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 much work has been done in proposing indices in the literature, less has been done on evaluating the performance of the optimisers. In this paper, we implement a data collection object in the optimisation of projeciton pursuit 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 such as 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 is available for diagnosing optimisation procedures. 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 $f : \mathbb{R}^n \to \mathbb{R}$ 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: $g_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots k$ and hence the problem can be written as $\min_x f(x)$ *subject to* $g_i(x) \leq 0$. Here methods such as 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), an exploratory data analysis tool 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, characterised by the projection basis, that gives the most interesting low-dimensional view. The interestingness of the structure is defined by the index function, a function of the projection basis.

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 optima (local and global), the trade off between computational burden and proximity to the optima, 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 outlines three existing algorithms. Section 2.4 presents the new visual diagnostics. A data structure is defined to capture information during the optimisation, and different diagnostic plots are designed with the data collected. Section 2.5 shows applications of how these diagnostic plots can be used to discover interesting aspects of different optimisers and guide improvement to the existing 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

Projection pursuit guided tour combines two different methods (projection pursuit and guided tour) in exploratory data analysis. Projection pursuit, coined by Friedman and Tukey (1974), detects interesting structures (e.g. clustering, outliers and skewness) in multivariate data via low dimensions projection. Guided tour is one variation of a broader class of data visualisation methods, tour, which visualises high-dimensional data through a series of animated projection.

Notation is first established before defining projection pursuit. 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 $\mathbb{R}^d$, 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 basis Define $f : \mathbb{R}^{n \times d} \mapsto \mathbb{R}$ to be an index function that maps the projected data $\mathbf{Y}$ onto an scalar. 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 (friedman1974projection; Hall et al., 1989; Cook et al., 1993; Posse, 1995) and to classify groups (Lee et al., 2005; Lee and Cook, 2010).

As a general visualisation method, tour produces animations of high dimensional data via rotations low dimension planes. Different tour types choose these planes differently: grand tour (Cook et al.,

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.

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 the geodesic interpolation between these planes to form a tour path. Figure 1 shows a sketch of the tour path where the blue frames are produced by the projection pursuit optimisation and the white frames are interpolations between the blue frames. 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 projection pursuit is stated as follows: Given a randomly start 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 orthonormal. Several features of this optimisation are worth-noticing. 1) The optimisation is constrained and the space of the bases is a d-dimensional sphere. 2) The objective function may not be differentiable for a constructed index function. 3) 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. 4) The optimisation procedure needs to be easy to compute since the tour animation needs to be played in real-time during the optimisation.

Existing algorithms

Three optimisers have been implemented in the **tourr** (Wickham et al., 2011) package: simulated annealing (SA), simulated annealing with jump out (SAJO), and pseudo derivative (PD). Simulated annealing (SA) is a random search optimiser that samples a candidate basis $\mathbf{A}_l$ in the neighbourhood of the current basis $\mathbf{A}_{\mathrm{cur}}$ by $\mathbf{A}_l = (1-\alpha)\mathbf{A}_{\mathrm{cur}} + \alpha\mathbf{A}_{\mathrm{rand}}$ where α controls the radius of the sampling neighbourhood and $\mathbf{A}_{\mathrm{rand}}$ 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 $\mathbf{A}_{\mathrm{cur}}$, the optimiser outputs $\mathbf{A}_l$ for guided tour to construct an interpolation path. The neighbourhood parameter α is adjusted by a cooling parameter: $\alpha_{j+1} = \alpha_j * \mathrm{cooling}$ 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 SA is summarised in Algorithm 1. Posse (1995) has proposed a slightly different cooling scheme by introducing a halving parameter c. In his proposal, the α is only adjusted if the last iteration takes more than c times to find a better basis.

Simulated annealing with jump out (SAJO) (Kirkpatrick et al., 1983; Bertsimas et al., 1993) uses the

---

Algorithm 1: Simulated annealing (SA)

---

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

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

---

same sampling process as SA but allows a probabilistic acceptance of a basis with lower index value than the current. Given an initial value of 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, SAJO 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 $I_{(.)}$ denotes the index value of a given basis. This implementation allows the optimiser to make a move and explore the basis space even the landing basis does have a higher index value and hence enable the optimiser to jump out of a local optimum. The algorithm 2 highlights how SAJO differs from SA in the inner loop.

