Visual Diagnostics for Constrained Optimisation with Application to Guided Tours

by H.Sherry Zhang, Dianne Cook, Ursula Laa, Nicolas Langrené, Patricia Menéndez

Abstract Guided tour searches for interesting low-dimensional views of high-dimensional data via optimising a projection pursuit index function. The first paper of projection pursuit by Friedman and Tukey (1974) stated that "the technique used for maximising the projection index strongly influences both the statistical and the computational aspects of the procedure." While many indices have been proposed in the literature, less work has been done on evaluating the performance of the optimisers. In this paper, we implement a data collection object for the optimisation in the guided tour and introduce visual diagnostics based on the data object collected. These diagnostics and workflows can be applied to a broad class of optimisers, to assess their performance. An R package, **ferrn**, has been created to implement the diagnostics.

Introduction

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

The goal of continuous optimisation is to find the best solution within the space of all feasible solutions where typically the best solution is decided by an objective function. Broadly speaking, optimization can be unconstrained or constrained (Kelley, 1999). The unconstrained problem can be formulated as a minimization (or maximization) problem such as $\min_x f(x)$ where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is an objective function with certain properties defined in an L^p space. In this case, solutions rely on gradient descent or ascent methods. In the constrained optimization problem additional restrictions are introduced via a set of functions that can be convex or non-convex: $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ for $i = 1, \dots k$ and hence the problem can be written as $\min_x f(x)$ *subject to* $g_i(x) \leq 0$. Here methods such as Langrange multipliers and convex optimization methods including linear and quadratic programming can be used.

The focus of this paper is on the optimisation problem arising in the projection pursuit guided tour (Buja et al., 2005) which is an exploratory data analysis tool that is defined to detect *interesting structures* or features in high-dimensional data through a set of lower-dimensional projections that cover the entire high dimensional space using interpolation methods called tours (Cook et al., 2008). The target of the optimisation is to identify the most *interesting* low-dimensional views of the data given by a corresponding projection matrix. The most *interesting* structures are formally defined by a function of projections, called index function which is optimized to uncover the most revelling structures in a high dimensional space (Cook et al., 1993).

The optimization challenges encountered in the projection pursuit guided tour problem are common to those of optimization in general. Examples of those include the existence of multiple maxima (local and global), the trade off between computational burden and proximity to the maxima, dealing with noisy objective functions that might be non smooth and non differentiable (Jones et al., 1998). Those are not unique to this context and therefore the visualization tools and optimization methods presented in this paper can be easily applied to any other optimization problems.

The remainder of the paper is organised as follows. Section 2.2 provides an overview of optimisation methods, specifically line search methods. Section 2.3 reviews projection pursuit guided tour, defines the optimisation problem and introduces three existing algorithms. Section 2.4 presents the new visual diagnostics. A data structure is defined to capture information during the optimisation, and used in different types of diagnostic plots. Section 2.5 shows applications of how these plots can be used to understand and compare different algorithms. We also discuss how these insights contribute to modifications that improve the algorithms. Finally, Section 2.6 describes the R package: **ferrn**, that implements the visual diagnostics.

Optimisation methods

Optimization problems are ubiquitous in many areas of study. While in some cases analytical solutions can be found, the majority of problems rely on numerical methods to find the optimal solution. These numerical methods follow iterative approaches that aim at finding the optimum by progressively improving the current solution until a desirable accuracy is achieved. Although this principle seems uncomplicated, a number of challenges arise such as the possible existence of multiple maxima (local and global), constraints and noisy objective function, and the trade-off between desirable accuracy and computational burden. In addition, the optimization results might depend on the algorithm starting values, affecting the consistency of results.

Optimization methods can be divided into various classes, such as global optimisation (Kelley, 1999; Fletcher, 2013), convex optimisation (Boyd et al., 2004) or stochastic optimisation (Nocedal and Wright, 2006). Our interest is on constrained optimization (Bertsekas, 2014) as defined in the introduction section, and assuming it is not possible to find a solution to the problem in the way of a closed-form. That is, the problem consists of finding the minimum or maximum of a function $f \in L^p$ in the constrained $\mathbb{A}$ space.

A large class of methods utilises the gradient information of the objective function to perform the optimisation iterations, with the most notable one being the gradient ascent (descent) method. Although gradient optimization methods are popular, they rely on the availability of the objective function derivatives and on the complexity of the constraints. Derivative-free methods, 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. Examples of those include response surface methodology (Box and Wilson, 1951), stochastic approximation (Robbins and Monro, 1951), random search (Fu, 2015) and heuristic methods (Sörensen and Glover, 2013). Later, we will present a simulated annealing optimisation algorithm, which belongs to the class of random search methods, for optimisation with the guided tour.

A common search scheme utilised by both derivative-free methods and gradient methods is 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 desire convergence is reached. While gradient-based methods choose the search direction by the gradient, derivative-free methods uses local information of the objective function to determine the search direction. The choice of step size also needs considerations, as inadequate step sizes might prevent the optimisation method to converge to an optimum. An ideal step size can be chosen via 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.

Several R implementations address optimization problems with both general purpose as well as task specifics solvers. The most prominent one within the general solvers is `optim()` in the **stats** (R Core Team, 2020) package, which provides both gradient-based and derivative-free optimisation functions. Another general solver specialised in non-linear optimisation is **nloptr** (Johnson, 2020). Specific solvers for simulated annealing includes `optim(...,method "SANN")` and package **GenSA** (Xiang et al., 2013) that deals with more complicated objective functions. For other task specific solvers, readers are recommended to visit the relevant sections in CRAN task review on *optimisation and mathematical programming* (Theussl et al., 2020).

Projection pursuit guided tour

The projection pursuit guided tour combines two different methods in exploratory data analysis, focusing on different aspects. Projection pursuit, coined by Friedman and Tukey (1974), detects interesting structures (e.g. clustering, outliers and skewness) in multivariate data via low dimensions projection. The guided tour is using ideas from projection pursuit to define a particular variation in a broader class of data visualisation methods, building on the grand tour approach (Asimov, 1985).

