,2]
[1,]  0.001196285  0.03273881
[2,] -0.242432715  0.96965761
[3,] -0.970167484 -0.24226493
```

```
            [,1]         [,2]
[1,]  0.05707994 -0.007220138
[2,] -0.40196202 -0.915510160
[3,] -0.91387549  0.402230054
```

Both optimisers have found the third variable in the first direction and the second variable in the second direction. Note, however, the different orientation of the basis, following from the different sign in the second column. One would expect to see this in the torus plot as the final bases match each other when projected onto one torus circle (due to the same sign in the first column) and are symmetric when projected onto the other (due to the different sign in the

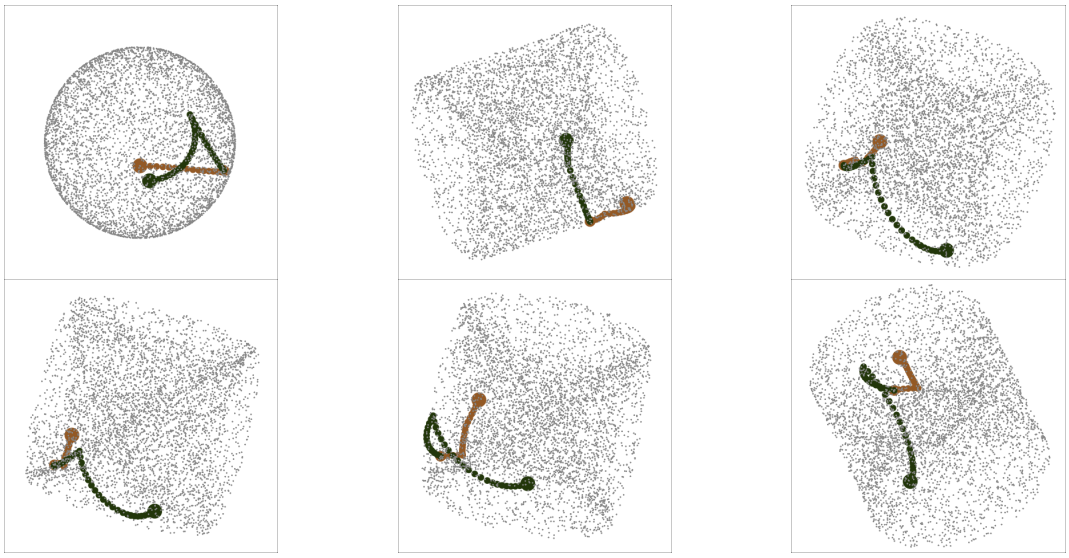

Figure 2.7: *Six frames selected from rotating the 2D basis space along with two search paths optimised by PD (brown) and CRS (green). The projection problem is a 2D projection with three variables using the holes index. The grey points are randomly generated 2D projection bases in the space, and it can be observed that these points form a torus. The full video of the animation can be found in the html version of the paper.*

second column). In Figure 2.7, this can be seen most clearly in frame 5 where the two circles are rotated into a line from our view.

2.5 Diagnosing an optimiser

In this section, several examples will be presented to show how the diagnostic plots discover something unexpected in projection pursuit optimisation, and guide the implementation of new features.

2.5.1 Simulation setup

Random variables with different distributions have been simulated as follows:

$$x_1 \overset{d}{=} x_8 \overset{d}{=} x_9 \overset{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{2.2}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{2.3}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{2.5}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{2.6}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{2.7}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{2.8}$$

Variables `x1`, `x8` to `x10` are normal noise with zero mean, and unit variance and `x2` to `x7` are normal mixtures with varied weights and locations. All the variables have been scaled to have overall unit variance before projection pursuit. The holes index (Cook et al. 2008), used for detecting bimodality of the variables, is used throughout the examples unless otherwise specified.

2.5.2 A problem of non-monotonicity

An example of non-monotonic interpolation has been given in Figure 2.3: a path that passes bases with a higher index value than the target one. For SA, a non-monotonic interpolation is justified since target bases do not necessarily have a higher index value than the current one, while this is not the case for CRS. The original trace plot for a 2D projection problem, optimised by CRS, is shown on the left panel of Figure 2.8, and one can observe that the non-monotonic interpolation has undermined the optimiser to realise its full potential. Hence, an interruption is implemented to stop at the best basis found in the interpolation. The right panel of Figure 2.8 shows the trace plot after implementing the interruption, and while the first two interpolations are identical, the basis at time 61 has a higher index value than the target in the third interpolation. Rather than starting the next iteration from the target basis on time 65, CRS starts the next iteration at time 61 on the right panel and reaches a better final basis.

2.5.3 Close but not close enough

Once the final basis has been found by an optimiser, one may want to push further in the close neighbourhood to see if an even better basis can be found. A polish search takes the final basis of an optimiser as the start of a new guided tour to search for local improvements. The polish algorithm is similar to the CRS but with three distinctions: 1) a hundred rather than one candidate bases are generated each time in the inner loop; 2) the neighbourhood size is reduced in the inner loop, rather than in the outer loop; and 3) three more termination conditions have

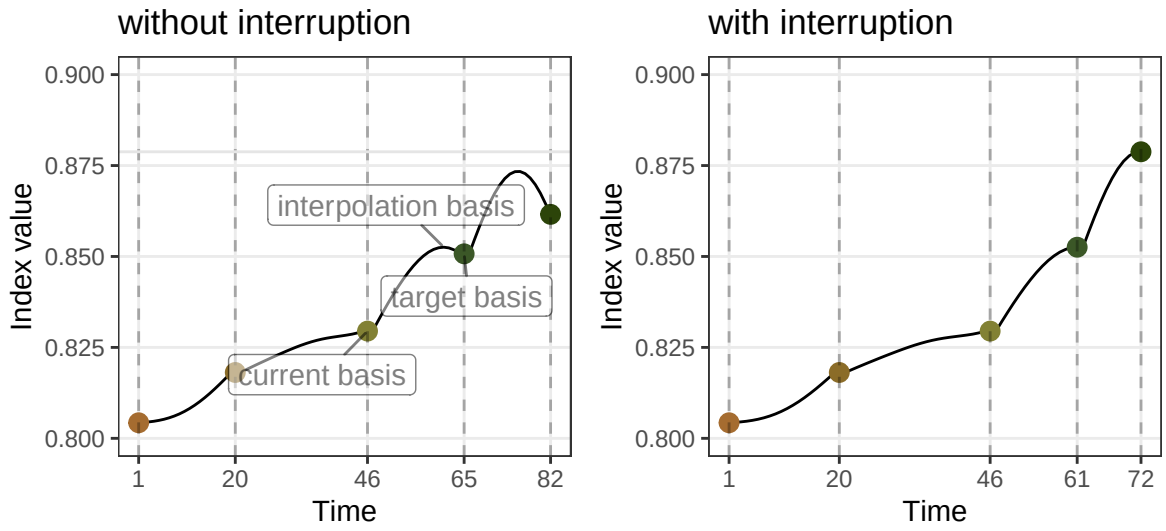

Figure 2.8: *Comparison of the interpolation before and after implementing the interruption for the 2D projection problem on* `boa6` *data using holes index, optimised by CRS. On the left panel, the basis with a higher index value is found during the interpolation but not used. On the right panel, the interruption stops the interpolation at the basis with the highest index value for each iteration and results in a final basis with a higher index value, as shown on the right panel.*

been added to ensure the new basis generated is distinguishable from the current one in terms of the distance in the space, the relative change in the index value, and neighbourhood size:

1) the distance between the basis found and the current needs to be larger than 1e-3;
2) the relative change of the index value needs to be larger than 1e-5; and
3) the alpha parameter needs to be larger than 0.01.

Figure 2.9 presents the projected data and trace plot of a 2D projection, optimised by CRS and followed by the polish step. The top row shows the initial projection, the final projection after CRS, and the final projection after polish, respectively. The end basis found by CRS reveals the four clusters in the data, but the edges of each cluster are not clean-cut. Polish works with this end basis and further pushes the index value to produce clearer edges of the cluster, especially along the vertical axis.

2.5.4 Seeing the signal in the noise

The holes index function used for all the examples before this section produces a smooth interpolation, while this is not the case for all the indexes. An example of a noisy index function for 1D projections compares the projected data, $\mathbf{Y}_{n\times 1}$, to a randomly generated normal

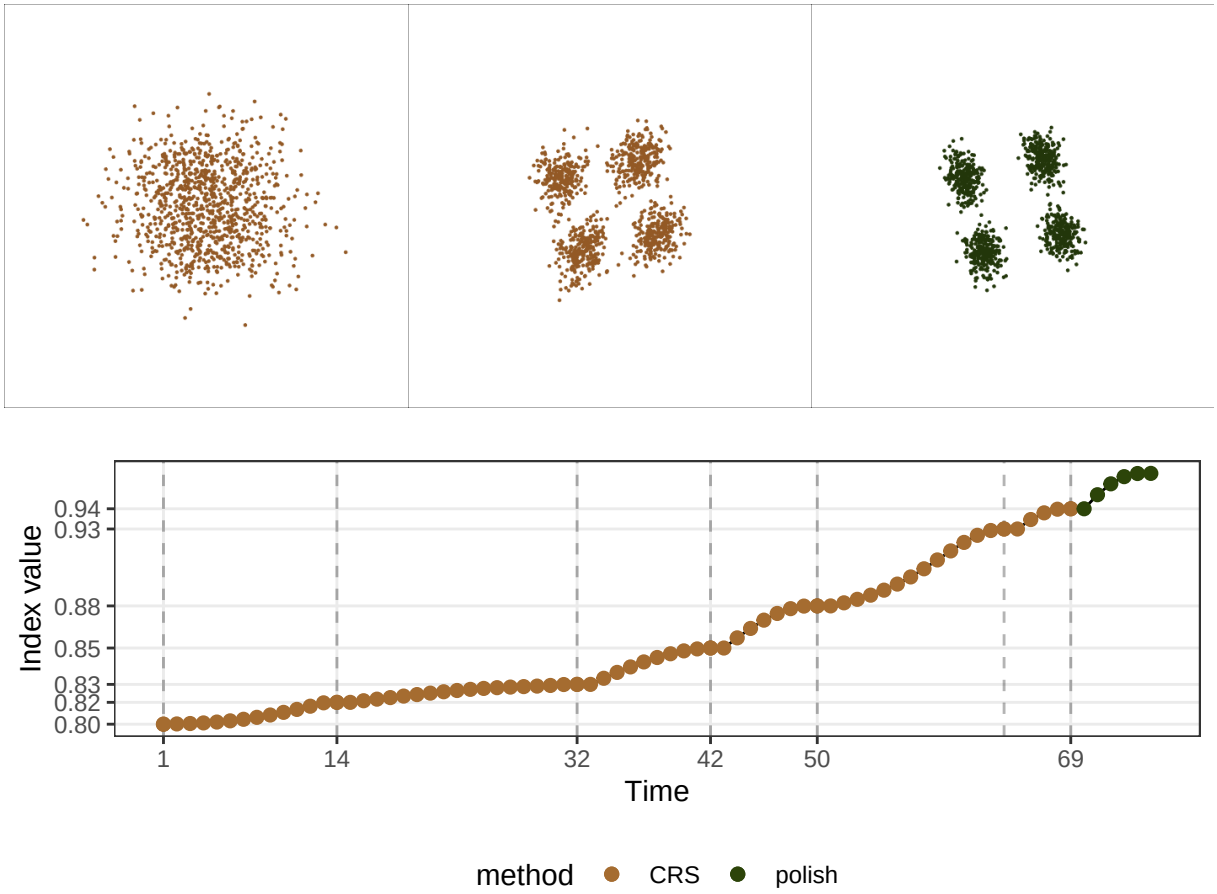

Figure 2.9: *Comparison of the projected data before and after using polishing for a 2D projection problem on* `boa6` *data using holes index. The top row shows the initial projected data and the final views after CRS and polish search, and the second row traces the index value. The clustering structure in the data is detected by CRS (top middle panel), but the polish step improves the index value and produces clearer boundaries of the clusters (top right panel), especially along the vertical axis. Note that the parameter* `max.tries` *is set to 400 in this experiment for CRS to do its best.*

distribution, $\mathcal{N}_{n\times 1}$, using the Kolmogorov test. Let $F_{\cdot}(n)$ be the empirical cumulative distribution function (ECDF) with two possible subscripts, Y and $\mathcal{N}$, representing the projected and randomly generated data, and n denoting the number of observations, the Kolmogorov index $I^K(n)$, is defined as:

$$I^K(n) = \max\left[F_Y(n) - F_{\mathcal{N}}(n)\right].$$

With a non-smooth index function, two research questions are raised:

1) whether any optimiser fails to optimise this non-smooth index; and

2) whether the optimisers can find the global optimum despite the presence of a local optimum.

Figure 2.10 presents the trace and PCA plots of all three optimisers, and as expected, none of the interpolated paths are smooth. There is barely any improvement made by PD, indicating its failure in optimising non-smooth index functions. While CRS and SA have managed to get close to the index value of the theoretical best, the trace plot shows that it takes SA much longer to find the final basis. This long interpolation path is partially due to the fluctuation in the early iterations, where SA tends to generously accept inferior bases before concentrating near the optimum. The PCA plot shows the interpolation path and search points, excluding the last termination iteration. Pseudo-Derivative (PD) quickly gets stuck near the starting position. Comparing the amount of random search done by CRS and SA, it is surprising that SA does not carry as many samples as CRS. Combining the insights from both the trace and PCA plot, one can learn the two different search strategies by CRS and SA: CRS frequently samples in the space and only make a move when an improvement is guaranteed to be made, while SA first broadly accepts bases in the space and then starts the extensive sampling in a narrowed subspace. The specific decision of which optimiser to use will depend on the index curve in the basis space, but if the basis space is non-smooth, accepting inferior bases at first, as SA has done here, can lead to a more efficient search in terms of the overall number of points evaluated.

The next experiment compares the performance of CRS and SA when a local maximum exists. Two search neighbourhood sizes, 0.5 and 0.7, are compared to understand how a large search neighbourhood would affect the final basis and the length of the search. Figure 2.11 shows 80 paths simulated using 20 seeds in the PCA plot, faceted by the optimiser and search size. With CRS and a search size of 0.5, despite being the simplest and fastest, the optimiser fails in three instances where the final basis lands neither near the local nor the global optimum. With a larger search size of 0.7, more seeds have found the global maximum. Comparing CRS and SA for a search size of 0.5, SA does not seem to improve the final basis found, despite having longer interpolation paths. However, the denser paths near the local maximum are an indicator that SA is working hard to examine if there is any other optimum in the basis space, but the relatively small search size has diminished its ability to reach the global maximum. With a larger search size, almost all the seeds (16 out of 20) have found the global maximum, and some final bases are much closer to the theoretical best, as compared to the three other cases. This indicates that

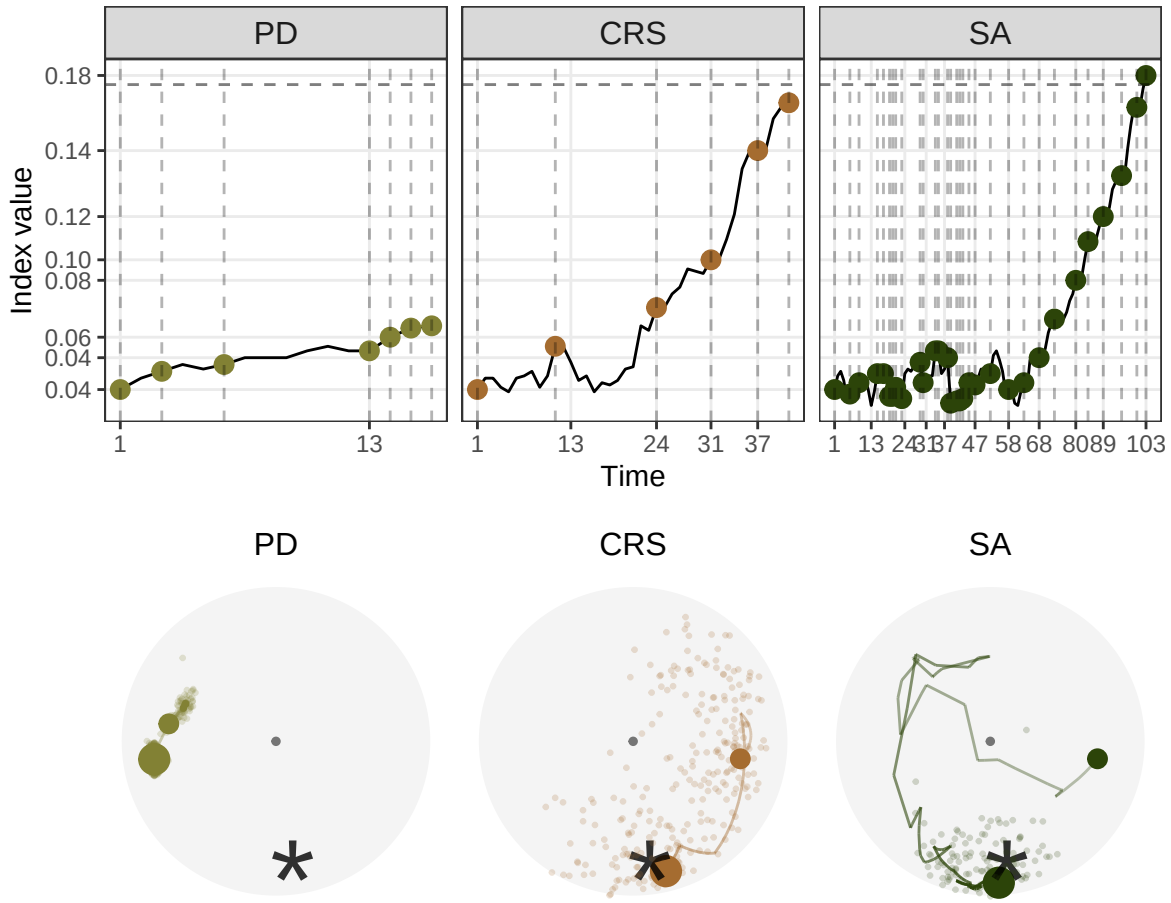

Figure 2.10: *Comparison of the three optimisers in optimising $I^{nk}(n)$ index for a 1D projection problem on a five-variable dataset,* `boa5`*. Both CRS and SA succeed in the optimisation, PD fails to optimise this non-smooth index. Further, SA takes much longer than CRS to finish the optimisation, but finishes off closer to the theoretical best.*

SA, with a reasonable large search window, is able to overcome the local optimum and optimise close towards the global optimum.

2.5.5 Reconciling the orientation

One interesting situation observed in the previous examples is that, for some simulations, as shown on the left panel of Figure 2.12, the target basis is generated on the other half of the basis space, and the interpolator seems to draw a straight line to interpolate. Bases with opposite signs do not affect the projection and index value, but we would prefer the target to have the same orientation as the current basis. The orientation of two bases can be computationally checked by calculating the determinant – a negative value suggests the two bases have a different orientation. For 1D bases, this can be corrected by flipping the sign on one basis. For higher dimensions, it can be a bit more difficult because the orthonormality of the basis needs to be

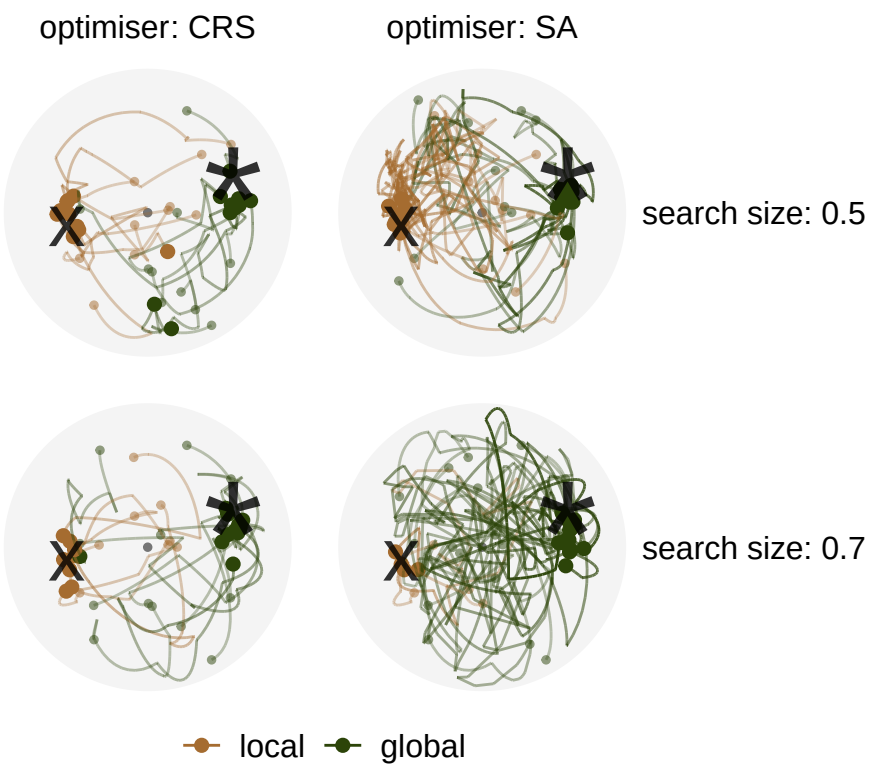

Figure 2.11: *Comparing 20 search paths in the PCA-projected basis space faceted by two optimisers: CRS and SA, and two search sizes: 0.5 and 0.7. The optimisation is on the 1D projection index, $I^{nk}(n)$, for* `boa6` *data, where a local optimum, annotated by the cross (x), is presented in this experiment, along with the global optimum (*).*

also maintained when an individual vector is flipped. Here, an orientation check is carried out once a new target basis is generated, and the sign in the target basis will be flipped if a negative determinant is obtained. The interpolation after implementing the orientation check is shown on the right panel of Figure 2.12, where the unsatisfactory interpolation no longer exists.

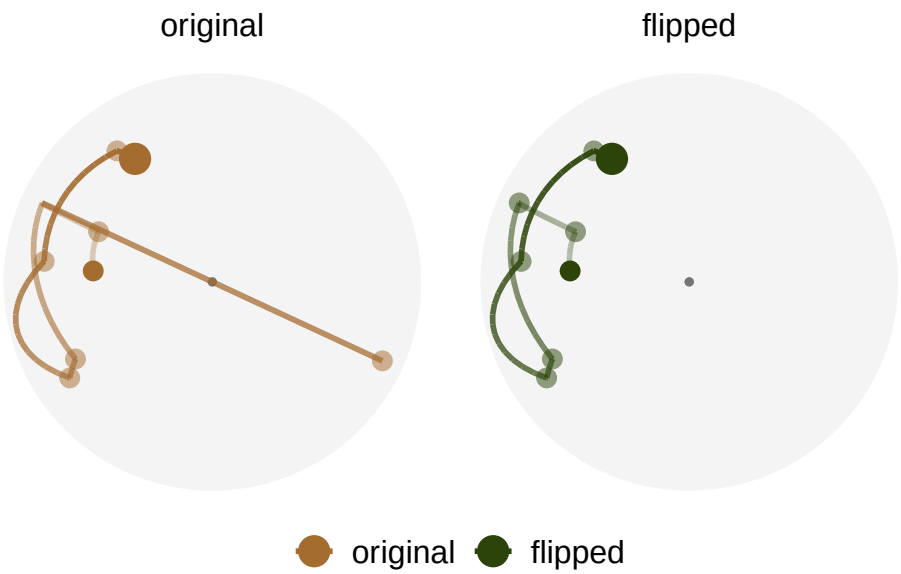

Figure 2.12: *Comparison of the interpolation in the PCA-projected basis space before and after reconciling the orientation of the target basis. Optimisation is on the 1D projection index, $I^{nk}(n)$, for boa6 data using CRS with seed 2463. The dots represent the target basis in each iteration, and the path shows the interpolation. On the left panel, one target basis is generated with an opposite orientation to the current basis (hence appear on the other side of the basis space), and the interpolator crosses the origin to perform the interpolation. The right panel shows the same interpolation after implementing an orientation check, and the undesirable interpolation disappears.*

2.6 Implementation

This project contributes to the software development in two packages: a data collection object is implemented in `tourr` (Wickham et al. 2011), while the visual diagnostics of the optimisers is implemented in `ferrn` (Zhang et al. 2021). The functions in the `ferrn` (Zhang et al. 2021) package are listed below:

- Main plotting functions:
  - `explore_trace_search()` produces summary plots in Figure 2.2.
  - `explore_trace_interp()` produces trace plots for the interpolation points in Figure 2.3.
  - `explore_space_pca()` produces the PCA plot of projection bases on the reduced space. Figure 2.4 includes the additional details of anchor and search bases, which can be turned on by the argument `details = TRUE`. The animated version is produced with argument `animate = TRUE`.

  - `explore_space_tour()` produces animated tour view on the full space of the projection bases in Figure 2.6.

- `get_*()` extracts and manipulates certain components from the existing data object.

  - `get_anchor()` extracts target observations.
  - `get_basis_matrix()` flattens all the bases into a matrix.
  - `get_best()` extracts the observation with the highest index value in the data object.
  - `get_dir_search()` extracts directional search observations for PD search.
  - `get_interp()` extracts interpolated observations.
  - `get_interp_last()` extracts the ending interpolated observations in each iteration.
  - `get_interrupt()` extracts the ending interpolated observations and the target observations if the interpolation is .interrupted
  - `get_search()` extracts search observations.
  - `get_search_count()` extracts the count of search observations.
  - `get_space_param()` produces the coordinates of the centre and radius of the basis space.
  - `get_start()` extracts the starting observation.
  - `get_theo()` extracts the theoretical best observations, if given.

- `bind_*()` incorporates additional information outside the tour optimisation into the data object.

  - `bind_theoretical()` binds the theoretical best observation in simulated experiment.
  - `bind_random()` binds randomly generated bases in the projection bases space to the data object.
  - `bind_random_matrix()` binds randomly generated bases and outputs in a matrix format.

- `add_*()` provides wrapper functions to create ggprotos for different components for the PCA plot

  - `add_anchor()` for plotting anchor bases.
  - `add_anno()` for annotating the symmetry of start bases.
  - `add_dir_search()` for plotting the directional search bases with magnified distance.

  - `add_end()` for plotting end bases.
  - `add_interp()` for plotting the interpolation path.
  - `add_interp_last()` for plotting the last interpolation bases for comparing with target bases when interruption is used.
  - `add_interrupt()` for linking the last interpolation bases with target ones when interruption is used.
  - `add_search()` for plotting search bases.
  - `add_space()` for plotting the circular space.
  - `add_start()` for plotting start bases.
  - `add_theo()` for plotting theoretical best bases, if applicable.

- Utilities

  - `theme_fern()` and `format_label()` for better display of the grid lines and axis formatting.
  - `clean_method()` to clean up the name of the optimisers.
  - `botanical_palettes()` is a collection of colour palettes from Australian native plants. Quantitative palettes include daisy, banksia, and cherry, and sequential palettes contain fern and acacia.
  - `botanical_pal()` as the colour interpolator.
  - `scale_color_*()` and `scale_fill_*()` for scaling the colour and fill of the plot.

2.7 Conclusion

This paper has provided several visual diagnostics that can be used for understanding a complex optimisation procedure and are implemented in the `ferrn` package. The methods were illustrated using the optimisers available for projection pursuit guided tour. Here the constraint is the orthonormality condition of the projection bases, which corresponds to optimisation over spheres and torii. The approach described broadly applies to other constrained optimisers. Although the manifold in p-space might be different the diagnostic techniques are the same. A researcher would begin by saving the path of the optimiser in a form required to input into the `ferrn` package, as described in this paper. One might generally make more samples from the constrained space upon which to assess and compare the optimisation paths. These high-dimensional data objects can then be viewed using the tour.

The progressive optimisation of a target function and its coverage of the search space can be viewed in both reduced 2D space and the full space. These visualisations can lead to insights for evaluating and comparing the performance of multiple optimisers operating on the same task. They can provide a better understanding of existing methods or motivate the development of new approaches. For example, we have compared how three optimisers perform when maximising a non-smooth index function and have illustrated how the pseudo-derivative search fails in this setting. The observations from our experiments have also been translated into improved optimisation methods for the guided tour, e.g., we introduced the option to interrupt the search if a better basis is found along the path.

This work might be considered an effort to bring transparency into algorithms. Although little attention is paid by algorithm developers to providing ways to output information during intermediate steps, this is an important component for enabling others to understand and diagnose the performance. Algorithms are an essential component of artificial intelligence that is used to make daily life easier. Interpretability of algorithms is important to guard against aspects like bias and inappropriate use. The data object underlying the visual diagnostics here is an example of what might be useful in algorithm development generally.

2.8 Acknowledgements

This article is created using knitr (Xie 2015) and rmarkdown (Xie, Allaire, and Grolemund 2018) in R. The source code for reproducing this paper can be found at: https://github.com/huizezhang-sherry/paper-ferrn.

Chapter 3

Cubble: An R Package for Organizing and Wrangling Multivariate Spatio-temporal Data

Multivariate spatio-temporal data refers to multiple measurements taken across space and time. For many analyses, spatial and time components can be separately studied: for example, to explore the temporal trend of one variable for a single spatial location, or to model the spatial distribution of one variable at a given time. However for some studies, it is important to analyze different aspects of the spatio-temporal data simultaneously, for instance, temporal trends of multiple variables across locations. In order to facilitate the study of different portions or combinations of spatio-temporal data, we introduce a new class, cubble, with a suite of functions enabling easy slicing and dicing on different spatio-temporal components. The proposed cubble class ensures that all the components of the data are easy to access and manipulate while providing flexibility for data analysis. In addition, the cubble package facilitates visual and numerical explorations of the data while easing data wrangling and modelling. The cubble class and the tools implemented in the package are illustrated with examples from climate data analysis.

3.1 Introduction

Spatio-temporal data (Bivand et al. 2008; Lovelace, Nowosad, and Muenchow 2019; Pebesma and Bivand 2019) has a spatial component referring to the location of each observation and a temporal component that is recorded at regular or irregular time intervals. It may also include multiple variables measured at each spatial and temporal values. With spatio-temporal data, one can fix the time to explore the spatial features of the data, fix the spatial location/s to explore temporal aspects, or dynamically explore the space and time simultaneously. In order to computationally explore the spatial, temporal and spatio-temporal and multiple variable aspects of such data, it needs to be stored in a data object that allows the user to query, group and dissect all the different data faces.

The Comprehensive R Archive Network (CRAN) task view SpatioTemporal (Pebesma and Bivand 2022) gathers information about R packages designed for spatio-temporal data and it has a section on *Representing data* that lists existing spatio-temporal data representations used in R. Among them, the `spacetime` package (Pebesma 2012) implements four S4 classes to handle spatio-temporal data with different spatio-temporal layouts (full grid, sparse grid, irregular, and trajectory). The `stars` package (Pebesma 2021) implements an S3 class built from dense arrays.

However, the data representation implemented in those packages might present certain challenges when applying the principles of tidy data (Wickham 2014) for data analysis. The concept of tidy data is based on three principles regarding how data should be organized in tables to facilitate easier analysis: 1) one observation a row, 2) one variable a column, and 3) one type of observation a table. The third principle of tidy data is particularly relevant for spatio-temporal data since these data are naturally observed at different units: the spatial locations and the temporal units. While the tidyverse suite of R packages implements data wrangling and visualization tools primarily focused on working with single tables, there are not many tools available for handling relational data specifically for spatio-temporal data. This motivates a new design to organise spatio-temporal data in a way that would make data wrangling, visualizing and analyzing easier.

This paper presents the R package, `cubble`, which implements a new cubble class to organize spatial and temporal variables as two forms of a single data object so that they can be wrangled separately or combined, while being kept synchronized. Among the four spacetime layouts in

Pebesma (2012), the `cubble` class can handle the full grid layout and the sparse grid layout. The software is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=cubble.

The rest of the paper is organized as follows: Section 3.2 presents the main design and functionality of the `cubble` package. Section 3.3 explains how the `cubble` package deals with more advanced considerations, including data matching and how the package fits with existing static and interactive visualization tools. Moreover we also illustrate how the `cubble` package deals with spatio-temporal data transformations. Section 3.4 uses primarily Australian weather station data as examples to demonstrate the use of the package. An example of how the `cubble` package handles Network Common Data Form (NetCDF) data is also provided. Section 3.5 discuss the paper contributions and future directions.

3.2 The cubble package

The cubble class includes two subclasses: the spatial cubble and the temporal cubble, which can be pivoted back and forth to focus on the two aspects of the spatio-temporal data, as illustrated in Figure 3.1. This section provides an overview of the `cubble` package, including the cubble class and its attributes, class creation and coercion, a summary of implemented functionality, the compatibility with other spatial and temporal packages (`sf` and `tsibble`), and a comparison with other spatio-temporal packages (`stars` and `sftime`).

3.2.1 The cubble class

The cubble class is an S3 class built on tibble that allows the spatio-temporal data to be wrangled in two forms (subclasses):

- a spatial cubble with class `c("spatial_cubble_df", "cubble_df")`
- a temporal cubble with class `c("temporal_cubble_df", "cubble_df")`

In a spatial cubble object, spatial variables are organised as columns and temporal variables are nested within a specialised `ts` column. For example, the spatial cubble object, `cb_spatial` printed below, contains weather records of three airport stations from the Global Historical Climatology Network Daily (GHCND) database (Menne et al. 2012). In this case, the spatial cubble is convenient for wrangling the spatial variables:

```
cb_spatial
```

```
# cubble:   key: id [3], index: date, nested form
# spatial:  [144.8321, -37.98, 145.0964, -37.6655], Missing CRS!
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           long   lat  elev name              wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>              <dbl> <list>
1 ASN00086038  145. -37.7  78.4 essendon airport   95866 <tibble>
2 ASN00086077  145. -38.0  12.1 moorabbin airport  94870 <tibble>
3 ASN00086282  145. -37.7 113.  melbourne airport  94866 <tibble>
```

In a temporal cubble, temporal variables are expanded in the long form and spatial variables are stored as a data attribute. The temporal cubble object, `cb_temporal`, contains the same spatio-temporal data as the spatial cubble object, `cb_spatial`, but in a structure that is easier for temporal analysis:

```
cb_temporal
```

```
# cubble:   key: id [3], index: date, long form
# temporal: 2020-01-01 -- 2020-01-10 [1D], no gaps
# spatial:  long [dbl], lat [dbl], elev [dbl], name [chr], wmo_id
#   [dbl]
  id          date        prcp  tmax  tmin
  <chr>       <date>     <dbl> <dbl> <dbl>
1 ASN00086038 2020-01-01     0  26.8  11
2 ASN00086038 2020-01-02     0  26.3  12.2
3 ASN00086038 2020-01-03     0  34.5  12.7
4 ASN00086038 2020-01-04     0  29.3  18.8
5 ASN00086038 2020-01-05    18  16.1  12.5
# i 25 more rows
```

The cubble attributes

Both cubble objects inherit tibble's attributes (which originates from data frames): `class`, `row.names`, and `names`. Additionally, both have three specialised attributes: `key`, `index`, and `coords`, where `key` and `index` are used as introduced in the `tsibble` package (Wang, Cook, and Hyndman 2020). In cubble, the `key` attribute identifies the row in the spatial cubble (given the internal use of `tidyr::nest()` for nesting), and when combined with the `index` argument, it identifies the row in the temporal cubble. Currently, cubble only supports one variable as the key. The accepted temporal classes for `index` includes the base R classes `Date`, `POSIXlt`, `POSIXct`, as well as tsibble's `yearmonth`, `yearweek`, and `yearquarter` classes. The `coords` attribute represents an ordered pair of coordinates that can be either an unprojected pair of longitude and latitude, or a projected easting and northing value. Moreover, temporal cubbles have a special attribute called `spatial` to store the spatial variables. Shortcut functions are available to extract attributes from the temporal cubble object, for example, `spatial()` for extracting spatial variables:

```
spatial(cb_temporal)
```

```
# A tibble: 3 x 6
  id           long   lat  elev name              wmo_id
  <chr>       <dbl> <dbl> <dbl> <chr>              <dbl>
1 ASN00086038  145. -37.7  78.4 essendon airport   95866
2 ASN00086077  145. -38.0  12.1 moorabbin airport  94870
3 ASN00086282  145. -37.7 113.  melbourne airport  94866
```

3.2.2 Creation and coercion

The spatial and temporal aspect of spatio-temporal data are often stored separately in the database. For climate data, analysts may initially receive station metadata and then query the time series based on the metadata. A (spatial) cubble object can be constructed from separate spatial and temporal tables using the function `make_cubble()`. The three attributes `key`, `index`, and `coords` need to be specified. The following code creates a spatial cubble from its spatial component, `stations` and temporal component `meteo`:

```
make_cubble(spatial = stations, temporal = meteo,
            key = id, index = date, coords = c(long, lat))
```

```
# cubble:   key: id [3], index: date, nested form
# spatial:  [144.8321, -37.98, 145.0964, -37.6655], Missing CRS!
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           long   lat  elev name              wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>              <dbl> <list>
1 ASN00086038  145. -37.7  78.4 essendon airport   95866 <tibble>
2 ASN00086077  145. -38.0  12.1 moorabbin airport  94870 <tibble>
3 ASN00086282  145. -37.7 113.  melbourne airport  94866 <tibble>
```

Other R spatio-temporal objects can be coerced into a `cubble` object with the function `as_cubble()`. This includes a joined `tibble` or `data.frame` object, a NetCDF object, a `stars` object (Pebesma 2021), and a `sftime` object (Teickner, Pebesma, and Graeler 2022). In the example below, the spatial cubble object is created from `climate_flat, which combines the previousstations` and `meteo` into a single tibble object:

```
climate_flat |> as_cubble(key = id, index = date, coords = c(long, lat))
```

```
# cubble:   key: id [3], index: date, nested form
# spatial:  [144.8321, -37.98, 145.0964, -37.6655], Missing CRS!
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           long   lat  elev name              wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>              <dbl> <list>
1 ASN00086038  145. -37.7  78.4 essendon airport   95866 <tibble>
2 ASN00086077  145. -38.0  12.1 moorabbin airport  94870 <tibble>
3 ASN00086282  145. -37.7 113.  melbourne airport  94866 <tibble>
```

3.2.3 Functions and methods

The `cubble` package has several functions implemented for data wrangling and to facilitate data analysis as summarized in Table 3.1. In addition, for each of the three cubble classes there are

Category	Functions
base R	`[`, `[[<-`, `names<-`
tidyverse	`dplyr_row_slice`, `dplyr_col_modify`, `dplyr_reconstruct`, `select`, `mutate`, `arrange`, `filter`, `group_by`, `ungroup`, `summarise`, `select`, `slice`, `rowwise`, `rename`, `bind_rows`, `bind_cols`, `relocate`, `type_sum`, the slice family (`slice_head`, `slice_tail`, `slice_max`, `slice_min`, `slice_sample`) and the join family (`left_join`, `right_join`, `inner_join`, `full_join`, `anti_join`, `semi_join`)
cubble	`as_cubble`, `cubble`, `make_cubble`, `check_key`, `face_temporal`, `face_spatial`, `unfold`, `key`, `key_vars`, `key_data`, `index`, `index_var`, `coords`, `spatial`, `match_sites`, `match_spatial`, `match_temporal`, `geom_glyph`, `geom_glyph_box`, `geom_glyph_line`, `make_spatial_sf`, `make_temporal_tsibble`, `fill_gaps`, and `scan_gaps`

Table 3.1: *An overview of functions implemented in the cubble package, categorised into base R, tidyverse, and cubble functions.*

Class	Method
`cubble_df`	`[[<-`, `dplyr_col_modify`, `key_data`, `key_vars`, `key`, `print`
`spatial_cubble_df`	`[`, `names<-`, `tbl_sum`, `dplyr_reconstruct`, `dplyr_row_slice`, `face_spatial`, `face_temporal`, `unfold`, `arrange`, `rename`, `rowwise`, `group_by`, `ungroup`, `select`, `spatial`, `summarise`, `unfold`, `update_cubble`
`temporal_cubble_df`	`[`, `names<-`, `tbl_sum`, `arrange`, `dplyr_reconstruct`, `dplyr_row_slice`, `face_spatial`, `face_temporal`, `unfold`, `fill_gaps`, `group_by`, `ungroup`, `rename`, `rowwise`, `scan_gaps`, `select`, `spatial`, `summarise`, `tbl_sum`, `bind_rows`, `bind_cols`, `update_cubble`

Table 3.2: *An overview of the methods implemented in the three `cubble` classes. Methods are implemented in the `cubble_df` class when they behave consistently across the spatial and temporal cubble; otherwise, they are implemented separately.*

number of methods implemented that facilitates the handling of the data as shown in Table 3.2. In particular, the `cubble_df` class handles methods that behave consistently in both spatial and temporal cubble. When the method works differently internally on the spatial and temporal cubble, it is implemented separately in `spatial_cubble_df` and `temporal_cubble_df`.

The pair of cubble verbs, `face_temporal()` and `face_spatial()`, pivots the cubble object between its two forms or faces, as illustrated in Figure 3.1. The code applies `face_temporal()` on the spatial cubble, `cb_spatial`, introduced in Section 3.2.1 to get a temporal cubble:

```
face_temporal(cb_spatial)
```

```
# cubble:   key: id [3], index: date, long form
# temporal: 2020-01-01 -- 2020-01-10 [1D], no gaps
# spatial:  long [dbl], lat [dbl], elev [dbl], name [chr], wmo_id
#   [dbl]
  id          date        prcp  tmax  tmin
  <chr>       <date>     <dbl> <dbl> <dbl>
1 ASN00086038 2020-01-01     0  26.8  11
2 ASN00086038 2020-01-02     0  26.3  12.2
3 ASN00086038 2020-01-03     0  34.5  12.7
4 ASN00086038 2020-01-04     0  29.3  18.8
5 ASN00086038 2020-01-05    18  16.1  12.5
# i 25 more rows
```

Both verbs are the exact inverse of each other and apply both functions on a cubble object will result in the object itself:

```
face_spatial(face_temporal(cb_spatial))
```

```
# cubble:   key: id [3], index: date, nested form
# spatial:  [144.8321, -37.98, 145.0964, -37.6655], Missing CRS!
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           long   lat  elev name              wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>              <dbl> <list>
1 ASN00086038  145. -37.7  78.4 essendon airport   95866 <tibble>
2 ASN00086077  145. -38.0  12.1 moorabbin airport  94870 <tibble>
3 ASN00086282  145. -37.7 113.  melbourne airport  94866 <tibble>
```

To enable operations involve both spatial and temporal variables, the function `unfold` incorporates spatial variables into the temporal cubble. Below is an example to include the coordinate columns (`long` and `lat`) into `cb_temporal` to prepare the data for a glyph map transformation, which will be discussed in Section 3.3.3.

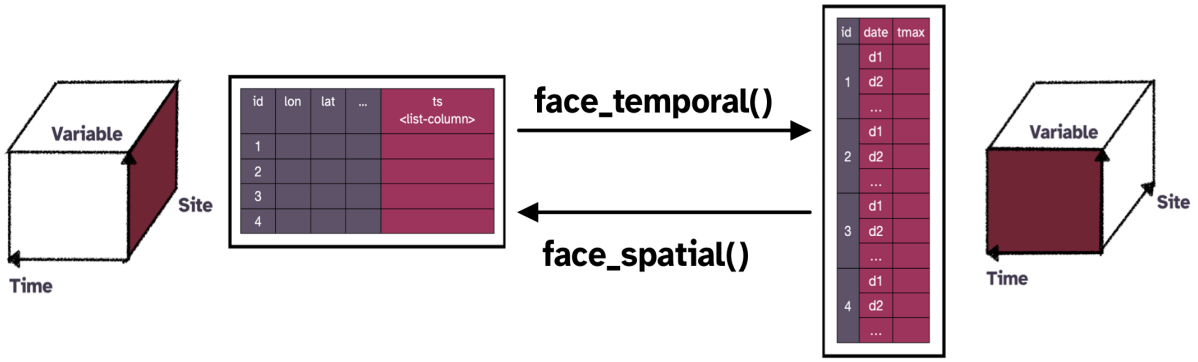

Figure 3.1: *Illustration of main functions. To focus on he temporal variables* `face_temporal()` *converts a spatial cubble into a temporal cubble. To focus on the spatial variables* `face_spatial()` *transforms a temporal cubble into a spatial cubble. This pivoting makes it easy to separately do spatial or temporal analysis.*

```
cb_temporal |> unfold(long, lat)
```

```
# cubble:   key: id [3], index: date, long form
# temporal: 2020-01-01 -- 2020-01-10 [1D], no gaps
# spatial:  long [dbl], lat [dbl], elev [dbl], name [chr], wmo_id
#   [dbl]
  id          date        prcp  tmax  tmin  long   lat
  <chr>       <date>     <dbl> <dbl> <dbl> <dbl> <dbl>
1 ASN00086038 2020-01-01     0  26.8  11    145. -37.7
2 ASN00086038 2020-01-02     0  26.3  12.2  145. -37.7
3 ASN00086038 2020-01-03     0  34.5  12.7  145. -37.7
4 ASN00086038 2020-01-04     0  29.3  18.8  145. -37.7
5 ASN00086038 2020-01-05    18  16.1  12.5  145. -37.7
# i 25 more rows
```

3.2.4 Compatibility with tsibble and sf

Analysts often have their preferred spatial or temporal data structure for spatial or temporal analysis, which they may wish to continue using for spatio-temporal analysis. With `cubble`, analysts can incorporate the `tsibble` class in a temporal cubble and the `sf` class in a spatial cubble.

Using a tsibble object as the temporal component

The `key` and `index` arguments in a `cubble` object corresponds to the tsibble counterparts and they can be safely omitted, if the temporal component is a tsibble object (`tbl_ts`). The `tsibble` class (`tbl_ts`) from the input will be carried over to the temporal cubble, indicated by the `[tsibble]` in the header and in the object class:

```
class(meteo_ts)
```

```
[1] "tbl_ts"     "tbl_df"     "tbl"        "data.frame"
```

```
ts_spatial <- make_cubble(
  spatial = stations, temporal = meteo_ts, coords = c(long, lat))
(ts_temporal <- face_temporal(ts_spatial))
```

```
# cubble:   key: id [3], index: date, long form, [tsibble]
# temporal: 2020-01-01 -- 2020-01-10 [1D], no gaps
# spatial:  long [dbl], lat [dbl], elev [dbl], name [chr], wmo_id
#   [dbl]
  id          date        prcp  tmax  tmin
  <chr>       <date>     <dbl> <dbl> <dbl>
1 ASN00086038 2020-01-01     0  26.8  11
2 ASN00086038 2020-01-02     0  26.3  12.2
3 ASN00086038 2020-01-03     0  34.5  12.7
4 ASN00086038 2020-01-04     0  29.3  18.8
5 ASN00086038 2020-01-05    18  16.1  12.5
# i 25 more rows
```

```
class(ts_temporal)
```

```
[1] "temporal_cubble_df" "cubble_df"          "tbl_ts"
[4] "tbl_df"             "tbl"                "data.frame"
```

Methods applied to tsibble objects (`tbl_ts`) can also be applied to the temporal cubble objects, for example, checking whether the data contain temporal gaps:

```
ts_temporal |> has_gaps()
```

```
# A tibble: 3 x 2
  id          .gaps
  <chr>       <lgl>
1 ASN00086038 FALSE
2 ASN00086077 FALSE
3 ASN00086282 FALSE
```

The temporal component of a created temporal cubble can include class `tbl_ts` to also be a tsibble object `tsibble` object using `make_temporal_tsibble()`. See the code example below using the `cb_temporal` object, created in Section 3.2.2:

```
cb_temporal |> make_temporal_tsibble()
```

```
# cubble:   key: id [3], index: date, long form, [tsibble]
# temporal: 2020-01-01 -- 2020-01-10 [1D], no gaps
# spatial:  long [dbl], lat [dbl], elev [dbl], name [chr], wmo_id
#   [dbl]
  id          date        prcp  tmax  tmin
  <chr>       <date>     <dbl> <dbl> <dbl>
1 ASN00086038 2020-01-01     0  26.8  11
2 ASN00086038 2020-01-02     0  26.3  12.2
3 ASN00086038 2020-01-03     0  34.5  12.7
4 ASN00086038 2020-01-04     0  29.3  18.8
5 ASN00086038 2020-01-05    18  16.1  12.5
# i 25 more rows
```

Using an sf object as the spatial component

Similarly, the spatial component of a cubble object can be an `sf` object and if the `coords` argument is omitted, it will be calculated from the `sf` geometry. The sf status is signalled by the `[sf]` label in the cubble header:

```
(sf_spatial <- make_cubble(
  spatial = stations_sf, temporal = meteo,
  key = id, index = date))
```

```
# cubble:   key: id [3], index: date, nested form, [sf]
# spatial:  [144.8321, -37.98, 145.0964, -37.6655], WGS 84
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           elev name  wmo_id  long   lat            geometry
  <chr>       <dbl> <chr>  <dbl> <dbl> <dbl>         <POINT [°]>
1 ASN00086038  78.4 esse~  95866  145. -37.7 (144.9066 -37.7276)
2 ASN00086077  12.1 moor~  94870  145. -38.0   (145.0964 -37.98)
3 ASN00086282 113.  melb~  94866  145. -37.7 (144.8321 -37.6655)
# i 1 more variable: ts <list>
```

```
class(sf_spatial)
```

```
[1] "spatial_cubble_df" "cubble_df"         "sf"
[4] "tbl_df"            "tbl"               "data.frame"
```

This allows applying functions from the `sf` package to a cubble object, for example, to handle coordinate transformation with `st_transform()`:

```
sf_spatial |> sf::st_transform(crs = "EPSG:3857")
```

```
# cubble:   key: id [3], index: date, nested form, [sf]
# spatial:  [16122635.6225205, -4576600.8687746, 16152057.3639371,
#   -4532279.35567565], WGS 84
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           elev name  wmo_id  long   lat            geometry
  <chr>       <dbl> <chr>  <dbl> <dbl> <dbl>         <POINT [°]>
1 ASN00086038  78.4 esse~  95866  145. -37.7 (16130929 -4541016)
2 ASN00086077  12.1 moor~  94870  145. -38.0 (16152057 -4576601)
3 ASN00086282 113.  melb~  94866  145. -37.7 (16122636 -4532279)
# i 1 more variable: ts <list>
```

The spatial component of a created cubble can also be an `sf` object using `make_spatial_sf()`:

```
cb_spatial |> make_spatial_sf()
```

```
# cubble:   key: id [3], index: date, nested form, [sf]
# spatial:  [144.8321, -37.98, 145.0964, -37.6655], WGS 84
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id           long   lat  elev name              wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>              <dbl> <list>
1 ASN00086038  145. -37.7  78.4 essendon airport   95866 <tibble>
2 ASN00086077  145. -38.0  12.1 moorabbin airport  94870 <tibble>
3 ASN00086282  145. -37.7 113.  melbourne airport  94866 <tibble>
# i 1 more variable: geometry <POINT [°]>
```

Comparison to other spatio-temporal classes

In R, there are other existing spatio-temporal data structures and this section compares and contrasts `cubble` with other existing alternatives, specifically `stars` and `sftime`. The `stars` package (Pebesma 2021) uses an array structure, as opposed to a tibble, to represent multivariate spatio-temporal data. While both `stars` and `cubble` support vector and raster data, it is a matter of choice on which structure to use given the application. Analysts working on satellite imageries may prefer the array structure in `stars`, while others originally working with spatio-temporal data in 2D data frames may find `cubble` easier to adopt from their existing computing workflow.

The `sftime` package (Teickner, Pebesma, and Graeler 2022) also builds from a tibble object and its focus is on handling irregular spatio-temporal data. This means `sftime` can also handle full space-time grids and sparse space-time layouts represented in `cubble`. However, `cubble` uses nesting to avoid storing spatial variables repetitively at each timestamp. This provides memory efficiency when data is observed frequently, i.e. daily or sub-daily, or the spatial geometry is computationally expensive to store repeatedly, i.e. polygons or multipolygons. Consider the `climate_aus` data in the `cubble` package with 639 stations observed daily throughout the year 2020. In that case, the `sftime` object is approximately 14 times larger than the corresponding `cubble` object (118 MB vs. 8.5 MB).

3.3 Other features and considerations

3.3.1 Spatial and temporal matching

A useful task in spatio-temporal data analysis is to combine related temporal series within a close geographic neighborhood. For example, we may want to examine data from weather stations with water flow records from nearby river sensors to understand how precipitation relates to river levels. This might be useful for predicting potential for droughts and floods.

Matching temporal data across different locations from different data sources could be done by initially identifying the corresponding spatial locations between the two data sets. Subsequently, a set of temporal features can be calculated for the series at the selected locations that can be used to matched the selected time series across locations. In cubble, locations from two datasets can be matched using the function `match_spatial()`. The function calculates the distance matrix of the locations between the two data sets and returns groups (`spatial_n_group`) with the smallest distances. For a given group, it is possible can include more locations with the argument `spatial_n_each` (default to 1 for one-on-one matching).

For the temporal matching a similarity score between the time series of spatially matched pairs is computed using the function `match_temporal()`. The similarity score is computed by a matching function which can be customized to any desired time series feature. The function `match_temporal()` takes as argument two time series in the form of a list and returns a single numerical value. By default, cubble uses a simple peak matching algorithm (`match_peak`) to count the number of peaks in two time series that fall within a specified time window.

The temporal matching requires two identifiers: one for separating each spatially matched group: `match_id` and one for separating the two data sources: `data_id`. Matching between different variables can be specified using the `temporal_by` argument, similar to the `by` syntax from dplyr's `*_join`.

```
match_temporal(
  <obj_from_match_spatial>,
  data_id = ... , match_id = ...,
  temporal_by = c("..." = "...")
)
```

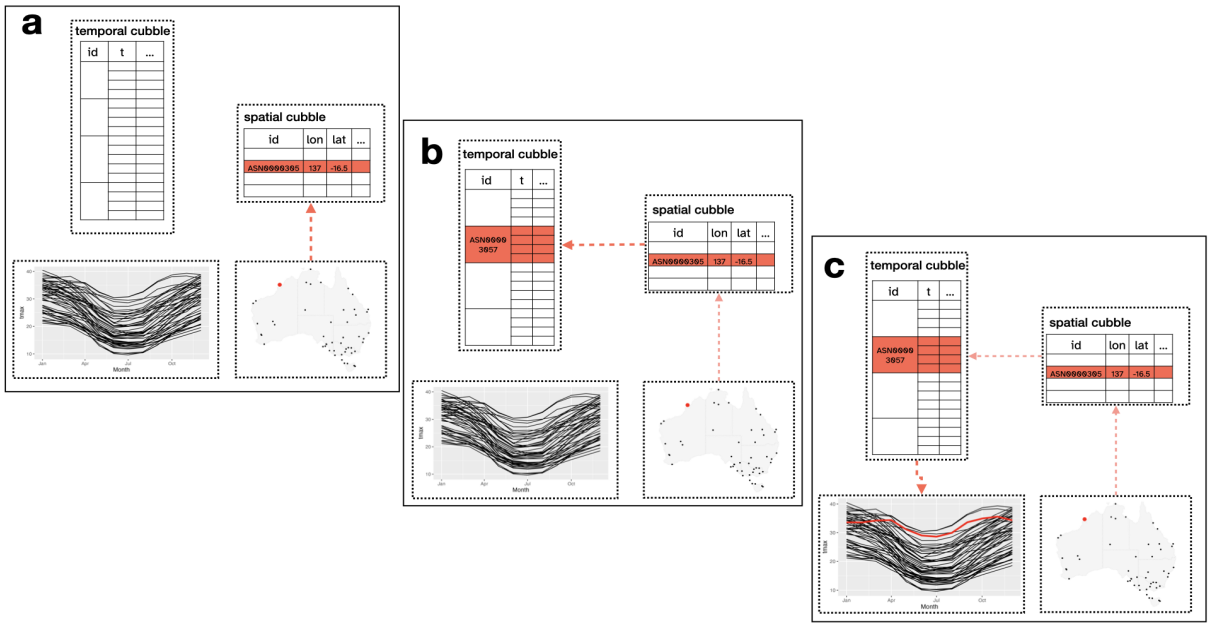

Figure 3.2: *Linking between multiple plots is made possible by shared* `crosstalk` *objects. When a station is selected on the map (a), the corresponding row in the spatial* `cubble` *will be activated. This will activate the row with the same id in the temporal* `cubble` *(b) to trigger an update of the line plot (c). The cubble package makes linking between spatial and temporal plots easy.*

3.3.2 Interactive graphics

The cubble workflow neatly allows building an interactive graphics pipeline (e.g., Buja et al. (1988); Buja, Cook, and Swayne (1996); Sutherland et al. (2000); Xie, Hofmann, and Cheng (2014); X. Cheng, Cook, and Hofmann (2016)), simplifying the data pre-processing and preparing the ingredients for linked plots. Specifically, the spatial and temporal cubble correspond to the spatial and temporal visualisation, such as a map or a time series plot, that can be linked using functionality in the `crosstalk` (J. Cheng and Sievert 2021) package.

Figure 3.2 illustrates the linking mechanism between a map and multiple time series. When a user selects a location on the map as shown on panel (a), the corresponding site is highlighted. This selection activates a row in the spatial cubble, which is then connected to the temporal cubble, resulting in the selection of all observations with the same ID as depicted in panel (b). Consequently the temporal cubble highlights the corresponding series in the time series plot displayed in panel (c). The linking can also be initiated from the time series plot by selecting points on the time series graph. This action selects rows with the same ID in the temporal cubble and the corresponding row in the spatial cubble so that points can be highlight on the map.

3.3.3 Spatio-temporal transformations

Visualizing both space and time is important for exploring and understanding the data more completely, aiding in decision-making, and facilitating effective communication. Several approaches are common: facet maps across time, map animations, or interactive graphics that link maps and time series plots among others. Faceted maps and spatio-temporal animations focus on the spatial pattern making it difficult to assess temporal trends. One alternative display is a glyph map (Wickham et al. 2012), on which each spatial location is represented by one time series line, referred to as a glyph, that traces the variable measured over time.

It is achieved through a coordinate transformation. The transformation uses linear algebra to convert the temporal coordinates (minor coordinates) into the spatial coordinates (major coordinates) and is implemented in the package `GGally` (Schloerke et al. 2021). The `cubble` package provides a new `ggproto` implementation to create glyph maps, `geom_glyph()` requiring four aesthetics : `x_major`, `y_major`, `x_minor`, and `y_minor`:

```
data |>
  ggplot() +
  geom_glyph(aes(x_major = ..., x_minor = ...,
                 y_major = ..., y_minor = ...))
```

Other useful controls to modify the glyph map that can be include are:

- the implementation of a polar coordinate glyph maps with `polar = TRUE`,
- the adjustment of the glyph size arguments using `width` and `height`,
- a transformation relative to all the series (`global_rescale` defaults to `TRUE`) or each single series, and
- the use of the reference boxes and lines with `geom_glyph_box()` and `geom_glyph_line()`.

3.4 Applications

Five examples are chosen to illustrate different aspects of the `cubble` package: creating a `cubble` object from two Coronavirus disease (COVID-19) data tables with the challenge of having different location names, using spatial transformations to make a glyph map of seasonal

temperature changes, matching river level data with weather station records to analyze water supply, reading NetCDF format data to replicate a climate reanalysis plot, and demonstrating the workflow to create interactively linked plots.

3.4.1 Victoria COVID spatio-temporal incidence and spread

Since the start of the COVID-19 pandemic, the Victoria State Government in Australia has been providing daily COVID-19 case counts per local government area (LGA). This data can be combined with map polygon data, available from the Australian Bureau of Statistics (ABS), to visualize COVID-19 incidence and spread. The COVID-19 count data (`covid`) and the LGA information (`lga`) are available in the `cubble` package as a `tsibble` object and an `sf` object, respectively. As is common, the different agencies have some difference in text id's labelling the spatial regions, so this example illustrates how this can be caught with `cubble`, and fixed. Discrepancies are flagged when creating the cubble object, notifying analysts of something that needs checking.

The `by` argument of the function `make_cubble()` is used to specify the spatial identifier in the two data sets:

```
cb <- make_cubble(lga, covid, by = c("lga_name_2018" = "lga"))
```

```
Warning: st_centroid assumes attributes are constant over geometries

! Some sites in the spatial table don't have temporal information

! Some sites in the temporal table don't have spatial information

! Use 'check_key()' to check on the unmatched key
The cubble is created only with sites having both spatial and
temporal information
```

The difference in LGA naming between both data sets triggers a warning, alerting the user to this discrepancy. The warning message suggests there are some differences between the LGA encoding used by Victoria government and ABS. The mismatches can be checked using `check_key()`, which takes the same inputs as `make_cubble()`, but returns a summary of key matches between the spatial and temporal input data:

```
(check_res <- check_key(
  spatial = lga, temporal = covid,
  by = c("lga_name_2018" = "lga")
))
```

```
$paired
# A tibble: 78 x 2
  spatial        temporal
  <chr>          <chr>
1 Alpine (S)     Alpine (S)
2 Ararat (RC)    Ararat (RC)
3 Ballarat (C)   Ballarat (C)
4 Banyule (C)    Banyule (C)
5 Bass Coast (S) Bass Coast (S)
# i 73 more rows