Algorithm 2: Simulated annealing with jump out (SAJO)

---

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

    Pseudo derivative (PD) search (Cook et al., 1995) uses a different strategy than SA and SAJO. Rather than randomly sample the basis space, PD first computes a search direction by evaluating vases close to the current basis. A step size is then chosen along the geodesic direction by another optimisation over an 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. Algorithm 3 summarises the inner loop of the PD.

Visual diagnostics

Before making diagnostic plots, information generated by the optimisation need to be collected and a pre-defined data structure solidifies the information being collected and streamlines the subsequent plot-making.

Data structure for diagnostics

Three main pieces of information are recorded for the projection pursuit optimisers: 1) projection bases $\mathbf{A}$, 2) index values I, and 3) state S. For SA and SAJO, possible state include `random_search`, `new_basis`, and `interpolation`. Pseudo derivative (PD) has a wider variety of state including `new_basis`,

---

Algorithm 3: Pseudo derivative (PD)

---

1 repeat

2   generate n random directions $\mathbf{A}_{\text{rand}}$;

3   compute $2n$ candidate bases deviate from $\mathbf{A}_{\text{cur}}$ by an angle of δ while ensure orthogonality;

4   compute the corresponding index value for each candidate bases;

5   determine the search direction as from $\mathbf{A}_{\text{cur}}$ to the candidate bases with the largest index value;

6   determine the step size via optimising the index value on the search direction over a 90 degree window;

7   find the optima $\mathbf{A}_{**}$ and compute $I_{**} = f(\mathbf{A}_{**})$, $p_{\text{diff}} = (I_{**} - I_{\text{cur}})/I_{**}$;

8   update $l = l + 1$;

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

`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 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 in Equation 3.

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

where $T' = T + k_J + l_{J+1}$. Note that there is no output in iteration $J + 1$ since the optimiser doesn't 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 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 of grammar of graphics (Wickham, 2010) in ggplot2. In grammar of graphics, plots are not produced by calling the commands, named by the appearance of the plot, i.e., boxplot and histogram, but by the concept of stacked layers. Seeing plots as stacked layers gives analysts the freedom to composite plots with multiple elements on hands.