To define projection pursuit, we first need to establish the notation used. Let $\mathbf{X}_{n\times p}$ be the data matrix, with n observations in p dimensions. A d-dimensional projection can be seen as a linear transformation from $\mathbb{R}^p$ into $\mathbb{R}^d$, and 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. Define $f : \mathbb{R}^{n\times d} \mapsto \mathbb{R}$ to be an index function that maps the projected data $\mathbf{Y}$ (corresponding to an associated projection matrix $\mathbf{A}$) onto an index value I (QUESTION: Isn't f the index function?). 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.

A number of index functions have been proposed in the literature to detect different data structures, including Legendre index (Friedman and Tukey, 1974), Hermite index (Hall et al., 1989), natural Hermite index (Cook et al., 1993), chi-square index (Posse, 1995), LDA index (Lee et al., 2005) for

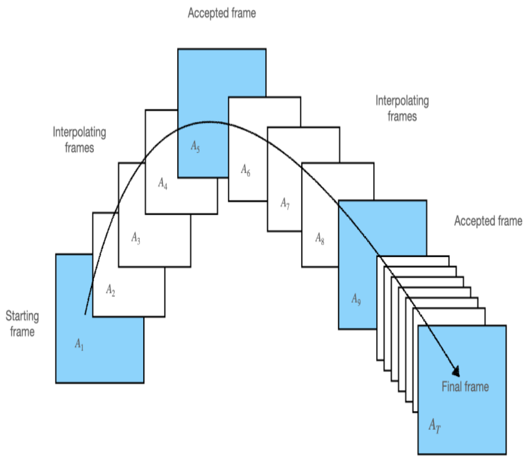

Figure 1: Each square (frame) represents the projected data with a corresponding basis. Blue frames are found by an optimisation algorithm iteratively whilst the white frames are constructed between two blue frames by geodesic interpolation.

supervised classification problems, and PDA index (Lee and Cook, 2010), which is an extension of the LDA index.

As a general visualisation method, a tour produces animations of high dimensional data via rotations between low dimension planes. Different tour types choose these planes differently, for example, a grand tour (Cook et al., 2008) selects the planes randomly to provide a general overview and a 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 planes with the aid of projection pursuit to gradually reveal the most interesting projection in the low dimension space. Given a random start, projection pursuit iteratively finds bases with higher index values and the guided tour constructs the geodesic interpolation between these planes to form a tour path. Intuitively, Figure 1 shows a sketch of the tour path where the blue frames are produced by the projection pursuit optimisation algorithm, and the white frames interpolate between them. Mathematical details of the geodesic interpolation can be found in Buja et al. (2005). The tour method has been implemented in the R package **tourr** (Wickham et al., 2011).

Optimisation in the tour

The optimisation problem in the tour context is stated as follows: Given a randomly generated starting basis $\mathbf{A}_1$, projection pursuit finds the final projection basis $\mathbf{A}_T$ that satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{X}\cdot\mathbf{A}) \tag{1}$$

$$s.t. \mathbf{A}'\mathbf{A} = I_d \tag{2}$$

where I_d is the d-dimensional identity matrix and the constraint requires the projection bases $\mathbf{A}$ to be orthogonal matrices.

Several features of this optimisation are worth noticing. First of all, this is a constrained optimisation problem as the decision variables form the entries of a projection basis, which is required to be orthonormal. It is also likely that the objective function may not be differentiable for a constructed index function and in these cases, gradient-based methods may not work well. Although finding the global maximum is the goal of an optimisation problem, it is also interesting to inspect local maximum in projection pursuit since it could present unexpected interesting projections. Lastly, there is also one computational consideration: the optimisation procedure needs to be fast to compute since the tour animation is played in real-time.

Existing algorithms

Below we introduce three line search algorithms in tour: `search_better`, `search_better_random`, and `search_geodesic`. The first two are simulated annealing algorithms whilst `search_geodesic` evaluates neighbourhood basis to determine the search direction.

Algorithm 1: random search

input : $f(.)$, α_1, $l_{\max}$, cooling
output: $\mathbf{A}_l$

generate random start $\mathbf{A}_1$ and set $\mathbf{A}_{\text{cur}} := \mathbf{A}_1$, $I_{\text{cur}} = f(\mathbf{A}_{\text{cur}})$, $j = 1$;
repeat
  set $l = 1$;
  repeat
    generate $\mathbf{A}_l = (1 - \alpha_j)\mathbf{A}_{\text{cur}} + \alpha_j \mathbf{A}_{\text{rand}}$ and orthogonalise $\mathbf{A}_l$;
    compute $I_l = f(\mathbf{A}_l)$;
    update $l = l + 1$;
  until $l > l_{\max}$ *or* $I_l > I_{cur}$;
  update $\alpha_{j+1} = \alpha_j * \text{cooling}$;
  construct the geodesic interpolation between $\mathbf{A}_{\text{cur}}$ and $\mathbf{A}_l$;
  update $\mathbf{A}_{\text{cur}} = \mathbf{A}_l$ and $j = j + 1$;
until $\mathbf{A}_l$ *is too close to* $\mathbf{A}_{cur}$ *in terms of geodesic distance*;

`search_better` is a random search device 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 α controls the radius of the sampling neighbourhood and $\mathbf{A}_{\text{rand}}$ is a randomly generated matrix with the same dimension as $\mathbf{A}_{\text{cur}}$. $\mathbf{A}_l$ is then orthogonalised to ensure the orthonormal constraint is fulfilled. When a basis is found with index value higher than the current basis $\mathbf{A}_{\text{cur}}$, the search terminates and outputs the basis for guided tour to construct an interpolation path. The next iteration of search begins after adjusting α by a cooling parameter: $\alpha_{j+1} = \alpha_j * \text{cooling}$. The termination condition is when the maximum number of iteration $l_{\max}$ is reached. The algorithm of `search_better` is summarised in Algorithm 1. A slightly different cooling scheme has been proposed by Posse (1995) to avoid the search space being reduced too fast. A halving parameter c is introduced and α is only adjusted if the last search takes more than c times to find an accepted basis.