$potential_pairs
# A tibble: 2 x 2
  spatial             temporal
  <chr>               <chr>
1 Kingston (C) (Vic.) Kingston (C)
2 Latrobe (C) (Vic.)  Latrobe (C)

$others
$others$spatial
character(0)

$others$temporal
[1] "Interstate" "Overseas"   "Unknown"
```

```
attr(,"class")
[1] "key_tbl" "list"
```

The result of the `check_key()` function is a list containing three elements: 1) matched keys from both tables, 2) potentially paired keys, and 3) others keys that can't be matched. Here, the main mismatch arises from the two LGAs: Kingston and Latrobe (Kingston is an LGA in both Victoria and South Australia and Latrobe is an LGA in both Victoria and Tasmania). Analysts can then reconcile the spatial and temporal data based on this summary and recreate the cubble object:

```
lga2 <- lga |>
  rename(lga = lga_name_2018) |>
  mutate(lga = ifelse(lga == "Kingston (C) (Vic.)", "Kingston (C)", lga),
         lga = ifelse(lga == "Latrobe (C) (Vic.)", "Latrobe (C)", lga))

covid2 <- covid |> filter(!lga %in% check_res$others$temporal)

(cb <- make_cubble(spatial = lga2, temporal = covid2))
```

```
# cubble:   key: lga [80], index: date, nested form, [sf]
# spatial:  [140.961682, -39.1339581, 149.976291, -33.9960517], WGS
#   84
# temporal: date [date], n [dbl], avg_7day [dbl]
  lga             long   lat                          geometry ts
  <chr>          <dbl> <dbl>                   <GEOMETRY [°]> <list>
1 Alpine (S)      147. -36.9 POLYGON ((146.7258 -36.45922, 1~ <tbl_ts>
2 Ararat (RC)     143. -37.5 POLYGON ((143.1807 -37.73152, 1~ <tbl_ts>
3 Ballarat (C)    144. -37.5 POLYGON ((143.6622 -37.57241, 1~ <tbl_ts>
4 Banyule (C)     145. -37.7 POLYGON ((145.1357 -37.74091, 1~ <tbl_ts>
5 Bass Coast (S)  146. -38.5 MULTIPOLYGON (((145.5207 -38.30~ <tbl_ts>
# i 75 more rows
```

3.4.2 Australian historical maximum temperature

The Global Historical Climatology Network (GHCN) provides daily climate measures for stations worldwide. In the `cubble` package, the cubble object `historical_tmax` contains daily maximum temperature data for 75 stations in Australia, covering two periods: 1971-1975 and 2016-2020. This example uses `dplyr` verbs to wrangle a cubble object, and pivot between the spatial and temporal form for different parts of the analysis. The result is glyph maps to compare the changes in temperature between these two periods, created with `ggplot2`.

To prevent overlapping of weather stations on the map, stations are selected to ensure a minimum distance of 50km. Distance between stations can be calculated with `sf::st_distance()` after turning the spatial cubble to also be an sf object with `make_spatial_sf()`:

```
a <- historical_tmax |> make_spatial_sf() |> st_distance()
a[upper.tri(a, diag = TRUE)] <- 1e6

(tmax <- historical_tmax |>
  filter(rowSums(a < units::as_units(50, "km")) == 0))
```

```
# cubble:   key: id [54], index: date, nested form
# spatial:  [141.2652, -39.1297, 153.3633, -28.9786], Missing CRS!
# temporal: date [date], tmax [dbl]
  id           long   lat  elev name                   wmo_id ts
  <chr>       <dbl> <dbl> <dbl> <chr>                   <dbl> <list>
1 ASN00047016  141. -34.0    43 lake victoria storage   94692 <tibble>
2 ASN00047019  142. -32.4    61 menindee post office    94694 <tibble>
3 ASN00048015  147. -30.0   115 brewarrina hospital     95512 <tibble>
4 ASN00048027  146. -31.5   260 cobar mo                94711 <tibble>
5 ASN00048031  149. -29.5   145 collarenebri (albert ~  95520 <tibble>
# i 49 more rows
```

The daily maximum temperature is then averaged into monthly series for each period within the temporal cube. In the code above, the last step with `unfold()` moves the two coordinate

columns (`long, lat`) into the temporal cubble, preparing the data for the construction of a glyph map:

```
(tmax <- tmax |>
  face_temporal() |>
  group_by(
    yearmonth = tsibble::make_yearmonth(
      year = ifelse(lubridate::year(date) > 2015, 2016, 1971),
      month = lubridate::month(date))
  )|>
  summarise(tmax = mean(tmax, na.rm = TRUE)) |>
  mutate(group = as.factor(lubridate::year(yearmonth)),
         month = lubridate::month(yearmonth)) |>
  unfold(long, lat))
```

```
# cubble:   key: id [54], index: yearmonth, long form
# temporal: 1971 Jan -- 2016 Dec [1M], has gaps!
# spatial:  long [dbl], lat [dbl], elev [dbl], name [chr], wmo_id
#   [dbl]
  yearmonth id           tmax group month  long   lat
      <mth> <chr>       <dbl> <fct> <dbl> <dbl> <dbl>
1  1971 Jan ASN00047016  31.1 1971      1  141. -34.0
2  1971 Jan ASN00047019  33.1 1971      1  142. -32.4
3  1971 Jan ASN00048015  33.9 1971      1  147. -30.0
4  1971 Jan ASN00048027  32.5 1971      1  146. -31.5
5  1971 Jan ASN00048031  33.3 1971      1  149. -29.5
# i 1,276 more rows
```

The code below counts the number of observations for each location, revealing that there are several with less than 24 observations – these stations lack temperature values for some months. In this example, those stations are removed by switching to the spatial cubble to operate on the spatial component over time, and then, move back into the temporal cubble (to make the glyph map):

```
tmax <- tmax |>
  face_spatial() |>
  rowwise() |>
  filter(nrow(ts) == 24) |>
  face_temporal()
```

The following code creates the glyph map (a) in Figure 3.3 (additional codes are needed for highlighting the single station, Cobar and styling) and the glyph map (c) is produced similarly after further processing the data.

```
nsw_vic <- ozmaps::abs_ste |>
  filter(NAME %in% c("Victoria","New South Wales"))


tmax |>
  ggplot(aes(x_major = long, x_minor = month,
             y_major = lat, y_minor = tmax,
             group = interaction(id, group))) +
  geom_sf(data =  nsw_vic, ...,  inherit.aes = FALSE) +
  geom_glyph_box(width = 0.8, height = 0.3) +
  geom_glyph(aes(color = group), width = 0.8, height = 0.3) +
  ...
```

3.4.3 River levels and rainfall in Victoria

The Bureau of Meteorology collects water level data that can be matched with precipitation data from climate weather stations. The data `river` from the `cubble` package contains water course level data for 71 river gauges collected in Victoria, Australia. Victoria weather station data can be subsetted from the `climate_aus` data in the `cubble` package. This example demonstrates the use of the matching function introduced in Section 3.3.1 to find river gauges that mirror changes in precipitation regimes captured in the climate weather stations in Victoria.

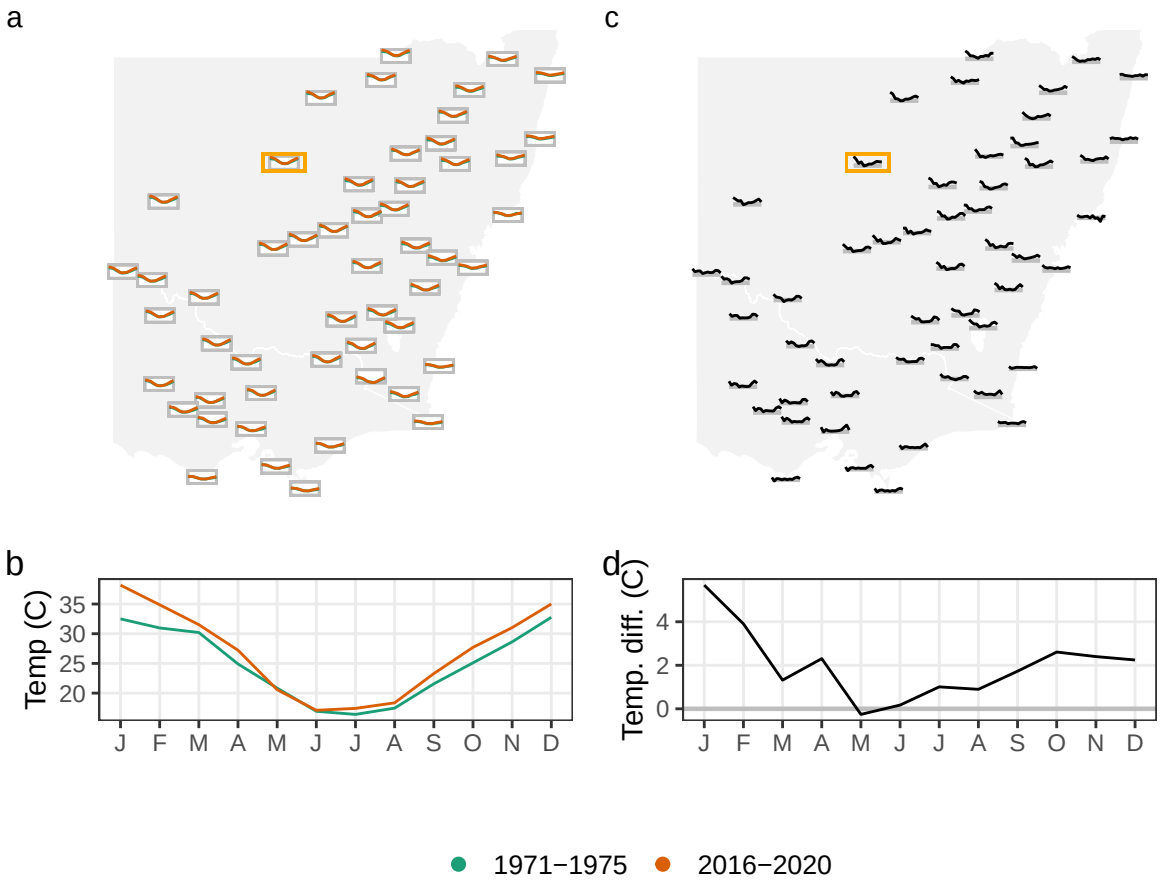

Figure 3.3: *Glyph maps comparing temperature change between 1971-1975 and 2016-2020 for 54 stations in Victoria and New South Wales, Australia. Overlaid line plots show monthly temperature (a) where a hint of late summer warming can be seen. Transforming to temperature differences (c) shows pronounced changes between the two periods. The horizontal guideline marks zero difference. One station, Cobar, is highlighted in the glyph maps and shown separately (b, d). Here the late summer (Jan-Feb) warming pattern, which is more prevalent at inland locations, is clear.*

```
climate_vic <- climate_aus |>
  filter(between(as.numeric(substr(id, 7, 8)), 76, 90)) |>
  mutate(type = "climate")
river <- cubble::river |> mutate(type = "river")
```

The first step is to perform a spatial match on the site locations between both data sets. The rainfall recorded at a particular station can directly impact the water level in nearby rivers, which emphasizes the need to find related or matching locations. With `match_spatial()`, we can obtain a summary of the 10 closest pairs of weather stations and river gauges:

```
res_sp <- match_spatial(df1 = climate_vic, df2 = river,
                        spatial_n_group = 10)
print(res_sp, n = 20)
```

```
# A tibble: 10 x 4
   from        to      dist group
   <chr>       <chr>    [m] <int>
 1 ASN00088051 406213 1838.     1
 2 ASN00084145 222201 2185.     2
 3 ASN00085072 226027 3282.     3
 4 ASN00080015 406704 4034.     4
 5 ASN00085298 226027 4207.     5
 6 ASN00082042 405234 6153.     6
 7 ASN00086038 230200 6167.     7
 8 ASN00086282 230200 6928.     8
 9 ASN00085279 224217 7431.     9
10 ASN00080091 406756 7460.    10
```

The results can also be returned as a list of matched cubbles, by setting the argument `return_cubble = TRUE`. After excluding the two pairs where a river station is matched to more than one weather stations (river station 226027 is matched twice in group 3 and 5 and similarly for station 230200 in group 7 and 8), all the results can be combined into a single cubble using `bind_rows()`,

```
res_sp <- match_spatial(
  df1 = climate_vic, df2 = river,
  spatial_n_group = 10, return_cubble = TRUE)
(res_sp <- res_sp[-c(5, 8)] |> bind_rows())
```

```
# cubble:   key: id [16], index: date, nested form, [sf]
# spatial:  [144.5203, -38.144913, 148.4667, -36.128657], WGS 84
# temporal: date [date], prcp [dbl], tmax [dbl], tmin [dbl]
  id          long   lat  elev name            wmo_id ts       type
```

```
  <chr>       <dbl> <dbl> <dbl> <chr>             <dbl> <list>   <chr>
1 ASN00088051  145. -37.0 290   redesdale         94859 <tibble> clim~
2 406213       145. -37.0  NA   CAMPASPE RIVER ~     NA <tibble> river
3 ASN00084145  148. -37.7  62.7 orbost            95918 <tibble> clim~
4 222201       148. -37.7  NA   SNOWY RIVER @ O~     NA <tibble> river
5 ASN00085072  147. -38.1   4.6 east sale airpo~  94907 <tibble> clim~
# i 11 more rows
# i 3 more variables: geometry <POINT [°]>, group <int>, dist [m]
```

To match the water level and precipitation time series across the matched locations, the function `match_temporal()` is used with the variables `group` and `type` identifying the matching group and the two data sources:

```
(res_tm <- match_temporal(data = res_sp,
                          data_id = type, match_id = group,
                          temporal_by = c("prcp" = "Water_course_level")))
```

```
# A tibble: 8 x 2
  group match_res
  <int>     <dbl>
1     1        30
2     2         5
3     3        14
4     4        20
5     6        23
# i 3 more rows
```

Similarly, the cubble output can be returned using the argument 'return_cubble = TRUE}. Here, we select the four pairs of time series (precipitation/water level) with the highest number of matching peaks and show them on the map (Figure 3.4 a). The time series of river levels is standardized to make the comparison easier in panel (b).

```
res_tm <- match_temporal(data =  res_sp,
                         data_id = type, match_id = group,
                         temporal_by = c("prcp" = "Water_course_level"),
                         return_cubble = TRUE)
(res_tm <- res_tm |> bind_rows() |> filter(group %in% c(1, 7, 6, 9)))
```

```
# cubble:   key: id [8], index: date, nested form, [sf]
# spatial:  [144.5203, -37.8817, 147.572223, -36.8472], WGS 84
# temporal: date [date], matched [dbl]
  id     long   lat  elev name  wmo_id type              geometry
  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>          <POINT [°]>
1 ASN0~  145. -37.0 290   rede~  94859 clim~  (144.5203 -37.0194)
2 4062~  145. -37.0  NA   CAMP~     NA river (144.5403 -37.01512)
3 ASN0~  146. -36.8 502   stra~  95843 clim~  (145.7308 -36.8472)
4 4052~  146. -36.9  NA   SEVE~     NA river (145.6828 -36.88701)
5 ASN0~  145. -37.7  78.4 esse~  95866 clim~  (144.9066 -37.7276)
# i 3 more rows
# i 4 more variables: group <int>, dist [m], ts <list>,
#   match_res <dbl>
```

3.4.4 ERA5: climate reanalysis data

The ERA5 reanalysis (Hersbach et al. 2020) provides hourly estimates of atmospheric, land and oceanic climate variables on a global scale and is available in the NetCDF format from Copernicus Climate Data Store (CDS). This example demonstrates a case of analysing raster spatio-temporal data using cubble, replicating Figure 19 from the Hersbach et al. (2020) paper. The plot shows the southern polar vortex splitting into two on 2002-09-26, and further splitting into four on 2002-10-04. Further explanation of why this is interesting can be found in the figure source, and also in Simmons et al. (2020) and Simmons et al. (2005).

A `ncdf4` object (Pierce 2019) can be converted into a cubble using `as_cubble()` and the NetCDF data can be subsetted with arguments `vars`, `long_range` and `lat_range`. In this example, the

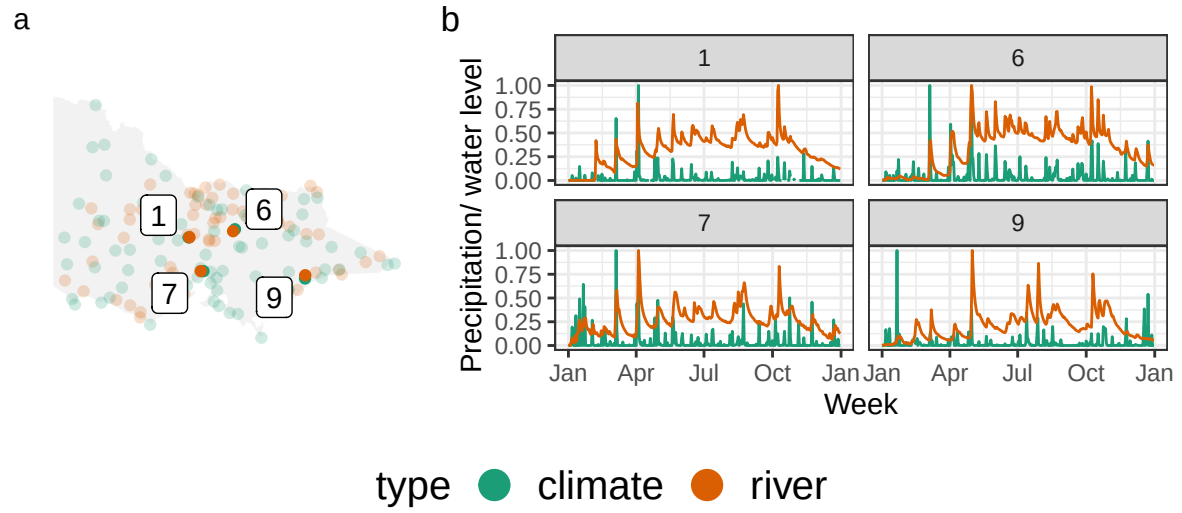

Figure 3.4: *Example of matching weather stations and river gauges. These four stations show on the map (a) and time (b) would be considered to be matching. Precipitation and water level have been standardised between 0 and 1 to be displayed in the same scale in (b). The peaks in the time series roughly match, and would reflect percipitation increasing water levels.*

variables q (specific humidity) and z (geopotential) are read in and the coordinates are subsetted to every degree in longitude and latitude:

```
raw <- ncdf4::nc_open(here::here("data/cubble/era5-pressure.nc"))
(dt <- as_cubble(
  raw, vars = c("q", "z"),
  long_range = seq(-180, 180, 1), lat_range = seq(-88, -15, 1)))
```

```
# cubble:   key: id [26640], index: time, nested form
# spatial:  [-180, -88, 179, -15], Missing CRS!
# temporal: time [date], q [dbl], z [dbl]
     id  long   lat ts
  <int> <dbl> <dbl> <list>
1     1  -180   -15 <tibble [8 x 3]>
2     2  -179   -15 <tibble [8 x 3]>
3     3  -178   -15 <tibble [8 x 3]>
4     4  -177   -15 <tibble [8 x 3]>
5     5  -176   -15 <tibble [8 x 3]>
# i 26,635 more rows
```

Once the NetCDF data is coerced into a cubble object, subsequent analysis can be conducted to filter on the date of interest, scale the variable specific humidity and create visualisation in ggplot to reproduce the ERA5 plot. A snippet of code to create Figure 3.5 is provided below with additional codes needed to style the plot.