Checking how hard the optimiser is working

A starting point of diagnosing an optimiser is to understand how many search an optimiser has conducted. One may want to simply plot the index value of the search points across its natural order, but a point geometry may work well if each iteration has a similar number of points. When some iterations have considerably more points than others, using a point geometry over-emphasizes the iterations that have more search points since these iterations will occupy the mast majority of the plot space. An alternative is to summarise the search in each iteration using boxplot and each iteration will
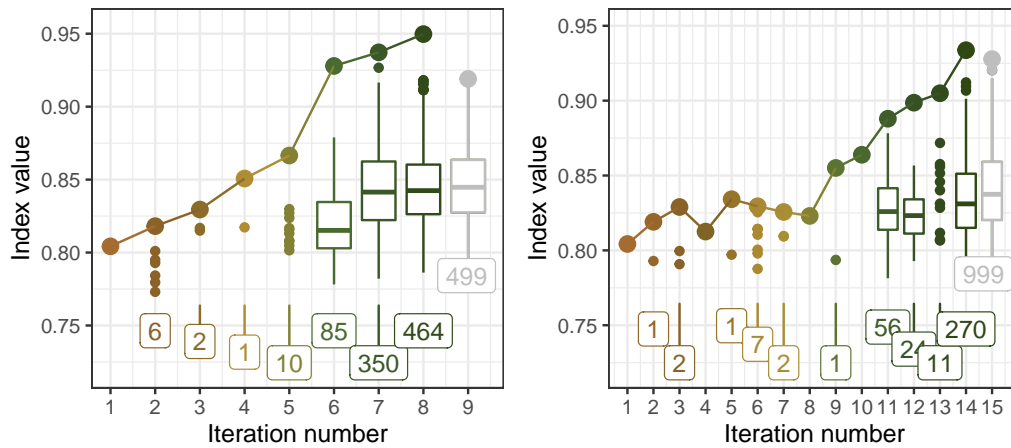
Figure 2: A comparison of search by two otimisers: SA (left) and SAJO (right) on a 2D projection of a six-variable dataset, boa6, using the holes index. Both optimisers find the final basis with similar index value while it takes SAJO more iteration because the algorithm allows a probabilistic acceptance of bases with lower index value, as observed in iteration 4 and 6-8.
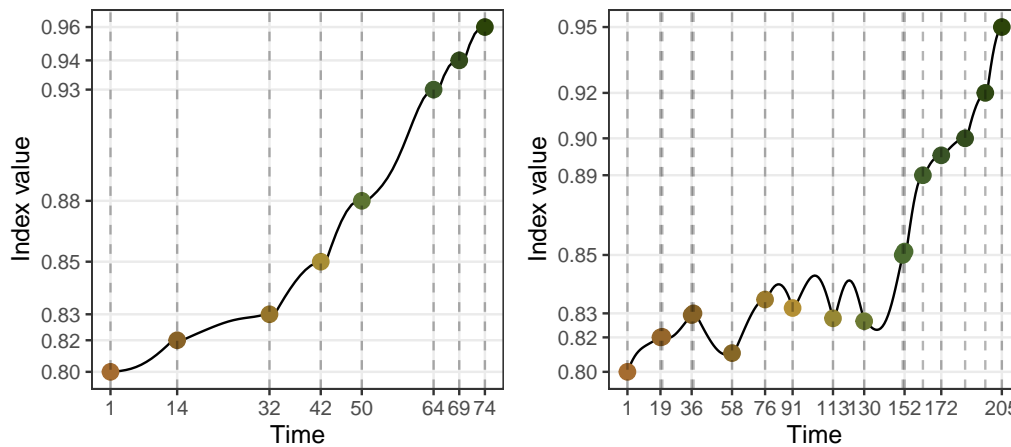
Figure 3: Trace plots of the interpolating bases with the same optimisation routine as the previous figure. Both traces are smooth while the change of index value in each iteration may not be monotinic.

then be spaced out equally. Occasionally, one may still want to switch back to a point geometry if the number of points is small in a particular iteration and this can be achieved via the `cutoff` argument in the search plot function. Additional annotations are added to facilitate better reading of the plot and these includes

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; and 3) the colour of the last iteration is in a grey scale to indicate no better basis found in this iteration.

Figure 2 shows an example of the search plot for SA (left) and SAJO (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 SA while this is not the case for SAJO. This feature gives SA an advantage in this simple example to quickly find the optimum.

Examining the optimisation progress

Another interest of diagnostic is to examine how the index value improves for interpolating bases since the projection on these bases is played by the tour animation. Trace plots are created with plotting the index value of the interpolation points against time. Figure 3 presents the trace plot of the same optimisation routine as Figure 2. The trace plot is smooth and an interesting discovery is that in SAJO,
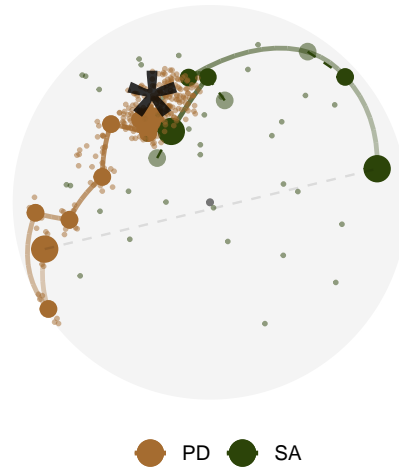
Figure 4: The PCA plot of 1D projection problem on the 5-variable dataset, boa5, using the holes index with two optimisers: SA and PD. All the bases in PD has been flipped and a grey dashed line has been annotated to indicate the symmetry of two starting bases.

the interpolation in iteration 6-8 is not monotonic. Sometimes, this non-monotonicity is acceptable since the target basis does not necessarily have a higher index value than the current one, while in some cases, the optimisation will be better off if these intermediate basis can be used as the start of the next iteration (see section 2.5.2 for a detailed discussion on this).