Algorithm 2: simulated annealing

repeat
  generate $\mathbf{A}_l = (1 - \alpha_j)\mathbf{A}_{\text{cur}} + \alpha_j \mathbf{A}_{\text{rand}}$ and orthogonalise $\mathbf{A}_l$;
  compute $I_l = f(\mathbf{A}_l)$, $T(l) = \frac{T_0}{\log(l+1)}$ and $P = \min\left\{\exp\left[-\frac{I_{\text{cur}} - I_l}{T(l)}\right], 1\right\}$;
  draw U from a uniform distribution: $U \sim \text{Unif}(0, 1)$;
  update $l = l + 1$;
until $l > l_{\max}$ *or* $I_l > I_{cur}$ *or* $P > U$;

Simulated annealing (`search_better_random`) (Kirkpatrick et al., 1983; Bertsimas et al., 1993) uses the same sampling process as `search_better` but allows a probabilistic acceptance of a basis with lower index value based on the annealing $T(l)$. Given an initial T_0, 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, simulated annealing gives it a second chance to be accepted with probability

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

where $I_{(\cdot)}$ denotes the index value of a given basis. This implementation allows the algorithm to jump out of a local maximum and enables a more holistic search of the whole parameter space. This feature is particularly useful when local maxima are present. The algorithm 2 highlights how simulated annealing differs from random search in the inner loop.

Cook et al. (1995) used a line search algorithm on the space of the projection bases. In `search_geodesic`, the search direction is computed using the most prominent direction that deviating an tiny angle of δ from the current basis. The step size is chosen by optimising the index value along the geodesic direction over an 90 degree angle from $-\pi/4$ to $\pi/4$ along the search direction chosen. The optima $\mathbf{A}_{**}$ is returned for the current iteration if it meets the percentage improve condition or when $l_{\max}$ is reached. Algorithm 3 summarises the inner loop in geodesic search.

Algorithm 3: search geodesic

repeat
- generate n random directions $\mathbf{A}_{\text{rand}}$;
- compute $2n$ candidate bases deviate from $\mathbf{A}_{\text{cur}}$ by an angle of δ while ensure orthogonality;
- compute the corresponding index value for each candidate bases;
- determine the search direction as from $\mathbf{A}_{\text{cur}}$ to the candidate bases with the largest index value;
- determine the step size via optimising the index value on the search direction over a 90 degree window;
- find the optima $\mathbf{A}_{**}$ and compute $I_{**} = f(\mathbf{A}_{**})$, $p_{\text{diff}} = (I_{**} - I_{\text{cur}})/I_{**}$;
- update $l = l + 1$;

until $l > l_{\max}$ *or* $p_{diff} > 0.001$;

Visual diagnostics

To be able to make diagnostics on the optimisers, the algorithms need to populate a data structure with key elements of the algorithm. When the algorithms run, key information regarding the decision variable, objective function and hyper-parameters needs to be recorded and stored as a data object so that it is ready to be supplied to the plotting functions for diagnostics.

Data structure for diagnostics

In the optimisation algorithms for projection pursuit, the three main elements to record are 1) projection bases: $\mathbf{A}$, 2) index values: I, and 3) State: S, which labels the observation with detailed stage in the optimisation. Possible values for `search_better` and `search_better_random` include `random_search`, `new_basis`, and `interpolation`. `search_geodesic` has a wider variety that includes `new_basis`, `direction_search`, `best_direction_search`, `best_line_search`, and `interpolation`.

Multiple iterators are also needed to index the data collected at different levels: t being a unique identifier that prescribes the natural ordering of each observation while j and l being the counter of the outer and inner loop, respectively, in Algorithm 1, 2 and 3 above. Other parameters of interest recorded include $V_1 =$ method that tags the name of the optimiser, and $V_2 =$ alpha that indicates the sampling neighbourhood size. A matrix notation of the data structure is presented in Equation 3.

$$
\left[\begin{array}{c|ccc|cc|cc}
t & \mathbf{A} & I & S & j & l & V_1 & V_2 \\ \hline
1 & \mathbf{A}_1 & I_1 & S_1 & 1 & 1 & V_{11} & V_{12} \\ \hline
2 & \mathbf{A}_2 & I_2 & S_2 & 2 & 1 & V_{21} & V_{22} \\
3 & \mathbf{A}_3 & I_3 & S_3 & 2 & 2 & V_{31} & V_{32} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & l_2 & \vdots & \vdots \\ \hline
\vdots & \vdots & \vdots & \vdots & 2 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & 2 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & k_2 & \vdots & \vdots \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T & \mathbf{A}_T & I_T & S_T & J & l_J & V_{T1} & V_{T2} \\ \hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & J & k_J & \vdots & \vdots \\ \hline
\vdots & \vdots & \vdots & \vdots & J+1 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T' & \mathbf{A}_{T'} & I_{T'} & S_{T'} & J+1 & l_{J+1} & V_{T'1} & V_{T'2}
\end{array}\right]
=
\left[\begin{array}{c}
\text{column name} \\ \hline
\text{search (start basis)} \\ \hline
\text{search} \\
\text{search} \\
\vdots \\
\text{search (accepted basis)} \\ \hline
\text{interpolate} \\
\text{interpolate} \\
\vdots \\
\text{interpolate} \\ \hline
\vdots \\ \hline
\text{search} \\
\vdots \\
\text{search (final basis)} \\ \hline
\text{interpolate} \\
\vdots \\
\text{interpolate} \\ \hline
\text{search (no output)} \\
\vdots \\
\text{search (no output)}
\end{array}\right] \tag{3}
$$

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

The data structure constructed above meets the tidy data principle (Wickham et al., 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., 2020) and **ggplot2** (Wickham, 2016).

The construction of diagnostic plots adopts the core concept in ggplot2: grammar of graphics (Wickham, 2010). In grammar of graphics, plots are not produced via calling the commands, named by the appearance of the plot, i.e., boxplot and histogram, but via the concept of stacked layers. Seeing plots as stacked layers empowers us to composite diagnostic plots with an emphasis on any variable in the data object without the redundancy of creating different commands for the same type of plots that highlights on different variables.