```
res <- dt |>
  face_temporal() |>
  filter(lubridate::date(time) %in%
           as.Date(c("2002-09-22", "2002-09-26",
                     "2002-09-30", "2002-10-04"))) |>
  unfold(long, lat) |>
  mutate(q = q* 10^6)

con <- rnaturalearth::ne_coastline("small", returnclass = "sf")
box <- st_bbox(c(xmin = -180, ymin = -90, xmax = 180, ymax = -15),
               crs = st_crs(con))
country <- con |>
  st_geometry() |>
  st_crop(box) |>
  st_cast("MULTILINESTRING")

res |>
  ggplot() +
  geom_point(aes(x = long, y = lat, color = q)) +
  geom_contour(data = res, aes(x = long, y = lat, z = z), ...) +
  geom_sf(data = country, ...) +
  ...
```

3.4.5 Australian temperature range

Interactive graphics can be especially useful for spatio-temporal data because they make it possible to look at the data in multiple ways on-the-fly. This last example describes the process of using `cubble` with the `crosstalk` package to build an interactive display connecting a map

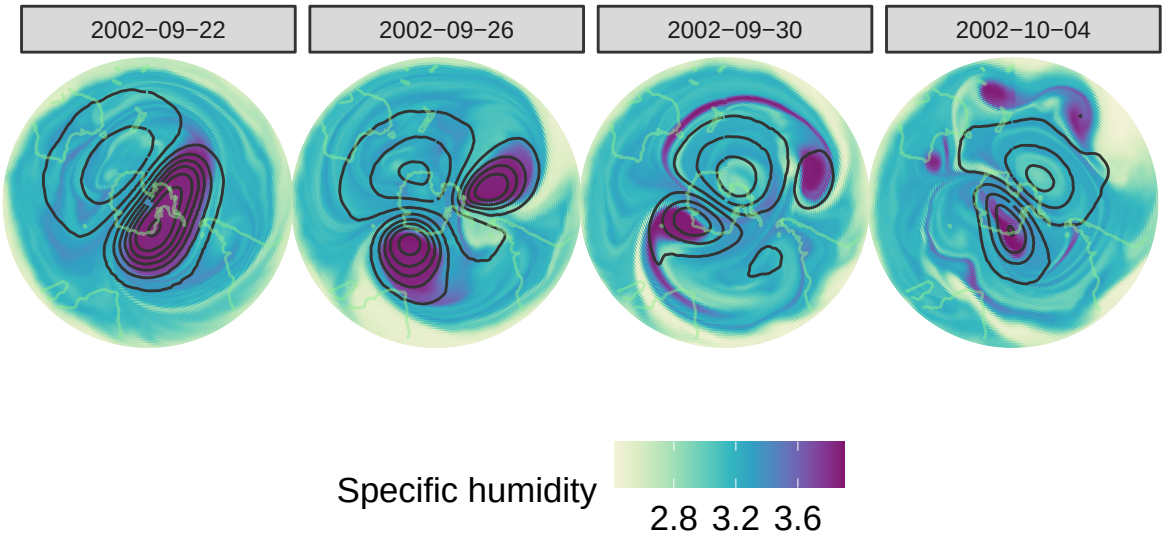

Figure 3.5: *An example illustrating that cubble can be used to readily reproduce common spatiotemporal analyses. This plot of ERA5 reanalysis (Fig. 19, Hersbach et al, 2020) shows the break-up of the southern polar vortex in late September and early October 2002. The polar vortex, signalled by the high specific humidity, splits into two on 2002-09-26 and further splits into four on 2002-10-04.*

of Australia, with ribbon plots of temperature range observed at a group of stations in 2020. The purpose is to explore the variation of monthly temperature range over the country.

Firstly, we summarise the daily data in `climate_aus` into monthly averages and calculate the variance of the monthly averages differences between the minimum and maximum temperatures. This variance will be used to color the temperature band later.

```
clean <- climate_aus |>
  face_temporal() |>
  mutate(month = lubridate::month(date)) |>
  group_by(month) |>
  summarise(
    tmax = mean(tmax, na.rm = TRUE),
    tmin = mean(tmin, na.rm = TRUE),
    diff = mean(tmax - tmin, na.rm = TRUE)
  ) |>
  face_spatial() |>
  rowwise() |>
  mutate(temp_diff_var = var(ts$diff, na.rm =TRUE))
```

The spatial and temporal cubble are then created into shared `crosstalk` objects, plotted as ggplots, and combined together using `crosstalk::bscols()`:

```
sd_spatial <- clean |> SharedData$new(~id, group = "cubble")


sd_temporal <- clean |>
  face_temporal() |>
  SharedData$new(~id, group = "cubble")


p1 <- sd_spatial |> ggplot() + ...
p2 <- sd_temporal |> ggplot() + ...
crosstalk::bscols(plotly::ggplotly(p1), plotly::ggplotly(p2), ...)
```

Figure 3.6 shows three snapshots of the interactivity. Plot (a) shows the initial state of the interactive display: all locations are shown as dots on the map, coloured by the temperature range, and the right plot shows the ribbons representing maximum to minimum for all stations. In plot (b) the station shows a high variance on the initial map, the "Mount Elizabeth" station, is selected and this produces the ribbon on the right. In plot (c) the lowest temperature in August is selected on the left map and this corresponds to the "Thredbo" station in the mountain area in Victoria and New South Wales. This station is compared to a station in the Tasmania island, the southernmost island of the country, selected on the map.

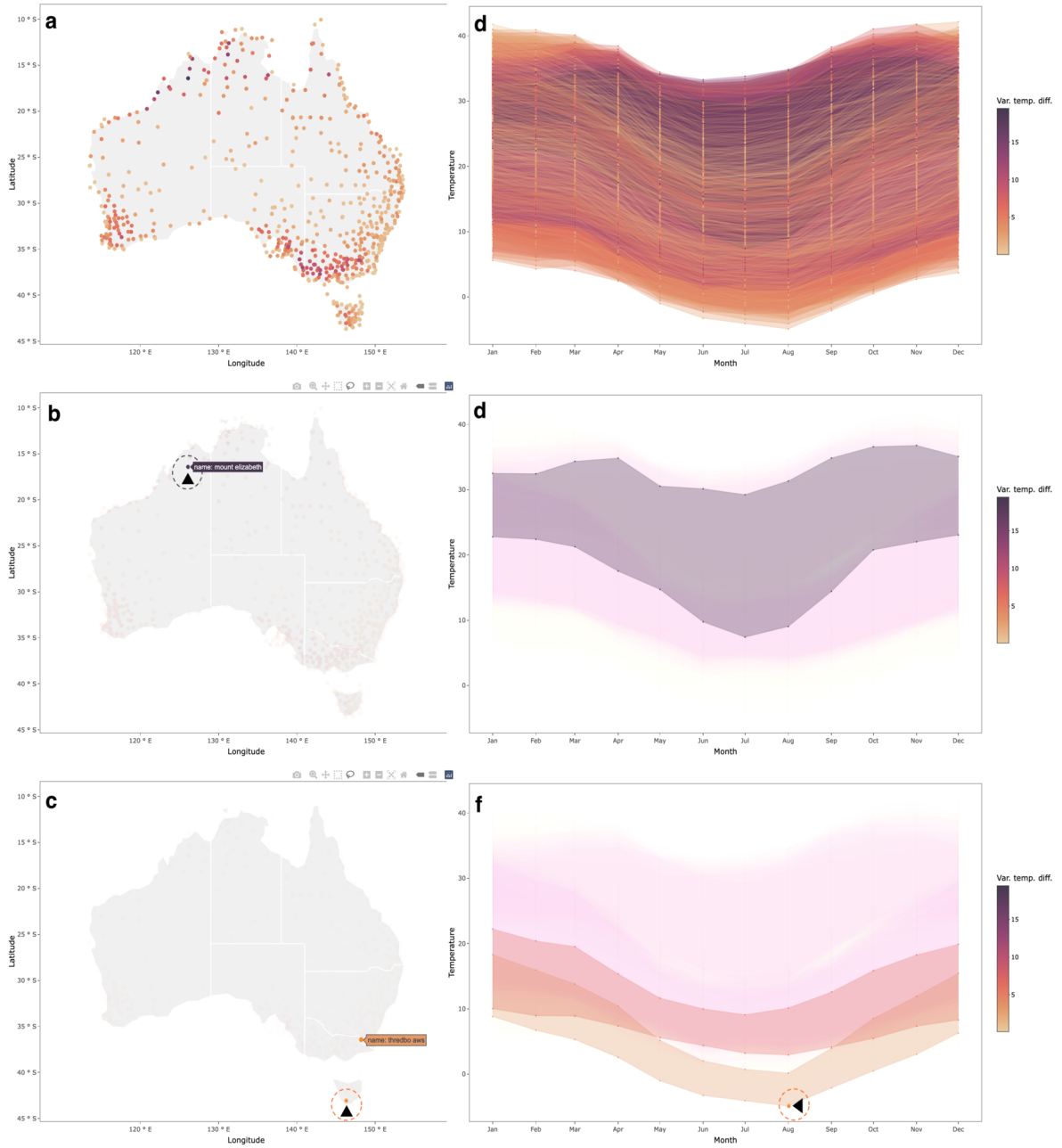

Figure 3.6: *Illustration of using cubble for interactive graphics. Here we explore temperature variation by linking a map and a seasonal display. Each row is a screen dump of the process. The top row shows all locations and all temperature profiles. Selecting a particular location on the map (here Mount Elizabeth) produces the plot in the second row. The maximum and minimum temperatures are shown using a ribbon. The bottom row first selects the lowest temperature in August in the seasonal display, which highlights the corresponding station on the map (Thredbo). Another station, located in the Tasmania Island, is then selected to compare its temperature variation with the Thredbo station.*

3.5 Conclusion

This paper presents the R package `cubble` for organizing, wrangling and visualizing spatio-temporal data. The package introduces a new data structure, `cubble`, consisting of two subclasses, spatial cubble and a temporal cubble, to organise spatio-temporal data in two different formats within the tidy data framework. The data structure and functions introduced in the package can be used and combined with existing tools for data wrangling, spatial and temporal data analysis, and visualization.

The paper includes several examples to illustrate how `cubble` is useful for spatio-temporal analysis. These examples cover different tasks of a typical data analysis workflow: handling data with spatial and temporal misalignment, matching data from multiple sources, and creating both static and interactive graphics. In addition, a re-working of an existing climate reanalysis using cubble is explained.

Possible future directions would be in two main directions: handling much larger data sets, and integrating smoothly with modeling. Spatio-temporal data can be huge, and the current recommendation is to first reduce the spatial and temporal components before constructing a cubble. Better support could be provided with additional pre-processing functions. Considerable effort has been exerted to provide tidy tools for models, with the `tidymodels` project. It provides a much better interface to the vast array of statistical and machine learning model architecture. A similar direction would be to extend cubble to provide a more unified interface to spatio-temporal modeling.

3.6 Acknowledgement

This work is funded by a Commonwealth Scientific and Industrial Research Organisation (CSIRO) Data61 Scholarship and started while Nicolas Langrené was affiliated with CSIRO's Data61. The article is created using the package `knitr` (Xie 2015) and `rmarkdown` (Xie, Allaire, and Grolemund 2018) in R with the `rticles::jss_article` template. The source code for reproducing this paper can be found at: https://github.com/huizezhang-sherry/paper-cubble.

Chapter 4

A Tidy Framework and Infrastructure to Systematically Assemble Spatio-temporal Indexes from Multivariate Data

Indexes are useful for summarizing multivariate information into single metrics for monitoring, communicating, and decision-making. While most work has focused on defining new indexes for specific purposes, more attention needs to be directed towards making it possible to understand index behavior in different data conditions, and to determine how their structure affects their values and variation in values. Here we discuss a modular data pipeline recommendation to assemble indexes. It is universally applicable to index computation and allows investigation of index behavior as part of the development procedure. One can compute indexes with different parameter choices, adjust steps in the index definition by adding, removing, and swapping them to experiment with various index designs, calculate uncertainty measures, and assess indexes' robustness. The paper presents three examples to illustrate the pipeline framework usage: comparison of two different indexes designed to monitor the spatio-temporal distribution of drought in Queensland, Australia; the effect of dimension reduction choices on the Global Gender Gap Index (GGGI) on countries' ranking; and how to calculate bootstrap confidence

intervals for the Standardized Precipitation Index (SPI). The methods are supported by a new R package, called `tidyindex`.

4.1 Introduction

Indexes are commonly used to combine and summarize different sources of information into a single number for monitoring, communicating, and decision-making. They serve as critical tools across the natural and social sciences. Examples include the Air Quality Index, El Niño-Southern Oscillation Index, Consumer Price Index, QS University Rankings, and the Human Development Index. In environmental science, climate indexes are produced by major monitoring centers, like the United States Drought Monitor and National Oceanic and Atmospheric Administration, to facilitate agricultural planning and early detection of natural disasters. In economics, indexes provide insight into market trends through combining prices of a basket of goods and services. In social sciences, indexes are used to monitor human development, gender equity, or university quality. The problem is that every index is developed in its own unique way, by different researchers or organizations, and often indexes designed for the same purpose cannot easily be compared.

To construct an index, experts typically start by defining a concept of interest that requires measurement. This concept often lacks a direct measurable attribute or can only be measured as a composite of various processes, yet it holds social and public significance. To create an index, once the underlying processes involved are identified, relevant and available variables are then defined, collected, and combined using statistical methods into an index that aims to measure the process of interest. The construction process is often not straightforward, and decisions need to be made, such as the selection of variables to be included, which might depend on data availability and the statistical definition of the index to be used, among others. For instance, the indexes constructed from a linear combination of variables require to decide the weight assigned to each variable. Some indexes have a spatial and/or temporal component, and variables can be aggregated to different spatial resolutions and temporal scales, leading to various indexes for different monitoring purposes. Hence, all these decisions can result in different index values and have different practical implications.

To be able to test different decision choices systematically for an index, the index needs to be broken down into its fundamental building blocks to analyze the contribution and effect of each

component. We call this process of breaking the index construction into different steps the index pipeline. Such decomposition of index components provides the means to standardize index construction via a pipeline and offers benefits for comparing among indexes, calculating index uncertainty, and assessing index robustness.

In this work, we provide statistical and computational methods for developing a data pipeline framework to construct and customize indexes using data. The proposed pipeline comprises various modules, including temporal and spatial aggregation, variable transformation and combination, distribution fitting, benchmark setting, and index communication. Given the decisions analysts need to make when combining multivariate data into indexes, the proposed pipeline enables the evaluation of how the specific choice can affect the index, as well as how the index may appear under alternative options. Furthermore, uncertainty calculation can also flow through the pipeline, providing the index with confidence measures.

The rest of the paper is structured as follows. Section 4.2 provides background about the development of indexes. Section 4.3 reviews the tidy framework in R and how index construction can benefit from such a framework. The details of the pipeline modules are presented in Section 4.4. Section 4.5 explains the design of the `tidyindex` package that implements the modules. Examples are given in Section 4.6 to illustrate three use cases of the pipeline.

4.2 Background to index development

There are many documents providing advice on how to construct indexes for different fields, and review articles describing the range of available indexes for specific purposes. The OECD handbook (OECD, European Union, and Joint Research Centre - European Commission 2008) provides a comprehensive guide for computing socio-economic composite indexes, with detailed steps and recommendations. The drought index handbook (Svoboda, Fuchs, et al. 2016) provides details of various drought indexes and recommendations from the World Meteorology Organization. Zargar et al. (2011), Hao and Singh (2015) and Alahacoon and Edirisinghe (2022) are review papers describing the range of possible drought indexes.

There is also some attention being given to the diagnosis of indexes, and incorporation of uncertainty. B. Jones and Andrey (2007) investigates the methodological choices made in the development of indexes for assessing vulnerable neighborhoods. Saisana, Saltelli, and Tarantola

(2005) describes incorporating uncertainty estimates and conducting sensitivity analysis on composite indexes. Tate (2012) and Tate (2013), similarly, make a comparative assessment of social vulnerability indexes based on uncertainty estimation and sensitivity analysis. Laimighofer and Laaha (2022) studies five uncertainty sources (record length, observation period, distribution choice, parameter estimation method, and GOF-test) of drought indexes.

There are also a few R packages supporting index calculation. The `SPEI` package (Vicente-Serrano, Beguería, and López-Moreno 2010) computes two drought indexes. The `gpindex` package (Martin 2023) computes price indexes, and the `fundiversity` package (Grenié and Gruson 2023) computes functional diversity indexes for ecological study. The package `COINr` (Becker et al. 2022) is more ambitious, making a start on following the broader guidelines in the OECD handbook to construct, analyze, and visualize composite indexes.

From reviewing this literature, and in the process of developing methods for making it easier to work with multivariate spatio-temporal data, it seems possible to think about indexes in a more organised, cohesive and standard manner. Actually, the area could benefit from a *tidy* approach.

4.3 Tidy framework

The tidy framework consists of two key components: tidy data and tidy tools. The concept of tidy data (Wickham 2014) prescribes specific rules for organizing data in an analysis, with observations as rows, variables as columns, and types of observational units as tables. Tidy tools, on the other hand, are concatenated in a sequence through which the tidy data flows, creating a pipeline for data processing and modeling. These pipelines are data-centric, meaning all the tidy tools or functions take a tidy data object as input and return a processed tidy data object, directly ready for the next operations to be applied. Also, the pipeline approach corresponds to the modular programming practice, which breaks down complex problems into smaller and more manageable pieces, as opposed to a monolithic design, where all the steps are predetermined and integrated into a single piece. The flexibility provided by the modularity makes it easier to modify certain steps in the pipeline and to maintain and extend the code base.

Examples of using a pipeline approach for data analysis can be traced back to the interactive graphics literature, including Buja et al. (1988); Sutherland et al. (2000); Wickham et al. (2009); Xie, Hofmann, and Cheng (2014). Wickham et al. (2009) argue that whether made explicit or not, a pipeline has to be presented in every graphics program, and making them explicit is

beneficial for understanding the implementation and comparing between different graphic systems. While this comment is made in the context of interactive graphics programs, it is also applicable generally to any data analysis workflow. More recently, the tidyverse suite (Wickham et al. 2019) takes the pipeline approach for general-purpose data wrangling and has gained popularity within the R community. The pipeline-style code can be directly read as a series of operations applied successively on tidy data objects, offering a method to document the data wrangling process with all the computational details for reproducibility.

Since the success of tidyverse, more packages have been developed to analyze data using the tidy framework for domains specific applications, a noticeable example of which is `tidymodels` for building machine learning models (Kuhn and Wickham 2020). To create a tidy workflow tailored to a specific domain, developers first need to identify the fundamental building blocks to create a workflow. These components are then implemented as modules, which can be combined to form the pipeline. For example, in supervised machine learning models, steps such as data splitting, model training, and model evaluation are commonly used in most workflow. In the `tidymodels`, these steps are correspondingly implemented as package `rsample`, `parsnip`, and `yardstick`, agnostic to the specific model chosen. The uniform interface in tidymodels frees analysts from recalling model-specific syntax for performing the same operation across different models, increasing the efficiency to work with different models simultaneously.

For constructing indexes, the pipeline approach adopts explicit and standalone modules that can be assembled in different ways. Index developers can choose the appropriate modules and arrange them accordingly to generate the data pipeline that is needed for their purpose. The pipeline approach provides many advantages:

- makes the computation more transparent, and thus more easily debugged.
- allows for rapidly processing new data to check how different features, like outliers, might affect the index value.
- provides the capacity to measure uncertainty by computing confidence intervals from multiple samples as generated by bootstrapping to original data.
- enables systematic comparison of surrogate indexes designed to measure the same phenomenon.
- it may even be possible to automate diagrammatic explanations and documentation of the index.

Adoption of this pipeline approach would provide uniformity to the field of index development, research, and application to improve comparability, reproducibility, and communication.

4.4 Details of the index pipeline

In constructing various indexes, the primary aim is to transform the data, often multivariate, into a univariate index. Spatial and temporal considerations are also factored into the process when observational units and time periods are not independent. However, despite the variations in contextual information for indexes in different fields, the underlying statistical methodology remains consistent across diverse domains. Each index can be represented as a series of modular statistical operations on the data. This allows us to decompose the index construction process into a unified pipeline workflow with a standardized set of data processing steps to be applied across different indexes.

An overview of the pipeline is presented in **?@fig-pipeline-steps**, illustrating the nine available modules designed to obtain the index from the data. These modules include operations for temporal and spatial aggregation, variable transformation and combination, distribution fitting, benchmark setting, and index communication. Analysts have the flexibility to construct indexes by connecting modules according to their preferences.

Now, we introduce the notation used for describing pipeline modules. Consider a multivariate spatio-temporal process,

$$\mathbf{x}(s;t) = \{x_1(s;t), x_2(s;t), \cdots, x_p(s;t)\} \qquad s \in D_s \subseteq \mathbb{R}^m, t \in D_t \subseteq \mathbb{R}^n \tag{4.1}$$

where:

- $x_j(s,t)$ represents a variable of interest for example precipitation, $j = 1, \cdots, p$ and
- s represents the geographic locations in the space $D_s \subseteq \mathbb{R}^m$. Examples of geographic locations include a collection of countries, longitude and latitude coordinates or regions of interest and,
- t denotes the temporal order in $D_t \subseteq \mathbb{R}^n$. For instance, time measurements could be recorded hourly, yearly, monthly, quarterly, or by season.

In what follows when geographic or temporal components of the $x_j(s,t)$ process are fixed we will be using suffix notation. For example, $x_{sj}(t)$ represents the data for a fixed location s as a function of time t. While $x_{tj}(s)$ denotes the spatial varying process for a fixed t. An overview of the notation for pipeline input, operation, and output is present in Table 1:

Table 1. Summary of the notation for input, operation, and output of each pipeline module.

Module	Input	Operation	Output
Temporal processing	$x_{sj}(t)$	$f[x_{sj}(t)]$	$x_{sj}^{\text{Temp}}(t') \quad t' \in D_{t'}$
Spatial processing	$x_{tj}(s)$	$g[x_{tj}(s)]$	$x_{tj}^{\text{Spat}}(s') \quad s' \in D_{s'}$
Variable transformation	$x_j(s;t)$	$T[x_j(s;t)]$	$x_j^{\text{Trans}}(s;t)$
Scaling	$x_j(s;t)$	$[x_j(s;t) - \alpha]/\gamma$	$x_j^{\text{Scale}}(s;t)$
Dimension reduction	$\mathbf{x}(s;t)$	$h[\mathbf{x}(s;t)]$	$\mathbf{y}(s;t) \quad \mathbf{y} \subseteq \mathbb{R}^d, d < p$
Distribution fit	$x_j(s;t)$	$F[x_j(s;t)]$	$P_j(s;t) \quad P(.) \in [0,1]$
Normalising	$x_j(s;t)$	$\Phi^{-1}[x_j(s;t)]$	$z_j(s;t)$
Benchmarking	$x_j(s;t)$	$u[x_j(s;t)]$	$b_j(s;t)$
Simplification	$x_j(s;t)$	$v[x_j(s;t)]$	$A_j(s;t) \in \{a_1, a_2, \cdots, a_z\}$

4.4.1 Temporal processing

The temporal processing module takes as input argument a single variable $x_{sj}(t)$ at location s as a function of time. In this step the original time series can be transformed or summarized into a new one via time aggregation, the transformation is represented by the function f, $x_{sj}^{\text{Temp}}(t') = f[x_{sj}(t)]$ where t' refers to the new temporal resolution after aggregation. An example of temporal processing done in the computation of the Standardized Precipitation Index (SPI) (McKee et al. 1993), consists of summing the monthly precipitation series over a rolling time window of size k. That is also known as the time scale. For SPI, the choice of the time scale k is used to control the accumulation period for the water deficit, enabling the assessment of drought severity across various types (meteorological, agricultural, and hydrological).

4.4.2 Spatial processing

The spatial processing module takes a single variable with a fixed temporal dimension, $x_{tj}(s)$, as input. This step transforms the variable from the original spatial dimension s into the new

dimension $s' \in D_{s'}$ through $x_{tj}^{\text{Spat}}(s') = g[x_{tj}(s)]$. The change of spatial dimension allows for the alignment of variables collected from different measurements, such as in-situ stations and satellite imagery, or originating from different resolutions. This also includes the aggregation of variables into different levels, such as city, state, and country scales.

4.4.3 Variable transformation

Variable transformation takes the input of a single variable $x_j(s;t)$ and reshapes its distribution using the function $T[.]$ to produce $x_j^{\text{Trans}}(s;t)$. When a variable has a skewed distribution, transformations such as log, square root, or cubic root can adjust the distribution towards normality. For example, in the Human Development Index (HDI), a logarithmic transformation is applied to the variable Gross National Income per capita (GNI), to reduce its impact on HDI, particularly for countries with high GNI values.

4.4.4 Scaling

Unlike variable transformation, scaling maintains the distributional shape of the variable. It includes techniques such as centering, z-score standardization, and min-max standardization and can be expressed as $[x_j(s;t) - \alpha]/\gamma$. In the Human Development Index (HDI), the three dimensions (health, education, and economy) are converted into the same scale (0-1) using min-max standardization.

Although the scaling might be considered to be a transformation, we have elected to make it a separate module because it is neater. Scaling simply changes the numbers in the data not the shape of a variable. Transformation will most likely change the shape, and is usually non-linear.

4.4.5 Dimension reduction

Dimension reduction takes the multivariate information $\mathbf{x}(s;t)$, where $\mathbf{x} \subseteq \mathbb{R}^p$, or a subset of variables in $\mathbf{x}(s;t)$, as the input. It summarises the high-dimensional information into a lower-dimension representation $\mathbf{y}(s;t)$, where $y \subseteq \mathbb{R}^d$ and $d < p$, as the output. The transformation can be based on domain-specific knowledge, originating from theories describing the underlying physical processes, or guided by statistical methods. For example, the Standardized Precipitation-Evapotranspiration Index (SPEI) (Beguería and Vicente-Serrano 2017) calculates the difference D between precipitation (P) and potential evapotranspiration (PET), using a

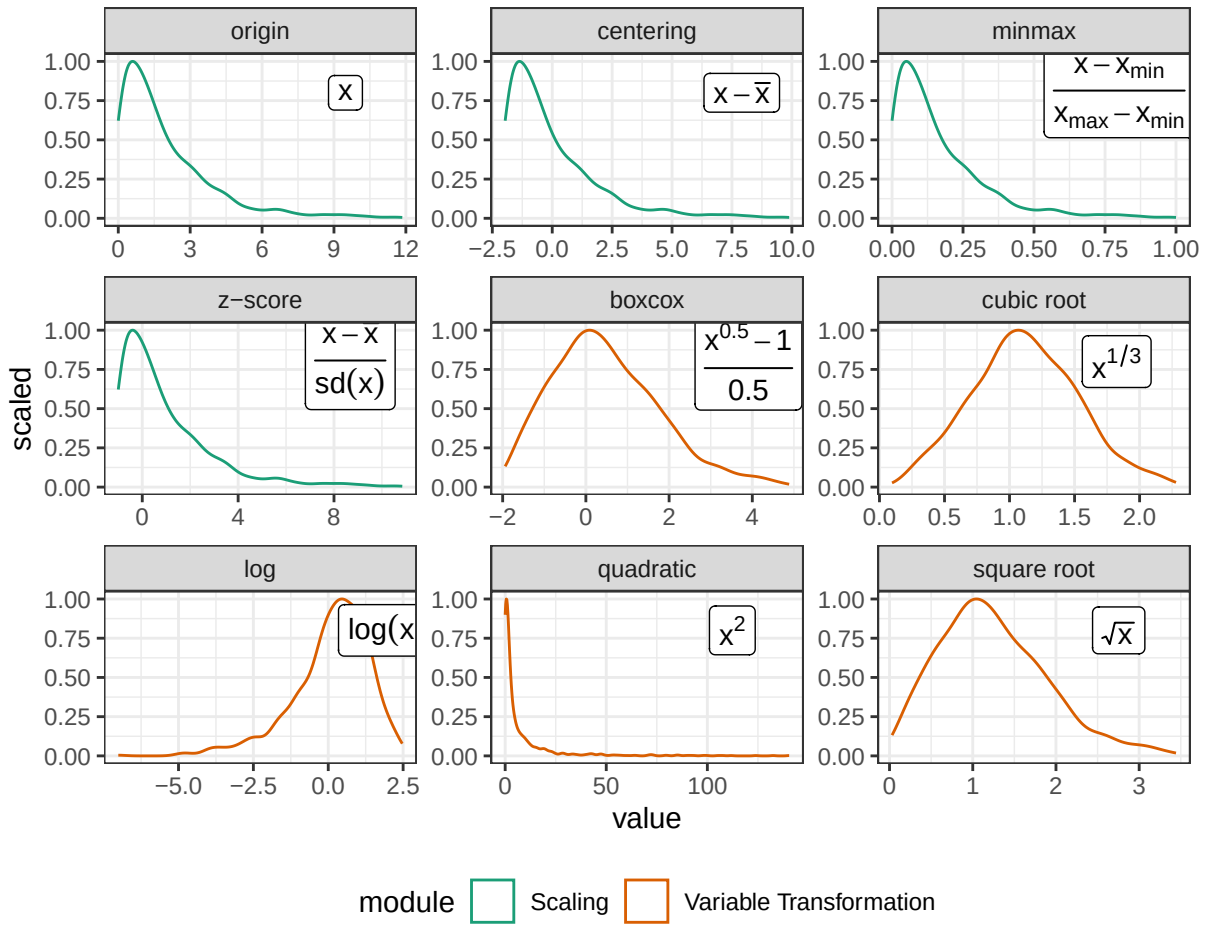

Figure 4.1: *Comparison of the module scaling (green) and variable transformation (orange). While both modules change the variable range, scaling maintains the same distributional shape, which is not the case with variable transformation.*

water balance model ($D = P - \text{PET}$). This is the only step that differs from the Standardized Precipitation Index (SPI), and can be considered to be a dimension reduction using a particular linear combination.

Linear combinations of variables are commonly used to reduce the dimension in statistical methodology, and chosen using a method like principal component analysis (PCA) (Hotelling 1933) or linear discriminant analysis (Fisher 1936), preparing contrasts to test particular elements in analysis of variance (Fisher 1970), or hand-crafted by a content-area expert. Linear combinations also form the basis for visualizing multivariate data, in methods such as tours (Wickham et al. 2011). This dimension reduction method can accommodate linear combinations as provided by any method, and hence is linear by design. The transformation module provides variable-wise non-linear transformation.

4.4.6 Distribution fit

Distribution fit applies the Cumulative Distribution Function (CDF) F of a distribution on the variable $x_j(s;t)$ to obtain the probability values $P_j(s;t) \in [0,1]$. In SPEI, many distributions, including log-logistic, Pearson III, lognormal, and general extreme distribution, are candidates for the aggregated series. Different fitting methods and different goodness of fit tests may be used to compare the distribution choice on the index value. This could be considered to be a variable transformation because it is usually conducted separately for each variable. However, very occasionally a fit is conducted on two or more variables simultaneously. For this reason, and because it usually is applied later in the pipeline it is neater to make this a separate module.

4.4.7 Normalising

Normalizing applies the inverse normal CDF Φ^{-1} on the input data to obtain the normal density $z_j(s;t)$. Normalizing can sometimes be confused with the scaling or variable transformation module, which does not involve using a normal distribution to transform the variable. It is arguably whether normalizing and distribution fit should be combined or separated into two modules. A decision has been made to separate them into two modules given the different types of output each module presents (probability values for distribution fit and normal density values for normalizing).

4.4.8 Benchmarking

Benchmark sets a value $b_j(s,t)$ for comparing against the original variable $x_j(s;t)$. This benchmark can be a fixed value consistently across space and time or determined by the data through the function $u[x_j(s;t)]$. Once a benchmark is set, observations can be highlighted for adjustments in other modules or can serve as targets for monitoring and planning.

4.4.9 Simplification

Simplification takes a continuous variable $x_j(s;t)$ and categorises it into a discrete set $A_j(s;t) \in \{a_1, a_2, \cdots, a_z\}$ through a piecewise constant function,

$$v[x_i(s;t)] = \begin{cases} a_0 & C_1 \leq x^i(s;t) < C_0 \\ a_1 & C_2 \leq x^i(s;t) < C_1 \\ a_2 & C_3 \leq x^i(s;t) < C_2 \\ \dots & \\ a_z & C_z \leq x^i(s;t) \end{cases} \tag{4.2}$$

This is typically used at the end of the index pipeline to simplify the index to communicate to the public the severity of the concept of interest measured by the index. An example of simplification is to map the calculated SPI to four categories: mild, moderate, severe, and extreme drought.

4.5 Software design

The R package `tidyindex` implements a proof-of-concept of the index pipeline modules described in Section 4.4. These modules compute an index in a sequential manner, as shown below:

```
DATA |>
  module1(...) |>
  module2(...) |>
  module3(...) |>
  ...
```

Each module offers a variety of alternatives, each represented by a distinct function. For example, within the `dimension_reduction()` module, three methods are available: `aggregate_linear()`, `aggregate_geometrical()`, and `manual_input()` and they can be used as:

```
dimension_reduction(V1 = aggregate_linear(...))
dimension_reduction(V2 = aggregate_geometrical(...))
dimension_reduction(V3 = manual_input(...))
```

Each method can be independently evaluated as a recipe, for example,

```
manual_input(~x1 + x2)
```

takes a formula to combine the variables `x1` and `x2` and return:

```
[1] "manual_input"
attr(,"formula")
[1] "x1 + x2"
attr(,"class")
[1] "dim_red"
```

This recipe will then be evaluated in the pipeline module with data to obtain numerical results. The package also offers wrapper functions that combine multiple steps for specific indexes. For instance, the `idx_spi()` function bundles three steps (temporal aggregation, distribution fit, and normalizing) into a single command, simplifying the syntax for computation. Analysts are also encouraged to create customized indexes from existing modules.

```
idx_spi <- function(...){
  DATA |>
    temporal_aggregate(...) |>
    distribution_fit(...)|>
    normalise(...)
}
```

The accompanied package, tidyindex, is not intended to offer an exhaustive implementation for all indexes across every domains. Instead, it provides a realization of the pipeline framework proposed in the paper. When adopting the pipeline approach to construct indexes, analysts may consider developing software that can be readily deployed in the cloud for production purposes.

4.6 Examples

This section uses the example of drought and social indexes to show the analysis made possible with the index pipeline. The drought index example computes two indexes (SPI and SPEI) with various time scales and distributions simultaneously using the pipeline framework to

understand the flood and drought events in Queensland. The second example focuses on the dimension reduction step in the Global Gender Gap Index to explore how the changes in linear combination weights affect the index values and country rankings.

4.6.1 Every distribution, every scale, every index all at once

The state of Queensland in Australia frequently experiences natural disaster events such as flood and drought, which can significantly impact its agricultural industry. This example uses daily data from Global Historical Climatology Network Daily (GHCND), aggregated into monthly precipitation, to compute two drought indexes – SPI and SPEI – at various time scales and fitted distributions, for 29 stations in the state of Queensland in Australia, spanning from January 1990 to April 2022. This example showcases the basic calculation of indexes with different parameter specifications within the pipeline framework. The dataset used in this example is available in the `tidyindex` package as `queensland` and blow prints the first few rows of the data:

```
# A tibble: 5 x 9
  id                ym  prcp  tmax  tmin  tavg  long   lat name
  <chr>          <mth> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 ASN00029038 1990 Jan  1682  34.3  24.7  29.5  142. -15.5 KOWANYAMA ~
2 ASN00029038 1990 Feb   416  35.2  23.2  29.2  142. -15.5 KOWANYAMA ~
3 ASN00029038 1990 Mar  2026  32.5  23.6  28.0  142. -15.5 KOWANYAMA ~
4 ASN00029038 1990 Apr   597  32.9  17.7  25.3  142. -15.5 KOWANYAMA ~
5 ASN00029038 1990 May   244  31.8  20.1  25.9  142. -15.5 KOWANYAMA ~
```

Figure 4.2 illustrates the pipeline steps of the two indexes. The two indexes are similar with the distinct that SPEI involves two additional steps – variable transformation and dimension reduction – prior to temporal processing. As introduced in Section 4.5, wrapper functions are available for both indexes as `idx_spi()` and `idx_spei()`, which allows for the specification of different time scales and distributions for fitting the aggregated series. In `tidyindex`, multiple indexes can be calculated collectively using the function `compute_indexes()`. Both SPI and SPEI are calculated across four time scales (6, 12, 24, and 36 months). The SPEI is fitted with two distributions (log-logistic and general extreme value distribution) and the gamma distribution is used for SPI:

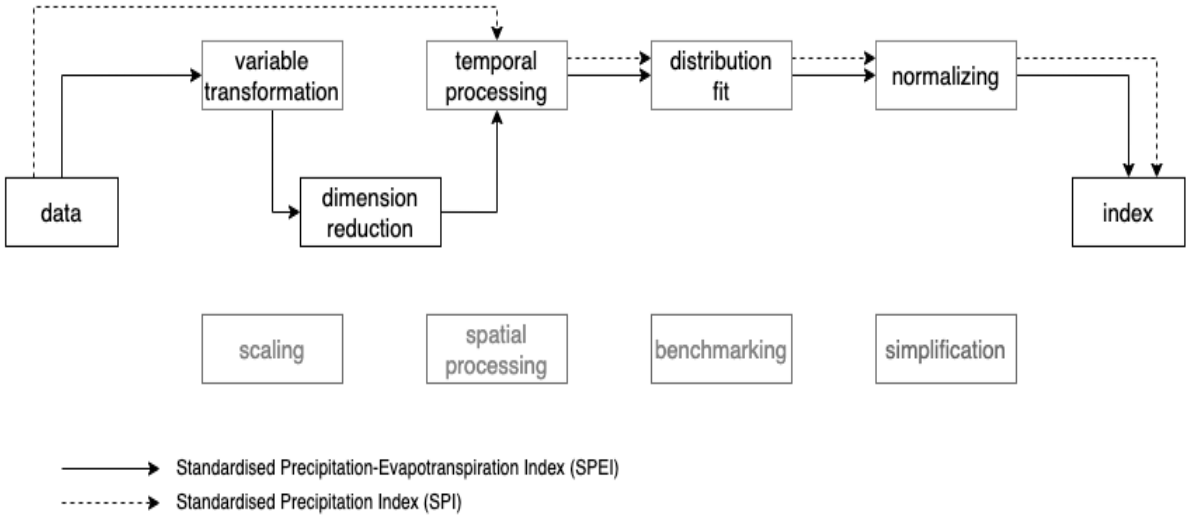

Figure 4.2: *Index pipeline for two drought indexes: the Standardized Precipitation Index (SPI) and the Standardized Precipitation-Evapotranspiration Index (SPEI). Both indexes share similar construction steps with SPEI having two steps additional steps (variable transformation and dimension reduction) to convert temperature into evapotranspiration and combine it with the precipitation series.*

```
.scale <- c(6, 12, 24, 36)
idx <- queensland %>%
  mutate(month = lubridate::month(ym)) |>
  init(id = id, time = ym, group = month) |>
  compute_indexes(
    spei = idx_spei(
      .tavg = tavg, .lat = lat,
      .scale = .scale, .dist = list(dist_gev(), dist_glo())),
    spi = idx_spi(.scale = .scale)
  )
```

```
# A tibble: 128,576 x 14
   .idx  .dist id              ym  prcp  tmax  tmin  tavg  long   lat
   <chr> <chr> <chr>         <mth> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 spei  gev   ASN000290~ 1990 Jun   170  29.7  16.2  23.0  142. -15.5
 2 spei  gev   ASN000290~ 1990 Jul   102  31.2  17.2  24.2  142. -15.5
 3 spei  gev   ASN000290~ 1990 Aug     0  31.3  13.1  22.2  142. -15.5
 4 spei  gev   ASN000290~ 1990 Sep     0  32.8  16.3  24.5  142. -15.5
```