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 another interest. Given the orthonormality constraint, the space of projection bases $\mathbf{A}_{p \times d}$ is a $p \times d$ dimension sphere and dimension reduction methods, i.e. principal component analysis is applied to first project all the bases onto a 2D space. In a projection pursuit guided tour optimisation, there are various type of bases involved: 1) The start basis; 2) The search bases that an 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 basis differs and the most important ones are the start, interpolating and end bases. Anchor and search bases can be turned on with argument `details = TRUE`. Sometimes, two optimisers can start with the same basis but finish with bases of opposite sign. This happens because the projection is invariant to the sign difference of the bases and so does the index value, however, this creates difficulties for comparing the optimisers since the end bases will be symmetric with respect 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 understanding 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 CRAN package **geozoo** (Schloerke, 2016). Along with the bases recorded during the optimisation and a zero basis, the parameters (centre and radius) of the 2D space can be estimated. PCA is performed to get the first two PC coordinates of all the bases and the centre of the circle is the PC coordinates of the zero matrix, and radius is estimated as the largest distance from the centre to the basis. The theoretical best basis is known for simulated data and can be labelled to compare the how close the end basis found by the optimisers is to the theoretical best. Various aesthetics, i.e. size, alpha 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 is used; Alpha can also be applied on the interpolation paths to show the start to finish from transparent to opaque.

Figure 4 shows the PCA plot of SA and PD for a 1D projection problem. Both optimisers find the optimum but PD gets closer. With 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 SA evaluates points randomly in the search space to search for the next target. There are dashed lines annotated for SA and it describes the interruption of the interpolation, which will be discussed in details in Section 2.5.2.
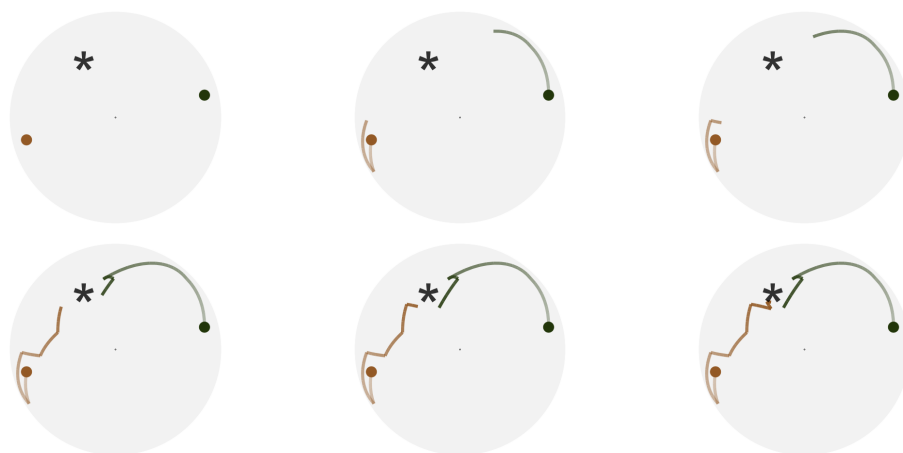
Figure 5: Six selected frames from the animated PCA plots in the previous figure. With animation, the start and end position of each optimisation is easier to identify. A full video of this aimation can be found at https://vimeo.com/user132007430/review/504242845/b73f37175a
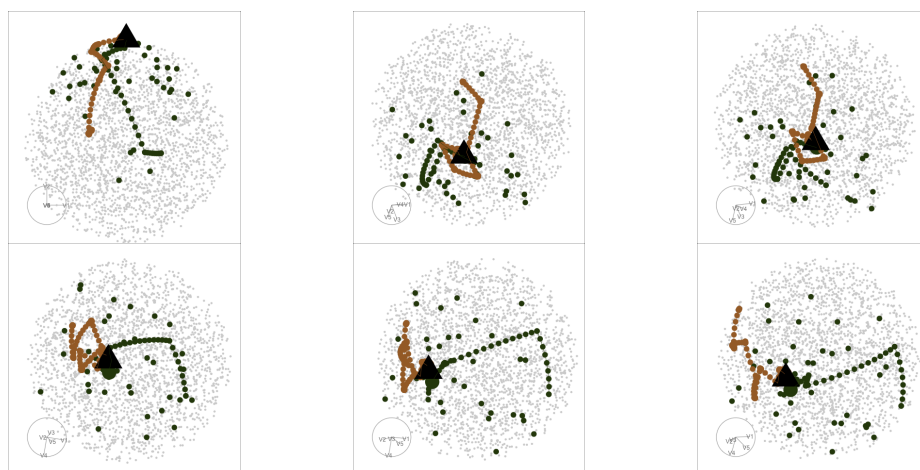
Figure 6: Six selected frames from the tour animation for viewing the same two optmisations as previous two figures. The tour animation allows for an appreciation of the search bases in the original high dimensional space. A full video of this aimation can be found at https://vimeo.com/user132007430/review/504328122/9be84db563