Checking how hard the optimiser is working

A primary interest of diagnosing an optimiser is to study how it progressively finds its optimum. A simple treatment of plotting the index value across its natural order will cause the graph to be disproportional to the iteration since it usually takes much longer for an optimiser to find a better

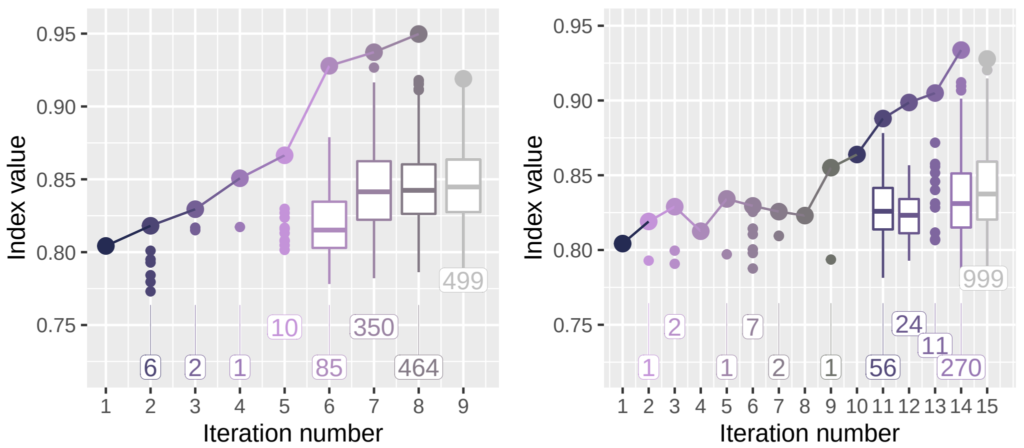

Figure 2: A comparison of search by two otimisers: `search_better` and `search_better_random` on a six-variable dataset `boa6` with holes index on a 2D problem. Both optimisers finish when the maximum number of try is reached and attain similar index value in the end. `search_better_random` takes longer to get to the final basis because it allows for probabilistic acceptance of inferior basis, which is a feature useful for more complex problem.

basis towards the end. Another option is to use summarisation in each iteration. Boxplot is a suitable candidate that can provide five points summary of each iteration and additional information can be separately added with new layers, for example, text information on the number of points searched in each iteration is added at the bottom of each iteration, the bases returned for interpolation are highlighted in larger size and linked, and information regarding the last iteration where no basis is returned is turned to grey scale. Furthermore, an option to switch from boxplot back to point geometry is helpful when the number of observation is small in one iteration and can be achieved via the `cutoff` argument.

Figure 2 shows the searches of two different optimisers: `search_better` and `search_better_random`. Both optimisers progress quickly at the first few iterations, take longer to find better basis in the later iterations, and finish off when hitting the maximum number of try, 499 and 999, respectively. Also, the target basis found in each iteration of `search_better` always has an increased index value while this is not the case for `search_better_random`. This explains why in this example, `search_better_random` takes longer to find the final basis, but in more complicated scenarios, the feature of probabilistic acceptance allows a more holistic search of the basis space and is more likely to find the global optimum.

Examining the optimisation progress

Viewing the index value of points on the interpolation path is another interest to the analysts since the projection on these bases will be played by the tour animation and it provides further information on how the index value changes when moving from one target basis to another. Figure 3 presents the index value against time on the interpolation path for the two optimisers with same configuration as 2. From Figure 3, we further know that to reach a target basis with lower index value, the optimiser plotted on the right first passes some bases with higher index value during the interpolation. If the bases on the interpolation path can be incorporated into the optimisation, we may be able to find the optimal basis quicker.

Understanding the optimiser's coverage of the search space

Apart from checking the progression of an optimiser, another interesting aspect of the diagnostics is to visualise how the visited bases looks like in its parameter space. Given the orthonormality constraint, the space of projection bases $\mathbf{A}_{p\times d}$ is a $p \times d$ dimension sphere and even with 1D projection, the basis space is high-dimensional. Dimension reduction methods, i.e. principal component analysis can be used to present the bases on 2D space with appropriate annotation to highlight the different key components of the optimisation.

In a projection pursuit guided tour optimisation, there are 7 different components that are worth-noticing in the exploration: 1) The spherical space of all the basses; 2) The starting basis of an

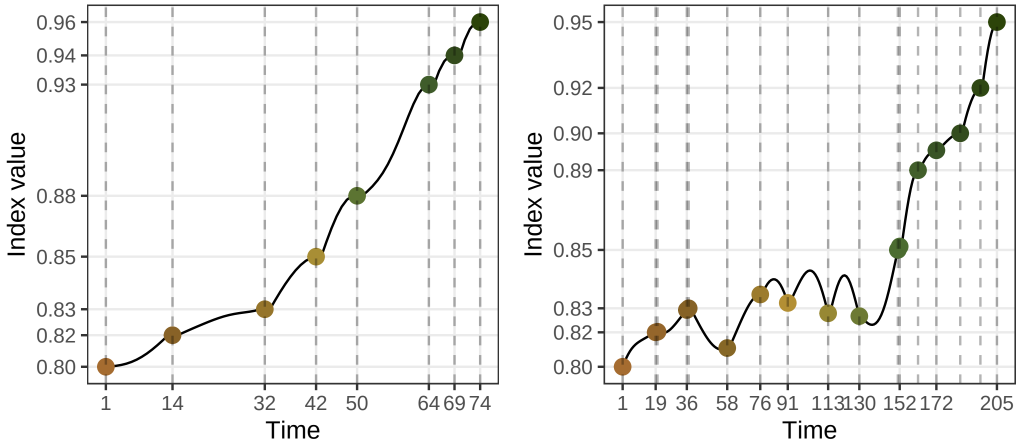

Figure 3: Trace plots of the bases on the interpolation path for two optimisers with the same configuration as the previous figure. The right path shows that to reach a lower target basis, the interpolation actually first passes some higher basis and this could potentially be information to be incorporated into the optimisers.