```
 5 spei  gev   ASN000290~ 1990 Oct     0  36.8  21.5  29.2  142. -15.5
 6 spei  gev   ASN000290~ 1990 Nov   278  36.3  24.8  30.6  142. -15.5
 7 spei  gev   ASN000290~ 1990 Dec  1869  34.4  24.5  29.4  142. -15.5
 8 spei  gev   ASN000290~ 1990 Dec  1869  34.4  24.5  29.4  142. -15.5
 9 spei  gev   ASN000290~ 1991 Jan  5088  31.2  24.4  27.8  142. -15.5
10 spei  gev   ASN000290~ 1991 Jan  5088  31.2  24.4  27.8  142. -15.5
# i 128,566 more rows
# i 4 more variables: name <chr>, month <dbl>, .scale <dbl>,
#   .value <dbl>
```

The output contains the original data, index values (`.index`), parameters used (`.scale`, `.method`, and `.dist`), and all the intermediate variables (`pet`, `.agg`, and `.fitted`). This data can be visualized to investigate the spatio-temporal distribution of the drought or flood events, as well as the response of index values to different time scales and distribution parameters at specific single locations. Figure 4.3 and Figure 4.4 exemplify two possibilities. Figure 4.3 presents the spatial distribution of SPI during two periods: October 2010 to March 2011 for the 2010/11 Queensland flood and October 2019 to March 2020 for the 2019 Australia drought, which contributes to the notorious 2019/20 bushfire season. Figure 4.4 displays the sensitivity of the SPEI series at the Texas post office to different time scales and fitted distributions. Larger time scales produce a smoother index across time, however, all time scales indicate an extreme drought (corresponding to -2 in SPEI) in 2020, confirming the severity of the drought across different time horizons. Moreover, the chosen distribution has less influence on the index, with general extreme value distribution tending to produce more extreme outcomes than log-logistic distribution for the extreme events (index > 2 or <-2).

4.6.2 Does a puff of change in variable weights cause a tornado in ranks?

The Global Gender Gap Index (GGGI), published annually by the World Economic Forum, measures gender parity by assessing relative gaps between men and women in four key areas: Economic Participation and Opportunity, Educational Attainment, Health and Survival, and Political Empowerment (World Economic Forum 2023). The index, defined on 14 variables measuring female-to-male ratios, first aggregates these variables into four dimensions (using the linear combination given by `V-wgt` in Table 4.1). The weights are the inverse of the standard

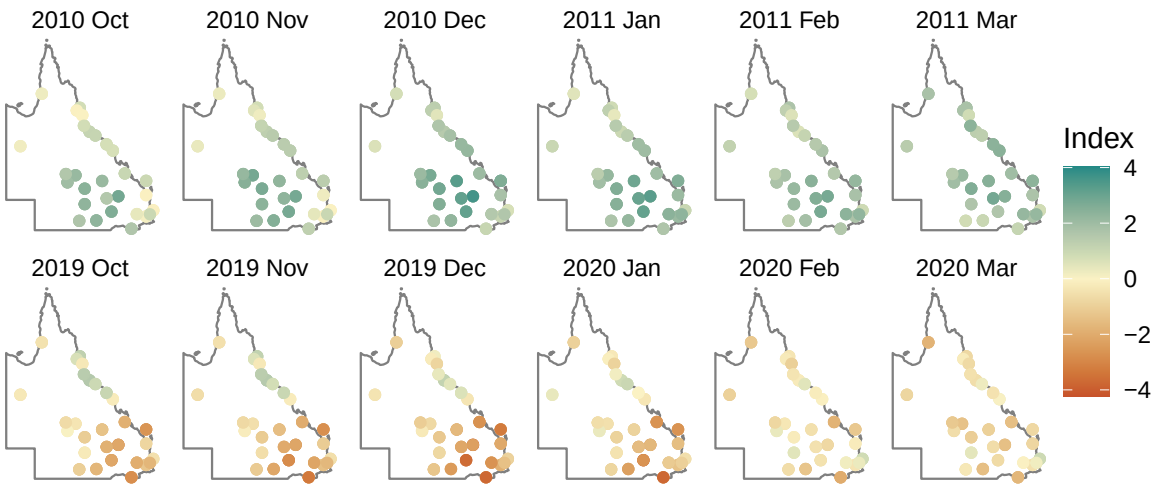

Figure 4.3: *Spatial distribution of Standardized Precipitation Index (SPI-12) in Queensland, Australia during two major flood and drought events: 2010/11 and 2019/20. The map shows a continuous wet period during the 2010/11 flood period and a mitigated drought situation, after its worst in 2019 December and 2020 January, likely due to the increased rainfall in February from the meteorological record.*

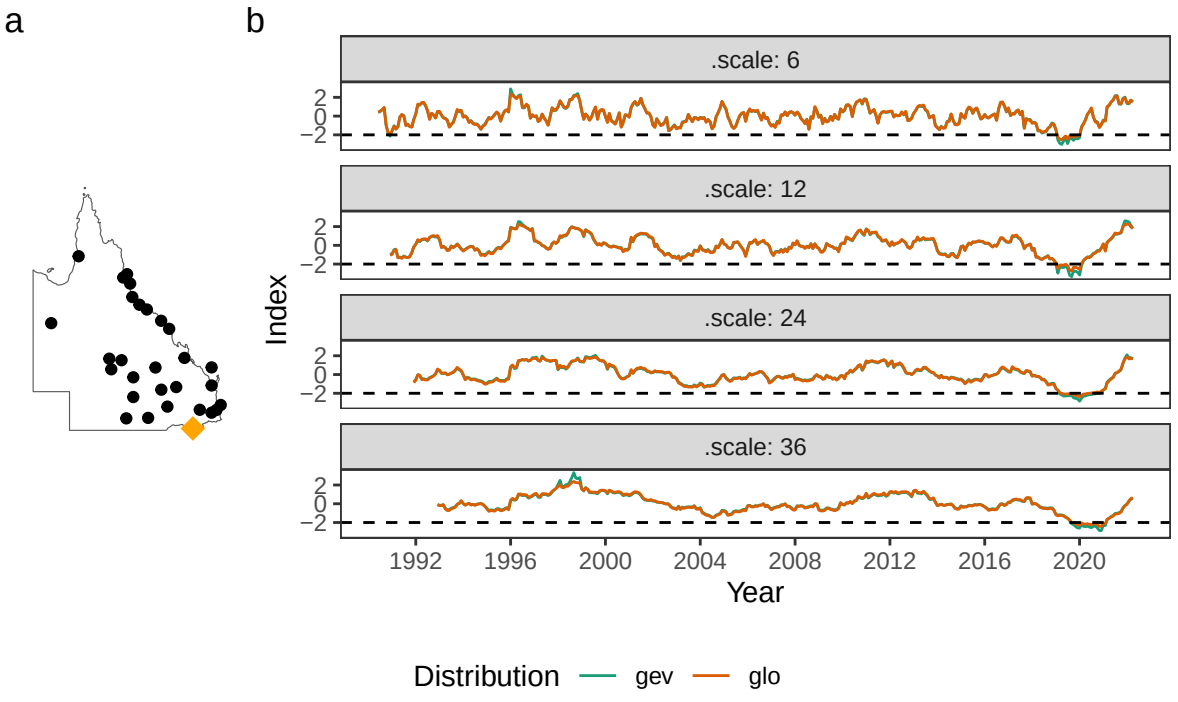

Figure 4.4: *Time series plot of Standardized Precipitation-Evapotranspiration Index (SPEI) at the Texas post office station (highlighted by a diamond shape in panel a). The SPEI is calculated at four time scales (6, 12, 24, and 36 months) and fitted with two distributions (Log Logistic and GEV). The dashed line at -2 represents the class "extreme drought" by the SPEI. A larger time scale gives a smoother index series, while also taking longer to recover from an extreme situation as seen in the 2019/20 drought period. The SPEI values from the two distribution fit mostly agree, while GEV can result in more extreme values, i.e. in 1998 and 2020.*

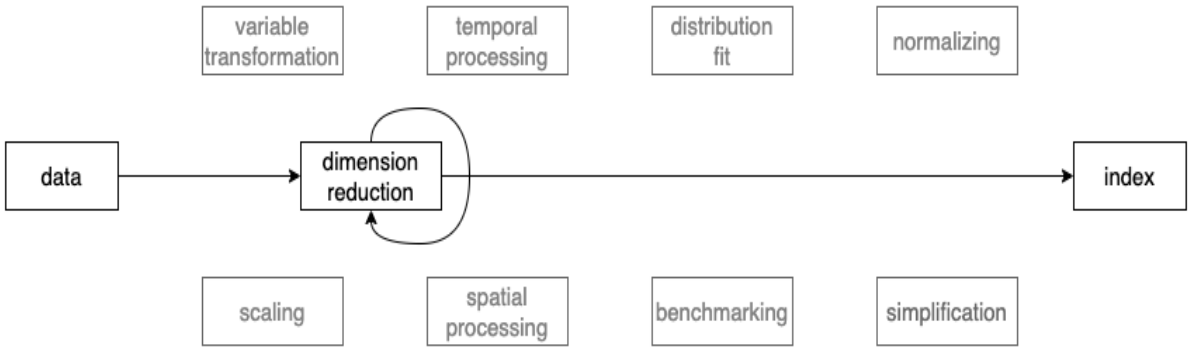

Figure 4.5: *Index pipeline for the Global Gender Gap Index (GGGI). The index is constructed as applying the module dimension reduction twice on the data.*

Table 4.1: *Weights for the two applications of dimension reduction to compute the Global Gender Gap Index. V-wgt is used to compute four new variables from the original 14. These are then equally combined to get the final index value.*

Variable	V-wgt	Dimension	D-wgt	wgt
Labour force participation	0.199	Economy	0.25	0.050
Wage equality for similar work	0.310			0.078
Estimated earned income	0.221			0.055
Legislators senior officials and managers	0.149			0.037
Professional and technical workers	0.121			0.030
Literacy rate	0.191	Education	0.25	0.048
Enrolment in primary education	0.459			0.115
Enrolment in secondary education	0.230			0.058
Enrolment in tertiary education	0.121			0.030
Sex ratio at birth	0.693	Health	0.25	0.173
Healthy life expectancy	0.307			0.077
Women in parliament	0.310	Politics	0.25	0.078
Women in ministerial positions	0.247			0.062
Years with female head of state	0.443			0.111

deviation of each variable, scaled to sum to 1, thus ensuring equal relative contribution of each variable to each of the four new variables. These new variables are then combined through another linear combination (`D-wgt` in Table 4.1) to form the final index value. Figure 4.5 illustrates that the pipeline is constructed by applying the dimension reduction module twice on the data. The data for GGGI does not needs to be transformed or scaled so these steps are not included, but they might still need to be used for other similar indexes.

The 2023 GGGI data is available from the Global Gender Gap Report 2023 in the country's economy profile and can be accessed in the `tidyindex` package as `gggi` with Table 4.1 as `gggi_weights`. The index can be reproduced with:

```
gggi %>%
  init(id = country) %>%
  add_meta(gggi_weights, var_col = variable) %>%
  dimension_reduction(
    index_new = aggregate_linear(
      ~labour_force_participation:years_with_female_head_of_state,
      weight = weight))
```

After initializing the `gggi` object and attaching the `gggi_weights` as meta-data, a single linear combination within the dimension reduction module is applied to the 14 variables (from column `labour_force_participation` to `years_with_female_head_of_state`), using the weight specified in the `wgt` column of the attached metadata. While computing the index from the original 14 variables, it remains unclear how the missing values are handled within the index, which impacts 68 out of the total 146 countries. However, after aggregating variables into the four dimensions, where no missing values exist, the index is reproducible for all the countries.

Figure 4.6 illustrates doing sensitivity analysis for GGGI, for a subset of 16 countries. Frame 12 shows the dot plot of the original index values sorted from highest to lowest. Leading the rankings are the Nordic countries (Iceland, Norway, and Finland) and New Zealand. The index values are between 0 and 1, and indicate proportional difference between men and women, with a value of 0.8 indicating women are 80% of the way to equality of these measures. There is a gap in values between these countries and the middle group (Brazil, Panama, Poland, Bangladesh, Kazakhstan, Armenia, and Slovakia), and another big drop to the next group (Pakistan, Iran, Algeria, and Chad). Afghanistan lags much further behind.

To make a simple illustration of sensitivity analysis, we slightly vary the weight for politics, between 0.07 and 0.52, while maintain equal weights among other dimensions. This can be viewed as an animation to examine change in relative index values as a response to the changing weights. This visualization technique, which presents a sequence of data projections, is referred to as a "tour" and the specific kind of tour used here to move between nearby projections is known as a "radial tour" (see Buja et al. (2005), Wickham et al. (2011), and Spyrison and Cook (2020) for more details).

Figure 4.6: *Exploring the sensitivity of the GGGI, by varying the politics component's contribution, for a subset of countries. Each panel shows a dotplot of the index values, computed for the linear combination represented by the segment plots below. Frame 12 shows the actual GGGI values, and countries are sorted from highest to lowest on this. Frames 1 and 6 show the GGGI if the politics component is reduced. Frames 18, 24, 29 show the GGGI when the politics component is increased. The most notable feature is that Bangladesh's GGGI drops substantially when politics is removed, indicating that this component plays a large role in it's relatively high value. Also, politics plays a substantial role in the GGGI's for the top ranked countries, because each of them drops, to the state of being similar to the middle ranked countries when the politics component's contribution is reduced. The animation can be viewed at* `https://vimeo.com/847874016`.

Frames 1 and 6 show linear combinations where politics contributes less than the original. It is interesting to note that the gap between the Nordic countries and the middle countries dissipates, indicating that this component was one reason for the relatively higher GGGI values of these countries. Also interesting is the large drop in value for Bangladesh.

Frames 18, 24, 29 show linear combinations where politics contributes more than the original. The most notable feature is that Bangladesh retains it's high index value whereas the other middle group countries decline, indicating that the politics score is a major component for Bangladesh's index value.

Ideally, an index should be robust against minor changes in its construction components. This is not the case with GGGI, where small changes to one component lead to fairly large change in the index. The modular pipeline framework for computing the index makes it easy to conduct this type of sensitivity analysis, where one or more components are perturbed and the index re-calculated. One aspect of the GGGI not well-described in the Global Gender Gap Report is handling of missing values that are present in the initial variables for many countries, something that is common for this type of data. This could also be made more transparent with the dimension reduction module, by specifying an imputation method or providing warnings about missing values.

4.6.3 Decoding uncertainty through the wisdom of the crowd

Errors in measurement, variability and sampling error, may arise at various stages of the pipeline calculation, including from different parameterization choices, as illustrated from Section 4.6.1, or from the statistical summarization procedures applied in the pipeline. Although it may not be possible to perfectly measure these errors, it is important that they are recognised and estimated for an index, so that it is possible to compute confidence intervals. In this example, the Texas post office station highlighted in Figure 4.4 is used to illustrate one possibility to compute a confidence interval for the Standardized Precipitation Index (SPI). Bootstrapping is used to account for the sampling uncertainty in the distribution fit step of the index pipeline and to assess its impact on the SPI series.

In SPI, the distribution fit step fits the gamma distribution to the aggregated precipitation series separately for each month. This results in 32 or 33 points, from January 1990 to April 2022, for estimating each set of distribution parameters. To account for this sampling uncertainty

with these samples, bootstrapping is used to generate replicates of the aggregated series. In the `tidyindex` package, this bootstrap sampling is activated when the argument `.n_boot` is set to a value other than the default of 1. In the following code, the Standardized Precipitation Index (SPI) is calculated using a time scale of 24. The bootstrap procedure samples the aggregated precipitation (`.agg`) for 100 iterations (`.n_boot = 100`) and then fits the gamma distribution. The resulting gamma probabilities are then transformed into normal densities in the normalizing step with `normalise()`.