Animating the diagnostic plots

Animation is another display to show how the search progresses from start to finish in the space and an `animate = TRUE` argument is used to enable the animation in the PCA plot. Figure 5 shows six frames from the animation of the PCA plot in Figure 4. An additional piece of information one can learn from this animation is that SA finds its end basis quicker than PD since SA finishes its search in the 5th frame while PD is still making more progression.

The tour looking at itself

As mentioned previously, the original $p \times d$ dimension space can be simulated via randomly generated bases in **geozoo** (Schloerke, 2016) package. While the PCA plot projects the bases from the direction that maximises the variance, the tour plot display of the original high dimensional space from various directions using animation. Figure 6 shows frames from the tour plot of the same two optimisations in its original 5D space.

Forming a torus

Something to fill here …

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.

Simulation setup

Random variables with different distributions have been simulated and the distributional form of each variable is presented in Equations 4 to 10. Variable x1, x8, x9 and 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 an overall unit variance before projection pursuit.

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

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

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

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

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

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

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

A problem of non-monotonicity

An example of non-monotonic interpolation has been given in Figure 3: a path that passes bases with higher index value than the target one. For SAJO, 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 SA. The original trace plot for a 2D projection problem, optimised by SA, is shown on the left panel of Figure 7 and one can observe clearly 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 7 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 interpolations. Rather than starting the next iteration from the target basis on Time 65, SA starts the next iteration at Time 61 on the right panel and reaches a better final basis.

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 breakthrough. The polish algorithm is similar to the SA 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 been added to ensure the new basis generated is distinguishable from the current one in terms of the distance in the space, percentage change in the index value, and neighbourhood size:

1) the distance between the basis found 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 8 presents the trace plot of a 2D projection, optimised by SA and followed by the polish. THe projection by the final basis of each algorithm is also shown. The end basis found by SA reveals the four clusters in the data, but the edges of each cluster are not clean-cut. Polish works with this end
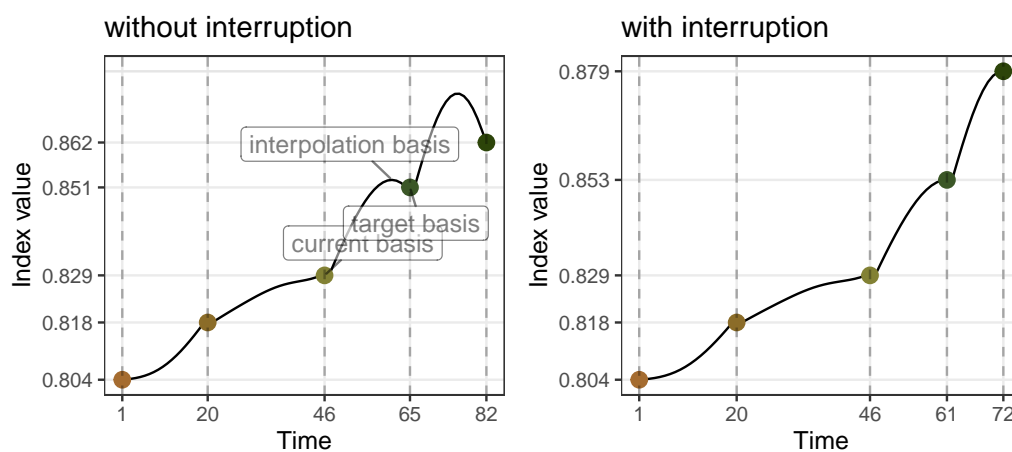
Figure 7: Trace plot of 2D projection on boa6 data with holes index, optimised by optimiser SA.
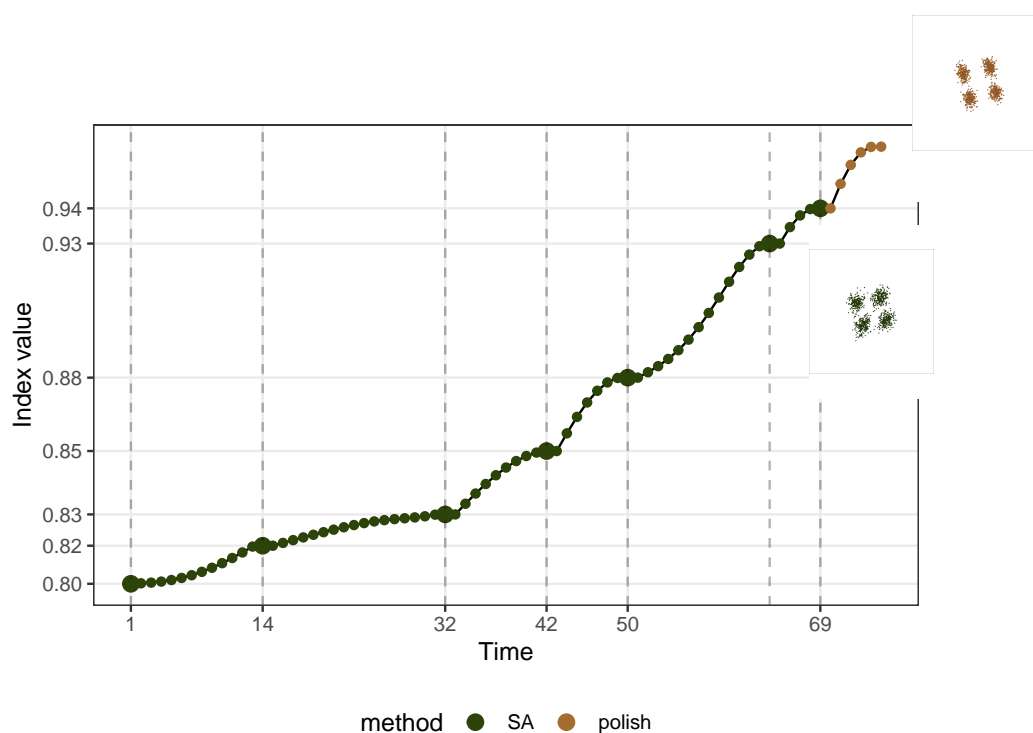
Figure 8: Trace plot of 2D projection on boa6 data with holes index, optimised first by SA and then by polish search. The polish step results in clearer cut on the edges of the four clusters in the data structure.
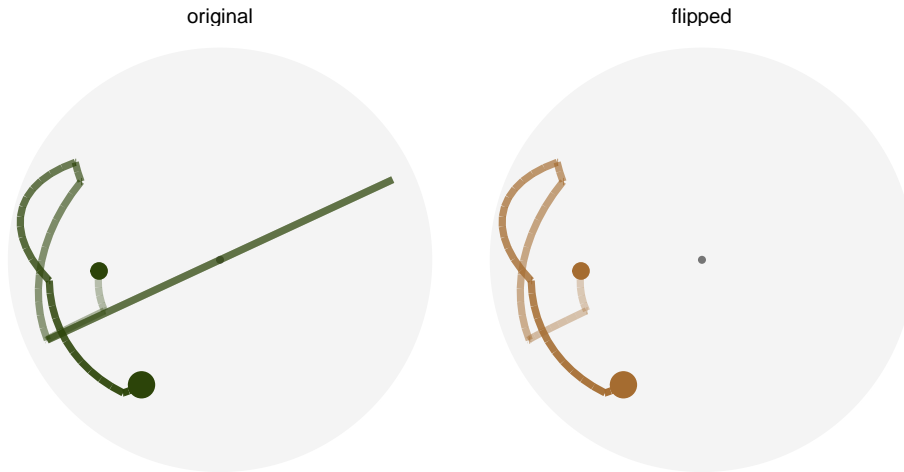
Figure 9: PCA plot of 1D projection of boa6 data, optimised by SA, with seed 2463.