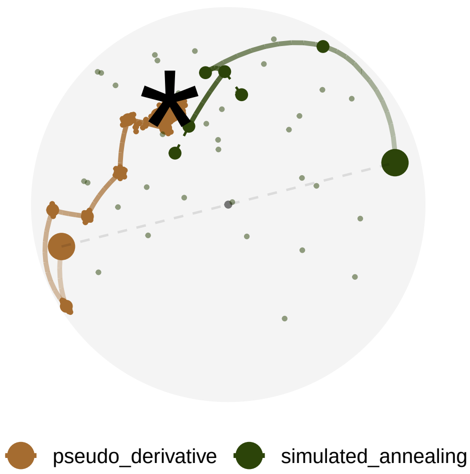

Figure 4: 1D projection on the 5-variable dataset boa5 with two optimisers: pseudo derivative and simulated annealing. All the bases in pseudo derivative has been flipped positive so that the two paths finish close to each other and to the theoretical best.

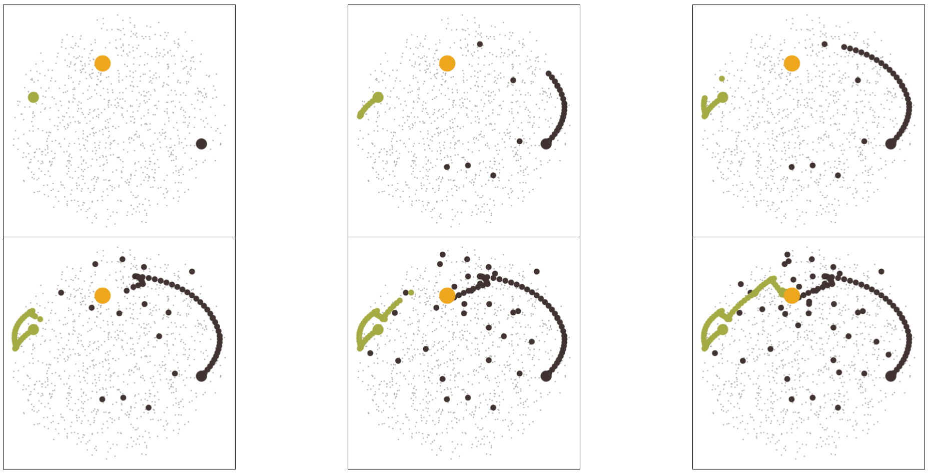

Figure 5: A selected number of frames from the animated PCA plot. With animation, it is easier to track the progression from the start to finish in each algorithm.

optimisation routine; 3) The anchor points where a new target basis is found; 4) The search basis that an optimiser has evaluated; 5) The interpolation path that constructed by the guided tour; 6) The annotation of the interrupted path and 7) The theoretical best basis, if applicable. Despite all have to be displayed on the diagnostic plot, these components differ in their importance with the most interesting ones being the starting point, anchor points and the interpolation path.

Various other components can be down-played via size, transparency, linetype and proper approximation. The basis space is generated via random bases on the original high dimensional space via the CRAN package **geozoo** (Schloerke, 2016) and projected down to 2D. Displaying the search space with hundreds of randomly generated dots can be dizzy, especially when the more important start, anchor and search bases are also displayed with point geometry. Given the fact that the basis space is a circle when projected to 2D, we can estimate the center and radius with all the bases recorded in the data object. Comparing to the start and anchor points, search points are less important as they fail to improve on the index value during the optimisation and hence an adjusted size and transparency are applied to them. A potential scenario that could happen is that two optimisers can start at the same basis but finishes with bases in opposite signs. This sign difference would not make a difference in the projected data but the PCA projection of the bases will display the finish points symmetric to each other, which can be disturbing for comparing the path close to the finish. One solution is to flip the sign of one optimiser so that they start at symmetric position but finish close to each other. To remind the analyst of this symmetry in the PCA space, a light dashed line is drawn to connect the starting points of two paths. The theoretical best basis is available if the data is simulated and can serve as a guide on how the optimisers progress to the final basis. Also, the search paths also have an increasing alpha hue from start to finish to indicate the direction of the optimsation when the theoretical guide is not available. To allow flexible annotation in different scenarios, a range of arguments are made available to adjust in the main plotting function `explore_space_pca()`.

Figure 4 shows an example of the PCA plot on the same optimisers on a 1D projection problem with 5 variables.

Animating the diagnostic plots

Animated plots can be informative in diagnostics, especially in the case of PCA plot when the starting and ending of the search is not clear. Figure 5 shows six frames of an animated version of Figure 4 and this time, it shows that `search_better` finds the optimum quicker than `search_geodesic`.

The tour looking at itself

While viewing the bases on the reduced space via PCA shed some lights on the space the optimisers have explored, the visualisation on the original $p \times d$ dimension enables a more holistic stereoscopic view of the search. To view a high dimensional ($d \geq 3$) object on a screen, an approach is to play the rotation of the object in animation and this can be done via a regular grand tour. Compared to the PCA plot, the animated rotation (tour) displayed in Figure 6 gives a more well-rounded view of the search and one can view the curved region of the tour path from different angles, which may not

Figure 6: A selected number of frames from the tour animation for viewing the 5D space of all the projection bases. The second frame on the top row views the space from a direction that is close to the one in the PCA plot. The tour animation allows for a more holistic view of the full space in high dimensions from different angles.

be presented in the PCA plot. Also the grand tour animation encompasses the PCA projection since the rotation from PCA is just one angle that maximises the variance of the bases and the grand tour produces a sequence of angles that view the search from different directions. As an evidence, the last frame in Figure 6 is a frame select from the tour animation that is close to the PCA angle and the projection looks similar to the one in Figure 4.

Diagnosing an optimiser

For a particular index function, the best algorithm to optimise relates to the character of the index and the data. If the index function is smooth and has a single maximum, all of the three algorithms introduced above can find the maximum. When multiple optima are present, `search_better` may get stuck at a local maximum and in the case where the index function is non-smooth, `search_geodesic` may even fail to find the maximum. In this section, examples will be presented to outline how the diagnostic plots can be used to compare the performance of optimisers in different scenarios.