```
DATA |>
  temporal_aggregate(.agg = temporal_rolling_window(prcp, scale = 24)) |>
  distribution_fit(.fit = dist_gamma(var = ".agg", method = "lmoms",
                                     .n_boot = 100)) |>
  normalise(.index = norm_quantile(.fit))
```

The confidence interval can then be calculated using the quantile method from the bootstrap samples. Figure 4.7 presents the 80% and 95% confidence interval for the Texas post office station, in Queensland, Australia. From the start of 2019 to 2020, the majority of the confidence intervals lie below the extreme drought line (SPI = -2), suggesting a high level of certainty that the Texas post office is suffering from a drastic drought. The relatively wide confidence interval, as well as during the excessive precipitation events in 1996-1998 and 1999-2000, suggests a high variation of the gamma parameters estimated from the bootstrap samples and its difficulty to accurately quantify the drought and flood severity in extreme events.

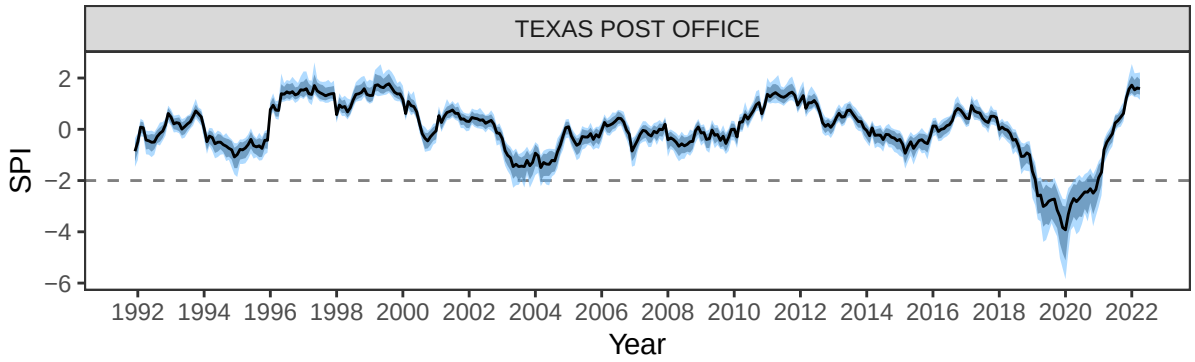

Figure 4.7: *80% and 95% confidence interval of the Standardized Precipitation Index (SPI-24) for the Texas post office station, in Queensland, Australia. A bootstrap sample of 100 is taken from the aggregated precipitation series to estimate gamma parameters and to calculate the index. The dashed line at SPI = -2 represents an extreme drought as defined by the SPI. Most parts of the confidence intervals from 2019 to 2020 sit below the extreme drought line and are relatively wide compared to other time periods. This suggests that while it is certain that the Texas post office is suffering from a drastic drought, there is considerable uncertainty in quantifying its severity, given the extremity of the event.*

4.7 Conclusion

The paper introduces a data pipeline comprising nine modules designed for the construction and analysis of indexes within the tidy framework. The pipeline offers a modular workflow to allow compute index with different parameterizations, to test minor changes to the original index definition, and to quantify uncertainties. The framework proposed in the paper is universal to index across diverse domains. Examples are provided, including the drought indexes (SPI and SPEI) and Global Gender Gap Index (GGGI), to demonstrate the index calculation with different time scales and distributions, to illustrate how slight adjustment of linear combination weights impact the index, and to calculate confidence intervals on the index.

4.8 Acknowledgement

This work is funded by a Commonwealth Scientific and Industrial Research Organisation (CSIRO) Data61 Scholarship. The article is created using Quarto (Allaire et al. 2022) in R (R Core Team 2021). The source code for reproducing this paper can be found at: https://github.com/huizezhang-sherry/paper-tidyindex.

Chapter 5

Conclusion

The three pieces of work in this thesis focus on diagnostics, data structure, and the pipeline workflow of contemporary multivariate spatio-temporal data. Chapter 2 introduces four plots that provide diagnostics of projection pursuit optimisation. When combined with guided tour, we can visualise the optimisation path of high-dimensional projection bases. While it is difficult to overcome failure encountered during optimisation in projection pursuit, the diagnostic plots implemented in this work provide a visual tool to understand its cause. These plots have been implemented in the R package `ferrn`, which has been on CRAN since March 2021 and has received over 8000 downloads from the CRAN mirror.

Many scientific fields heavily rely upon spatio-temporal data, which comprise time series data recorded at discrete locations. Examples of which include weather stations and river gauges. The second topic of the thesis aims to provide benefit to environmental scientists by making it easier to work with spatio-temporal data in different forms: pure spatial, pure temporal, and combined spatio-temporal. Chapter 3 discusses a new spatio-temporal data structure, applicable to both vector and raster data, that can easily pivot among these forms. Additionally, the data structure is also compatible with modern spatial and temporal data structure (`sf` and `tsibble`). The data structure is implemented in the `cubble` package, which has been on CRAN since August 2022 and has received over 45 stars on GitHub and over 5000 downloads from the CRAN mirror. The presentation of cubble on the Early Career and Student Statisticians Miniconference 2022 has won the People's Choice award.

Indexes are used in society to communicate complex multivariate information to the public or to make decisions. For example, drought and climate indexes guide the early warning monitoring systems in recommending actions for agriculture and other weather-dependent industries. The third topic of the thesis orchestrates a wide array of indexes proposed in various fields (in natural science, economics, and social science) to standardise the steps in index construction and analysis. The data pipeline proposed in Chapter 4 suggests a set of modulated steps for assembling indexes from multivariate spatio-temporal data under the tidy framework. The suggested pipeline offers a unified syntax for performing tasks that are universally applicable to indexes across all domains, including computing the index with different parameters, evaluating the robustness of the index, and calculating uncertainty measures. The `tiydindex` package that implements the data pipeline is currently a proof of concept for the pipeline workflow and is planned to be submitted onto CRAN.

5.1 Future work

5.1.1 Optimisers for projection pursuit optimization

The optimisation in projection pursuit is a high-dimensional problem that aims to optimise the interesting statistics, also called the index function, on the projection matrices. This objective function could be non-linear, computationally expensive to calculate the gradient, and may have local optima, which are also interesting for projection pursuit to explore, along with the global optimum. When approaching a new problem, users often try the three available stochastic optimizers introduced in Section 2.3.1, with different parameterizations, to find the optimizer works the best for the problem on hand. Optimizers that have shown success in other domain applications from the optimisation literature, such as genetic algorithms, can be tested for performance in addressing the projection pursuit optimisation.

5.1.2 Extending the glyphmap to other glyphs

The cubble approach transforms the time series lines from the temporal coordinates to the spatial coordinates and plots them as a glyph on the map. This is different from insetting a set of plots onto a base map, which could lead to plot overlap when observations are closely located. The glyph map approach can be generalised to transform different shapes including points and polygons, or even plots, from one set of coordinate to another. This would allow to display

more complex glyphs, for example, a scatterplot of two variables, against each other, to observe multivariate spatio-temporal relationship simultaneously on the map.

5.1.3 Applying the tidyindex framework for domain specific indexes

The tidyindex work introduced in Chapter 4 can be further expanded into a suite of tools, like tidymodel, for constructing and analysing indexes. The next step is to apply the index pipeline to specific domains, for example, drought indexes. Drought is a natural phenomenon with persistent lack of water and is monitored for meteorological, agricultural, hydrological, and social economic purpose. Despite hundreds of drought indexes proposed in the literature, there lacks a consensus among researchers on which index, or combination of indexes, to adopt for practical problems. The tidyindex pipeline can be extended to domain-specific toolkits for researchers to build, share, and compare their indexes.

5.2 Final remark

Figure 5.1 shows three five-letter words, each representing a keyword in the three main chapters of this thesis, arranged in the layout of a Wordle puzzle. Interestingly, "glyph" emerged as the Wordle solution on 18th November 2022, and "index" claimed its place on 15th August 2023 - just before the thesis submission! This evokes a similar feeling to come across the crossword clue "corolla leaf" (consider the variable name of the iris data), or alternatively, spot the Adélie penguins in the Sea Life.

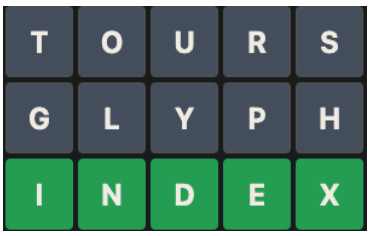

Figure 5.1: *Keywords of the three main chapters of this thesis, arranged in the layout of a Wordle puzzle.*

Bibliography

Alahacoon, Niranga, and Mahesh Edirisinghe. 2022. "A Comprehensive Assessment of Remote Sensing and Traditional Based Drought Monitoring Indices at Global and Regional Scale." *Geomatics, Natural Hazards and Risk* 13 (December): 762–99. https://doi.org/10.1080/19475705.2022.2044394.

Allaire, J. J., Charles Teague, Carlos Scheidegger, Yihui Xie, and Christophe Dervieux. 2022. *Quarto* (version 1.2). https://doi.org/10.5281/zenodo.5960048.

Andradóttir, Sigrún. 2015. "A Review of Random Search Methods." In *Handbook of Simulation Optimization*, 277–92. Springer. https://doi.org/10.1007/978-1-4939-1384-8.

Becker, William, Giulio Caperna, Maria Del Sorbo, Hedvig Norlen, Eleni Papadimitriou, and Michaela Saisana. 2022. "COINr: An R Package for Developing Composite Indicators." *Journal of Open Source Software* 7 (78): 4567. https://doi.org/10.21105/joss.04567.

Beguería, Santiago, and Sergio M. Vicente-Serrano. 2017. *SPEI: Calculation of the Standardised Precipitation-Evapotranspiration Index*. https://CRAN.R-project.org/package=SPEI.

Bertsekas, Dimitri P. 2014. *Constrained Optimization and Lagrange Multiplier Methods*. Academic press. https://doi.org/10.1016/C2013-0-10366-2.

Bertsimas, Dimitris, and John Tsitsiklis. 1993. "Simulated Annealing." *Statistical Science* 8 (1): 10–15. https://doi.org/10.1214/ss/1177011077.

Bivand, Roger S, Edzer J Pebesma, Virgilio Gómez-Rubio, and Edzer Jan Pebesma. 2008. *Applied Spatial Data Analysis with r*. Vol. 747248717. Springer.

Brooks, Samuel H. 1958. "A Discussion of Random Methods for Seeking Maxima." *Operations Research* 6 (2): 244–51. https://doi.org/10.1287/opre.6.2.244.

Buja, Andreas, and Daniel Asimov. 1986. "Grand Tour Methods: An Outline." In *Proceedings of the Seventeenth Symposium on the Interface of Computer Sciences and Statistics*, 63–67. USA: Elsevier North-Holland, Inc. https://dl.acm.org/doi/10.5555/26036.26046.

Buja, Andreas, Daniel Asimov, Catherine Hurley, and John A McDonald. 1988. "Elements of a Viewing Pipeline for Data Analysis." In *Dynamic Graphics for Statistics*, 277–308. Wadsworth, Belmont.

Buja, Andreas, Dianne Cook, Daniel Asimov, and Catherine Hurley. 2005. "Computational Methods for High-Dimensional Rotations in Data Visualization." *Handbook of Statistics* 24: 391–413. https://doi.org/10.1016/S0169-7161(04)24014-7.

Buja, Andreas, Dianne Cook, and Deborah F Swayne. 1996. "Interactive High-Dimensional Data Visualization." *Journal of Computational and Graphical Statistics* 5 (1): 78–99. https://doi.org/10.2307/1390754.

Cheng, Joe, and Carson Sievert. 2021. *Crosstalk: Inter-Widget Interactivity for HTML Widgets*. https://CRAN.R-project.org/package=crosstalk.

Cheng, Xiaoyue, Dianne Cook, and Heike Hofmann. 2016. "Enabling Interactivity on Displays of Multivariate Time Series and Longitudinal Data." *Journal of Computational and Graphical Statistics* 25 (4): 1057–76. https://doi.org/10.1080/10618600.2015.1105749.

Conn, Andrew R, Katya Scheinberg, and Luis N Vicente. 2009. *Introduction to Derivative-Free Optimization*. SIAM. https://doi.org/10.1137/1.9780898718768.

Cook, Dianne, and Andreas Buja. 1997. "Manual Controls for High-Dimensional Data Projections." *Journal of Computational and Graphical Statistics* 6 (4): 464–80. https://doi.org/10.2307/1390747.

Cook, Dianne, Andreas Buja, and Javier Cabrera. 1993. "Projection Pursuit Indexes Based on Orthonormal Function Expansions." *Journal of Computational and Graphical Statistics* 2 (3): 225–50. https://doi.org/10.2307/1390644.

Cook, Dianne, Andreas Buja, Javier Cabrera, and Catherine Hurley. 1995. "Grand Tour and Projection Pursuit." *Journal of Computational and Graphical Statistics* 4 (3): 155–72. https://doi.org/10.1080/10618600.1995.10474674.

Cook, Dianne, Andreas Buja, Eun-Kyung Lee, and Hadley Wickham. 2008. "Grand Tours, Projection Pursuit Guided Tours, and Manual Controls." In *Handbook of Data Visualization*, 295–314. Springer. https://doi.org/10.1007/978-3-540-33037-0_13.

Diaconis, Persi, and David Freedman. 1984. "Asymptotics of Graphical Projection Pursuit." *The Annals of Statistics*, 793–815. https://doi.org/10.1214/aos/1176346703.

Fisher, Ronald. 1936. "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics* 7 (2): 179–88.

———. 1970. "Statistical Methods for Research Workers." In *Breakthroughs in Statistics: Methodology and Distribution*, 66–70. Springer. https://doi.org/10.1007/978-1-4612-4380-9_6.

Friedman, Jerome H, and John W Tukey. 1974. "A Projection Pursuit Algorithm for Exploratory Data Analysis." *IEEE Transactions on Computers* 100 (9): 881–90. https://doi.org/10.1109/T-C.1974.224051.

Granville, Vincent, Mirko Krivánek, and J-P Rasson. 1994. "Simulated Annealing: A Proof of Convergence." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16 (6): 652–56. https://doi.org/10.1109/34.295910.

Grenié, Matthias, and Hugo Gruson. 2023. *fundiversity: Easy Computation of Functional Diversity Indices*. https://doi.org/10.5281/zenodo.4761754.

Hall, Peter. 1989. "On Polynomial-Based Projection Indices for Exploratory Projection Pursuit." *The Annals of Statistics* 17 (2): 589–605. https://doi.org/10.1214/aos/1176347127.

Hao, Zengchao, and Vijay P. Singh. 2015. "Drought Characterization from a Multivariate Perspective: A Review." *Journal of Hydrology* 527 (August): 668–78. https://doi.org/10.1016/j.jhydrol.2015.05.031.

Healy, Kieran. 2018. *Data Visualization: A Practical Introduction*. Princeton University Press. https://socviz.co/.

Hersbach, Hans, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, et al. 2020. "The ERA5 Global Reanalysis." *Quarterly Journal of the Royal Meteorological Society* 146 (730): 1999–2049. https://doi.org/10.1002/qj.3803.

Hotelling, Harold. 1933. "Analysis of a Complex of Statistical Variables into Principal Components." *Journal of Educational Psychology* 24 (6): 417.

Jones, Brenda, and Jean Andrey. 2007. "Vulnerability Index Construction: Methodological Choices and Their Influence on Identifying Vulnerable Neighbourhoods." *International Journal of Emergency Management* 4 (2): 269–95. https://doi.org/10.1504/IJEM.2007.013994.

Jones, Donald, Matthias Schonlau, and William Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization* 13 (4): 455–92. https://doi.org/10.1023/A:1008306431147.

Kirkpatrick, Scott, C Daniel Gelatt, and Mario P Vecchi. 1983. "Optimization by Simulated Annealing." *Science* 220 (4598): 671–80. https://doi.org/10.1126/science.220.4598.671.

Kuhn, Max, and Hadley Wickham. 2020. *Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles.* https://www.tidymodels.org.

Laa, Ursula, and Dianne Cook. 2020. "Using Tours to Visually Investigate Properties of New Projection Pursuit Indexes with Application to Problems in Physics." *Computational Statistics*, 1–35. https://doi.org/10.1007/s00180-020-00954-8.

Laimighofer, Johannes, and Gregor Laaha. 2022. "How Standard Are Standardized Drought Indices? Uncertainty Components for the SPI & SPEI Case." *Journal of Hydrology* 613 (October): 128385. https://doi.org/10.1016/j.jhydrol.2022.128385.

Lee, Eun-Kyung, and Dianne Cook. 2010. "A Projection Pursuit Index for Large p Small n Data." *Statistics and Computing* 20 (3): 381–92. https://doi.org/10.1007/s11222-009-9131-1.

Lee, Eun-Kyung, Dianne Cook, Sigbert Klinke, and Thomas Lumley. 2005. "Projection Pursuit for Exploratory Supervised Classification." *Journal of Computational and Graphical Statistics* 14 (4): 831–46. https://doi.org/10.1198/106186005X77702.

Li, Mingwei, Zhenge Zhao, and Carlos Scheidegger. 2020. "Visualizing Neural Networks with the Grand Tour." *Distill*. https://doi.org/10.23915/distill.00025.

Loperfido, Nicola. 2018. "Skewness-Based Projection Pursuit: A Computational Approach." *Computational Statistics and Data Analysis* 120 (C): 42–57. https://doi.org/10.1016/j.csda.2017.11.001.

———. 2020. "Kurtosis-Based Projection Pursuit for Outlier Detection in Financial Time Series." *The European Journal of Finance* 26 (2-3): 142–64. https://doi.org/10.1080/1351847X.2019.1647864.

Lovelace, Robin, Jakub Nowosad, and Jannes Muenchow. 2019. *Geocomputation with r*. CRC Press.

Martin, Steve. 2023. *Gpindex: Generalized Price and Quantity Indexes*. https://CRAN.R-project.org/package=gpindex.

McKee, Thomas B, Nolan J Doesken, John Kleist, et al. 1993. "The Relationship of Drought Frequency and Duration to Time Scales." In *Proceedings of the 8th Conference on Applied Climatology*, 17:179–83. 22. Boston, MA, USA.

Menne, Matthew J, Imke Durre, Russell S Vose, Byron E Gleason, and Tamara G Houston. 2012. "An Overview of the Global Historical Climatology Network-Daily Database." *Journal of Atmospheric and Oceanic Technology* 29 (7): 897–910. https://doi.org/10.1175/JTECH-D-11-00103.1.

Mitra, Debasis, Fabio Romeo, and Alberto Sangiovanni-Vincentelli. 1986. "Convergence and Finite-Time Behavior of Simulated Annealing." *Advances in Applied Probability* 18 (3): 747–71. https://doi.org/10.1109/CDC.1985.268600.

OECD, European Union, and Joint Research Centre - European Commission. 2008. *Handbook on Constructing Composite Indicators: Methodology and User Guide*. OECD. https://doi.org/10.1787/9789264043466-en.

Pebesma, Edzer. 2012. "Spacetime: Spatio-Temporal Data in R." *Journal of Statistical Software* 51 (7): 1–30. https://doi.org/10.18637/jss.v051.i07.

———. 2021. *Stars: Spatiotemporal Arrays, Raster and Vector Data Cubes*. https://CRAN.R-project.org/package=stars.

Pebesma, Edzer, and Roger Bivand. 2019. "Spatial Data Science." CRC Press: Boca Raton, FL, USA.

———. 2022. "CRAN Task View: Handling and Analyzing Spatio-Temporal Data." https://CRAN.R-project.org/view=SpatioTemporal.

Pierce, David. 2019. *Ncdf4: Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files*. https://CRAN.R-project.org/package=ncdf4.

Posse, Christian. 1995. "Projection Pursuit Exploratory Data Analysis." *Computational Statistics & Data Analysis* 20 (6): 669–87. https://doi.org/10.1016/0167-9473(95)00002-8.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rios, Luis Miguel, and Nikolaos V Sahinidis. 2013. "Derivative-Free Optimization: A Review of Algorithms and Comparison of Software Implementations." *Journal of Global Optimization* 56 (3): 1247–93. https://doi.org/10.1007/s10898-012-9951-y.

Romeijn, H. Edwin. 2009. "Random Search Methods." In *Encyclopedia of Optimization*, 3245–51. Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-74759-0_556.

Saisana, M., A. Saltelli, and S. Tarantola. 2005. "Uncertainty and Sensitivity Analysis Techniques as Tools for the Quality Assessment of Composite Indicators." *Journal of the Royal Statistical Society Series A: Statistics in Society* 168 (2): 307–23. https://doi.org/10.1111/j.1467-985X.2005.00350.x.

Schloerke, Barret. 2016. *Geozoo: Zoo of Geometric Objects*. https://CRAN.R-project.org/package=geozoo.

Schloerke, Barret, Dianne Cook, Joseph Larmarange, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg, and Jason Crowley. 2021. *GGally: Extension to Ggplot2*. https://CRAN.R-project.org/package=GGally.

Simmons, Adrian, Mariano Hortal, Graeme Kelly, Anthony McNally, Agathe Untch, and Sakari Uppala. 2005. "ECMWF Analyses and Forecasts of Stratospheric Winter Polar Vortex Breakup: September 2002 in the Southern Hemisphere and Related Events." *Journal of the Atmospheric Sciences* 62 (3): 668–89. https://doi.org/10.1175/JAS-3322.1.

Simmons, Adrian, Cornel Soci, Julien Nicolas, Bill Bell, P. Berrisford, Rossana Dragani, Johannes Flemming, et al. 2020. "Global Stratospheric Temperature Bias and Other Stratospheric Aspects of ERA5 and ERA5.1," no. 859 (January). https://doi.org/10.21957/rcxqfmg0.

Spall, James C. 2005. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Vol. 65. John Wiley & Sons. https://www.jhuapl.edu/ISSO/.

Spyrison, Nicholas, and Dianne Cook. 2020. "The r Journal: Spinifex: An r Package for Creating a Manual Tour of Low-Dimensional Projections of Multivariate Data." *The R Journal* 12: 243–57. https://doi.org/10.32614/RJ-2020-027.

Sutherland, Peter, Anthony Rossini, Thomas Lumley, Nicholas Lewin-Koh, Julie Dickerson, Zach Cox, and Dianne Cook. 2000. "Orca: A Visualization Toolkit for High-Dimensional Data." *Journal of Computational and Graphical Statistics* 9 (3): 509–29. https://www.jstor.org/stable/1390943.

Svoboda, Mark, Brian Fuchs, et al. 2016. "Handbook of Drought Indicators and Indices." *Drought and Water Crises: Integrating Science, Management, and Policy*, 155–208.

Tate, Eric. 2012. "Social Vulnerability Indices: A Comparative Assessment Using Uncertainty and Sensitivity Analysis." *Natural Hazards* 63 (2): 325–47. https://doi.org/10.1007/s11069-012-0152-2.

———. 2013. "Uncertainty Analysis for a Social Vulnerability Index." *Annals of the Association of American Geographers* 103 (3): 526–43. https://doi.org/10.1080/00045608.2012.700616.

Teickner, Henning, Edzer Pebesma, and Benedikt Graeler. 2022. *Sftime: Classes and Methods for Simple Feature Objects That Have a Time Column*.https://github.com/r-spatial/sftime.

Tukey, John W. 1977. *Exploratory Data Analysis*. Vol. 2. Reading, MA.

Unwin, Antony. 2015. *Graphical Data Analysis with r*. Vol. 27. CRC Press. https://doi.org/10.1201/9781315370088.

Vicente-Serrano, Sergio M., Santiago Beguería, and Juan I. López-Moreno. 2010. "A Multiscalar Drought Index Sensitive to Global Warming: The Standardized Precipitation Evapotranspiration Index." *Journal of Climate* 23 (7): 1696–1718. https://journals.ametsoc.org/view/journals/clim/23/7/2009jcli2909.1.xml.

Wang, Earo, Dianne Cook, and Rob J Hyndman. 2020. "A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data." *Journal of Computational and Graphical Statistics* 29 (3): 466–78. https://doi.org/10.1080/10618600.2019.1695624.

White, R C. 1971. "A Survey of Random Methods for Parameter Optimization." *Simulation* 17 (5): 197–205. https://doi.org/10.1177/003754977101700504.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (September): 1–23. https://doi.org/10.18637/jss.v059.i10.

———. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://doi.org/10.1007/978-0-387-98141-3.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the Tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Dianne Cook, Heike Hofmann, and Andreas Buja. 2011. "Tourr: An R Package for Exploring Multivariate Data with Projections." *Journal of Statistical Software* 40 (2). https://doi.org/10.18637/jss.v040.i02.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation*.

Wickham, Hadley, Heike Hofmann, Charlotte Wickham, and Dianne Cook. 2012. "Glyph-Maps for Visually Exploring Temporal Patterns in Climate Data and Models." *Environmetrics* 23 (5): 382–93. https://doi.org/10.1002/env.2152.

Wickham, Hadley, Michael Lawrence, Dianne Cook, Andreas Buja, Heike Hofmann, and Deborah F. Swayne. 2009. "The Plumbing of Interactive Graphics." *Computational Statistics* 24 (2): 207–15. https://doi.org/10.1007/s00180-008-0116-x.

Wilke, Claus O. 2019. *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. O'Reilly Media. https://clauswilke.com/dataviz/.

Wilkinson, Leland, Anushka Anand, and Robert Grossman. 2005. "Graph-Theoretic Scagnostics." In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, 157–64. https://doi.org/10.1109/INFVIS.2005.1532142.

Wilkinson, Leland, and Graham Wills. 2008. "Scagnostics Distributions." *Journal of Computational and Graphical Statistics* 17 (2): 473–91.

World Economic Forum. 2023. "The Global Gender Gap Report 2023." https://www3.weforum.org/docs/WEF_GGGR_2023.pdf.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.name/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Heike Hofmann, and Xiaoyue Cheng. 2014. "Reactive Programming for Interactive Graphics." *Statistical Science* 29 (2): 201–13. https://www.jstor.org/stable/43288470?seq=1.

Zabinsky, Zelda B. 2013. *Stochastic Adaptive Search for Global Optimization*. Vol. 72. Springer Science & Business Media. https://doi.org/10.1007/978-1-4419-9182-9.

Zargar, Amin, Rehan Sadiq, Bahman Naser, and Faisal I Khan. 2011. "A Review of Drought Indices." *Environmental Reviews* 19 (NA): 333–49. https://www.jstor.org/stable/envirevi.19.333.

Zhang, H. Sherry, Dianne Cook, Ursula Laa, Nicolas Langrené, and Patricia Menéndez. 2021. *Ferrn: Facilitate Exploration of touRR optimisatioN*. https://github.com/huizezhang-sherry/ferrn/.

———. 2023a. *Cubble: A Vector Spatio-Temporal Data Structure for Data Analysis*. https://github.com/huizezhang-sherry/cubble.

———. 2023b. *Tidyindex: A General Data Pipeline for Computing Indexes*. https://github.com/huizezhang-sherry/tidyindex.