basis and further pushes the index value to produce much clear edges of the cluster, especially along the vertical axis.

Reconciling the orientation

Something to fill here …

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 indices. A 1D projection function, $I^{nk}(n)$, compares the projected data, $\mathbf{Y}_{n \times 1}$, to a randomly generated normal distribution, $\mathcal{N}_{n \times 1}$, based on the Kolmogorov test. Let $F_{\cdot}(n)$ be the ECDF function, with two possible subscripts Y and $\mathcal{N}$ representing the projected and randomly generated data, and n denoting the number of observation, the Normal Kolmogorov index, $I^{nk}(n)$, is defined as:

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

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 10 presents the trace and PCA plots of all three optimisers and as expected, none of the interpolated path is smooth. There is barely any improvement made by PD, indicating its failure in optimising non-smooth index. While SA and SAJO have managed to get close to the index value of the theoretical best, trace plot shows that it takes SAJO much longer to interpolate towards the final basis. This long interpolation path is partially due to the fluctuation in the early iterations, where the SAJO 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. [A sentence for PD]. Comparing the amount of random search done by SA and SAJO, it is surprising that SAJO doesn't carry that many of sampling as SA. Combine the insight from both the trace and PCA plot, one can learn the two different search strategies by SA and SAJO: SA frequently samples in the space and only make a move when an improvement is guaranteed to be made, while SAJO 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 what SAJO has done, can lead to a more efficient search, in terms of the overall number of points evaluated.