Simulation setup

Random variables with different structures have been simulated and the distribution of each is presented in Equations 4 to 10. Variable `x1`, `x8`, `x9` and `x10` are normal distributed with zero mean and unit variance and `x2` to `x7` are mixtures of normal distributions with varied weights and locations. The mixture variables have been scaled to have an overall unit variance before running the projection pursuit.

$$x_1 \overset{d}{=} x_8 \overset{d}{=} x_9 \overset{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{4}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{5}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{7}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{8}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{9}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{10}$$

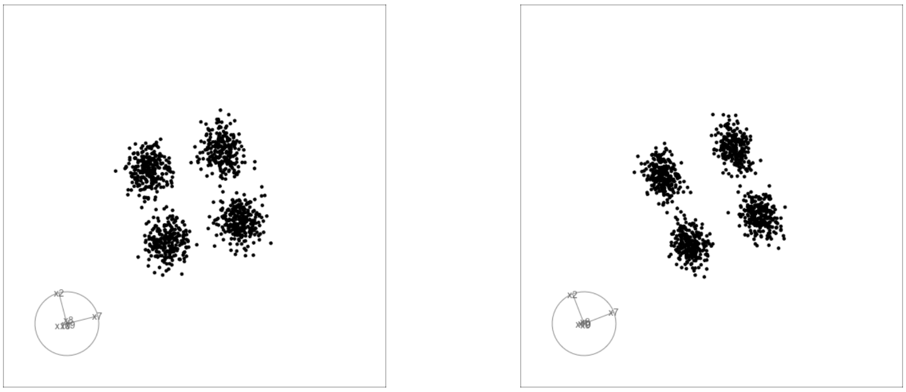

Figure 7: Two-D projection on `boa6` data with holes index optimised by `search_geodesic`. The left panel shows the final projected data before polish and the right panel shows the one after. The separation of the clusters on the y axis becomes sharper after the polish.

A problem of non-monotonicity

In section 2.4.3, an interpolation with increase-then-decrease pattern has been presented. This pattern is undesirable since the optimiser could have started the next iteration from the highest basis on the tour path, as annotated as the interpolated basis in the plot, but instead, it is forced to start from the target basis. This motivates the design of an interruption to check the index value on the tour path so that the interpolating bases is accepted only up to the one with the largest index value. After implementing this interruption, the search finds a higher final index value with fewer steps as shown in the right panel of Figure **??**.

Close but not close enough

Once the final basis has been found by an algorithm, one may want to push further to investigate whether there is an even better basis in the close neighbourhood. This motivates the polish search where the final basis is supplied as the start of a new guided tour to search for any local breakthrough.

Similar to `search_better` as a stochastic random search, `search_polish` has a different scheme of reducing the search neighbourhood. In each search-interpolation iteration, `search_better` has a fixed neighbourhood parameter alpha and this alpha is reduced by the cooling parameter only after an iteration finishes. On the contrary, `search_polish` allows alpha to be reduced during each iteration to exploit the search in the neighbourhood. Further, to avoid the case where alpha becomes too small and the further search is meaningless, three more stopping criteria have been added, on top of the original `max.tries` limit. These include:

1) the distance between the candidate basis and the current basis needs to be larger than 1e-3;
2) the percentage change of the index value need to be larger than 1e-5; and
3) the alpha parameter on itself needs to be larger than 0.01

Figure 7 presents the final projections found before and after applying `search_polish` on `search_geodesic`. Polish search improves the index value from 0.9618 to 0.9627 with reduction of weights on the non-informative variables. In terms of the projected data as in Figure 7, polish works to sharpen the edges of each cluster.

Seeing the signal in the noise

The index function, up until this point, are all smooth, while this is not the case for all the index functions. `norm_kol`, a 1D projection function based on the Kolmogorov test, compares the difference between the 1D projected data, $\mathbf{Y}_{n\times 1}$ and a randomly generated normal distribution, y_n based on the empirical cumulated distribution function (ECDF). Denote the ECDF function as $F_{.}(u)$ with the subscript indicating either the projection or the random normal variable, the `norm_kol` index is defined by

$$\max\left[F_{\mathbf{P}}(u) - F_{y}(u)\right]$$

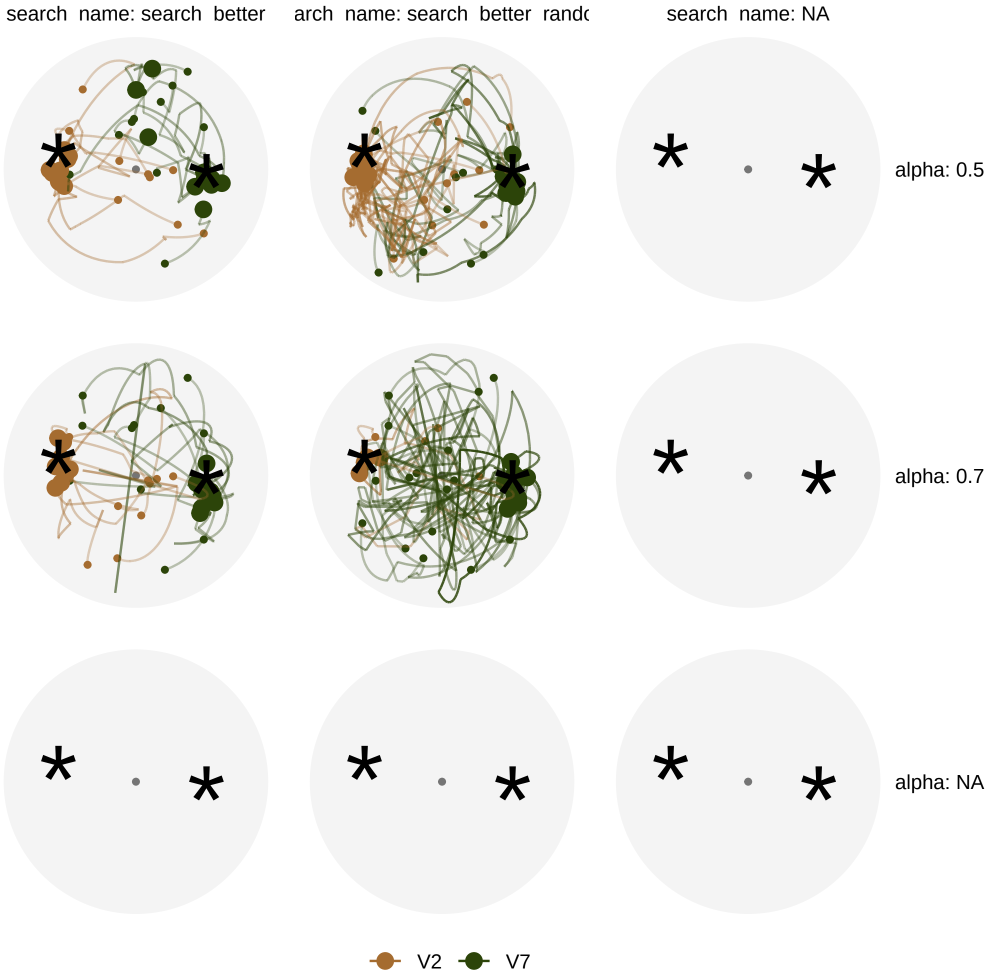

Figure 8: One-D projection of `norm_kol` index on `boa6` data optimised by `search_better`, starting from 20 different seeds.

Figure **??** compares the tracing plot of two optimisers: `search_geodesic` and `search_better`. This time, the interpolated path is no longer smooth when using either algorithm and `search_geodesic` fails to optimise this index with barely any improvement of the index value. On the other hand, `search_better` is doing relatively well. With the theoretical best basis [0, 1, 0, 0, 0] producing an index value of 0.17, `search_better` finds the final basis [0.0376, -0.9916, -0.0581, -0.0831, 0.0716] with an index value of 0.165. A further polish step will give a marginal improvement of index value to 0.175 with a basis of [0.0223, -0.9965, -0.0352, -0.0591, 0.0418]. At this stage, the difference between the theoretical best and what has been found is likely due to simulation error since the best possible basis for a simulated data will be slightly off the theoretical best basis, which is derived based on the distributional assumptions in Equations 4 to 10.

The second experiment with the noisy index is to understand how the optimisers perform when a local maximum is present. The dataset used is `boa6` where `x2` and `x7` are informative. The two theoretical best bases are [0, 1, 0, 0, 0, 0] and [0, 0, 1, 0, 0, 0] with index value 0.176 and 0.235, respectively. Hence, the global maximum happens when variable `x7` is found.

The conclusion from this experiment is that the usage of `search_better_random` and increasing the search space are methods that can avoid getting trapped in local maxima but the solution will still depend on the starting points of the simulation and the seed used.

Implementation

The implementation of this projection has been divided into two packages: the data collection object is implemented in the existing CRAN package **tourr** (Wickham et al., 2011) while the optimiser diagnostics have been implemented in a new package, **ferrn**. When a guided tour is run, the users can choose if the data from optimisation should be collected via the `verbose` argument. Once the data object has been obtained, the package, **ferrn**, can provide four diagnostic plots as shown in Section 2.4. The structure of package functionality has been listed below.

- Main plotting functions:
  - `explore_trace_search()` produces summary plots, as shown in Figure 2
  - `explore_trace_interp()`produces trace plots for the interpolation points, as shown in Figure 3
  - `explore_space_pca()` produces plots of projection basis on the reduced space by PCA, as shown in Figure 4. Animated version in Figure 5 can be turned on via the argument `animate = TRUE`
  - `explore_space_tour()` produces animated tour view on the full space of the projection bases, as shown in Figure 6.
- `get_*()` extracts and manipulates certain components from the existing data object.
  - `get_best()` extracts the best basis found in the data object
  - `get_start()` extracts the starting basis
  - `get_interp()` extracts the observations in the interpolation
  - `get_interp_last()` extracts the end observations of the interpolation in each iteration
  - `get_anchor()` extracts the target observations found by the optimiser
  - `get_search()` extracts the search observations evaluated by the optimiser
  - `get_search_count()` produces the summary table of the number of observation in each iteration
  - `get_center()` produces the center point of the basis space estimated by the starting points
  - `get_space_param()` produces the coordinates of the center and radius of the basis space
  - `get_theo()` extracts the theoretical observations from the data object
  - `get_interrupt()` extract the end point of the interpolation and the target point when an interruption happens
  - `get_basis_matrix()`: flattens all the bases into a matrix
- `bind_*()` incorporates additional information outside the tour optimisation into the data object.
  - `bind_theoretical()` incorporates the best possible basis to the existing data object with the supply of the index function and original data for producing the index value.
  - `bind_random()` generates 1000 points on the high dimensional surface of a sphere and binds it to the existing data object and output as a tibble object.
  - `bind_random_matrix()` binds the points to the basis matrix.
- Utilities
  - `add_*()` are internal wrapper functions that facilitate the composition of PCA plot
  - `theme_fern()` and `format_label()` for better display of the grid lines and axis formatting
  - `clean_method()` for clean up the name of the optimisers
  - `botanical_palettes` is a collection of color palettes from Australian native plants. Quantitative palettes include daisy, banksia and cherry and sequential palettes contain fern and acacia.
  - `botanical_pal()` as the color interpolator
  - `scale_color_botanical()` is a ggplot2 scale for botanical palettes.

Conclusion

This paper has illustrated setting up a data object that can be used for diagnosing a complex optimisation procedure. The ideas were illustrated using the optimisers available for projection pursuit guided tour. Here the constraint is the orthornormality condition of the projection bases. The approach used here could be broadly applied to understand other constrained optimisers.

Four diagnostic plots have been introduced to investigate the progression and the projection space of an optimiser. The implementation of these visualisations is designed to be easy-to-use with each plot can be produced with a simple supply of the data object. More advanced users may decide to modify on top of the basic plots or even build their own.

Most of the work in this project has been translated into code in two packages: the collection of the data object is implemented in the existing **tourr**(Wickham et al., 2011) package; manipulation and visualisation of the data object are implemented in the new **ferrn** package. Equipped with handy tools to diagnose the performance of optimisers, future work can extend the diagnostics to a wider range of index functions, i.e. scagnostics, association, and information index (Laa and Cook, 2020) and understand how the optimisers behave for index functions with different structures.

Acknowledgements

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

Bibliography

D. Asimov. The Grand Tour: A Tool for Viewing Multidimensional Data. *SIAM Journal of Scientific and Statistical Computing*, 6(1):128–143, 1985. URL `https://doi.org/10.1137/0906011`. [p2]

D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014. URL `https://doi.org/10.1016/C2013-0-10366-2`. [p2]

D. Bertsimas, J. Tsitsiklis, et al. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993. URL `https://doi.org/10.1214/ss/1177011077`. [p4]

G. E. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the royal statistical society: Series b (Methodological)*, 13(1):1–38, 1951. URL `https://doi.org/10.1111/j.2517-6161.1951.tb00067.x`. [p2]

S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. URL `https://doi.org/10.1017/CBO9780511804441`. [p2]

A. Buja, D. Cook, D. Asimov, and C. Hurley. Computational methods for high-dimensional rotations in data visualization. *Handbook of Statistics*, 24:391–413, 2005. URL `https://doi.org/10.1016/S0169-7161(04)24014-7`. [p1, 3]

D. Cook and A. Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997. URL `https://doi.org/10.2307/1390747`. [p3]

D. Cook, A. Buja, and J. Cabrera. Projection pursuit indexes based on orthonormal function expansions. *Journal of Computational and Graphical Statistics*, 2(3):225–250, 1993. URL `https://doi.org/10.2307/1390644`. [p1, 2]

D. Cook, A. Buja, J. Cabrera, and C. Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995. URL `https://doi.org/10.1080/10618600.1995.10474674`. [p4]

D. Cook, A. Buja, E.-K. Lee, and H. Wickham. Grand tours, projection pursuit guided tours, and manual controls. In *Handbook of Data Visualization*, pages 295–314. Springer, 2008. URL `https://doi.org/10.1007/978-3-540-33037-0_13`. [p1, 3]

R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013. URL `https://doi.org/10.1002/9781118723203`. [p2]

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 100(9):881–890, 1974. URL `https://doi.org/10.1109/T-C.1974.224051`. [p1, 2]

M. Fu. *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*. Springer, 2015. URL `https://doi.org/10.1007/978-1-4939-1384-8`. [p2]

P. Hall et al. On polynomial-based projection indices for exploratory projection pursuit. *The Annals of Statistics*, 17(2):589–605, 1989. URL `https://doi.org/10.1214/aos/1176347127`. [p2]

K. Healy. *Data visualization: a practical introduction*. Princeton University Press, 2018. ISBN 978-0691181622. URL `https://socviz.co/`. [p1]

S. G. Johnson. *The NLopt nonlinear-optimization package*, 2020. URL `https://github.com/stevengj/nlopt`. [p2]

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. URL `https://doi.org/10.1023/A:1008306431147`. [p1]

C. T. Kelley. *Iterative methods for optimization*. SIAM, 1999. URL `https://doi.org/10.1137/1.9781611970920`. [p1, 2]

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983. URL https://doi.org/10.1126/science.220.4598.671. [p4]

U. Laa and D. Cook. Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics. *Computational Statistics*, pages 1–35, 2020. URL https://doi.org/10.1007/s00180-020-00954-8. [p13]

E. Lee, D. Cook, S. Klinke, and T. Lumley. Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics*, 14(4):831–846, 2005. URL https://doi.org/10.1198/106186005X77702. [p2]

E.-K. Lee and D. Cook. A projection pursuit index for large p small n data. *Statistics and Computing*, 20 (3):381–392, 2010. URL https://doi.org/10.1007/s11222-009-9131-1. [p3]

M. Li, Z. Zhao, and C. Scheidegger. Visualizing neural networks with the grand tour. *Distill*, 2020. URL https://doi.org/10.23915/distill.00025. [p1]

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006. URL https://doi.org/10.1007/978-0-387-40065-5. [p2]

C. Posse. Projection pursuit exploratory data analysis. *Computational Statistics & Data Analysis*, 20(6): 669–687, 1995. URL https://doi.org/10.1016/0167-9473(95)00002-8. [p2, 4]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL https://www.R-project.org/. [p2]

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. URL https://doi.org/10.1214/aoms/1177729586. [p2]

B. Schloerke. *geozoo: Zoo of Geometric Objects*, 2016. URL https://CRAN.R-project.org/package=geozoo. R package version 0.5.1. [p9]

K. Sörensen and F. Glover. Metaheuristics. *Encyclopedia of operations research and management science*, 62: 960–970, 2013. URL https://doi.org/10.1007/978-1-4419-1153-7_1167. [p2]

S. Theussl, F. Schwendinger, and H. W. Borchers. *CRAN Task View: Optimization and Mathematical Programming*, 2020. URL https://cran.r-project.org/web/views/Optimization.html. [p2]

J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, MA, 1977. [p1]

A. Unwin. *Graphical data analysis with R*, volume 27. CRC Press, 2015. URL https://doi.org/10.1201/9781315370088. [p1]

H. Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1): 3–28, 2010. URL https://doi.org/10.1198/jcgs.2009.07098. [p6]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL https://doi.org/10.1007/978-0-387-98141-3. [p6]

H. Wickham, D. Cook, H. Hofmann, and A. Buja. tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18, 2011. URL http://doi.org/10.18637/jss.v040.i02. [p3, 12, 13]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2020. URL https://CRAN.R-project.org/package=dplyr. R package version 1.0.2. [p6]

H. Wickham et al. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. URL https://doi.org/10.18637/jss.v059.i10. [p6]

C. O. Wilke. *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media, 2019. ISBN 978-1492031086. URL https://clauswilke.com/dataviz/. [p1]

Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. Generalized Simulated Annealing for Global Optimization: The GenSA Package. *The R Journal*, 5(1):13–28, 2013. URL https://doi.org/10.32614/RJ-2013-002. [p2]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL https://yihui.name/knitr/. ISBN 978-1498716963. [p14]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. URL https://bookdown.org/yihui/rmarkdown. ISBN 978-1138359338. [p14]

H.Sherry Zhang
Monash University
Department of Econometrics and Business Statistics
huize.zhang@monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
dicook@monash.edu

Ursula Laa
University of Natural Resources and Life Sciences
Institute of Statistics
ursula.laa@boku.ac.at

Nicolas Langrené
CSIRO Data61
34 Village Street, Docklands VIC 3008 Australia
nicolas.langrene@csiro.au

Patricia Menéndez
Monash University
Department of Econometrics and Business Statistics
patricia.menendez@monash.edu