The next experiment compares the performance of SA and SAJO 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 11 shows 80 paths simulated using 20 seeds in the PCA plot, faceted by optimiser and search size. With SA 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
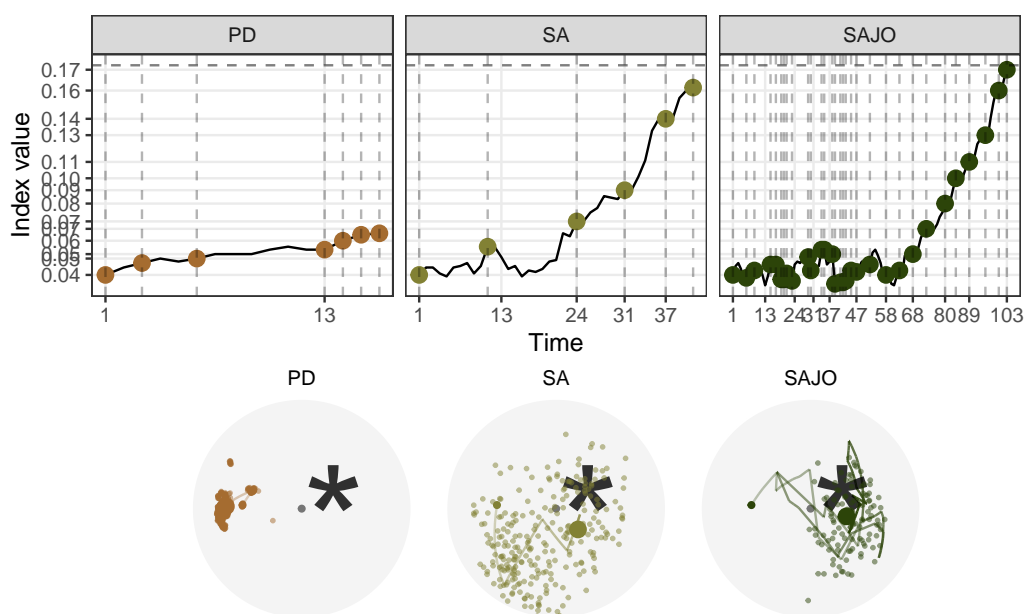
Figure 10: Trace and PCA plot of 1D projection on 5variable dataset, boa5, with `norm_kol` index, optimised by all three optimiers. Optimiser PD fails to optimise the non-smooth index while although both SA and SAJO success in optimise the index, SAJO takes much longer than SA and finishes off closer to the theoretial best.
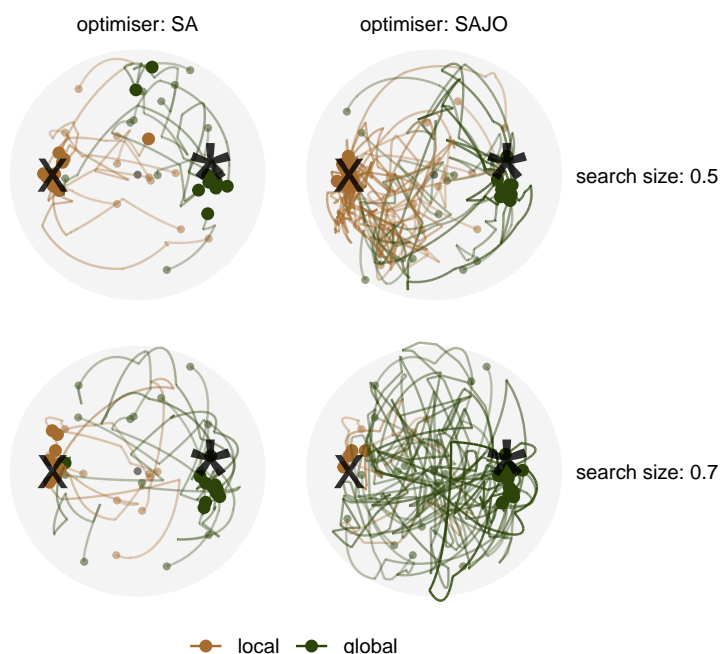
Figure 11: PCA plot of 1D projection on the 6variable data, boa6, with normal kolmogorov index. Two optimisers: SA and SAJO and two search sizes: 0.5 and 0.7 are used in the simulation for 20 replicates. Local optimum (x), is presented in this experiment, along with the global optimum (*). The final basis is larger and more opaque than the start basis for each path and color represents whether the global or local optimum is found, after polishing.

near the local nor the global optimum. With a larger search size of 0.7, more seeds have found the global maximum. Comparing SA and SAJO for a search size of 0.5, SAJO doesn't seem to improve the final basis found, despite having longer interpolation paths. However, the denser paths near the local maximum is an indicator that SAJO is working hard to examine if there is any other optimum in the basis space but the relative 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 SAJO, with a reasonable large search size, is able to overcome the local optimum and optimise close towards the global optimum.

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 to collect the data by binding animate_*() to a variable. Once the data object has been obtained, the **ferrn** package provides diagnostic plots shown in Section 2.4. The main functions of the **ferrn** package functionality is listed below.

- Main plotting functions:

  - explore_trace_search() produces summary plots in Figure 2
  - explore_trace_interp()produces trace plots for the interpolation points in Figure 3
  - explore_space_pca() produces the PCA plot of projection bases on the reduced space 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 in Figure 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 presented

- 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_*() create ggprotos for different components for the PCA plot

  - add_anchor() is a wrapper for plotting anchor bases
  - add_anno() is a wrapper for annotating the symmetry of start bases
  - add_dir_search() is a wrapper for plotting the directional search bases with magnified distance
  - add_end() is a wrapper for plotting end bases
  - add_interp() is a wrapper for plotting the interpolation path
  - add_interp_last() is a wrapper for plotting the last interpolation bases for comparing with target bases when interruption is carried
  - add_interrupt() is a wrapper for linking the last interpolation bases with target ones when interruption is carried
  - add_search() is a wrapper for plotting search bases

- `add_space()` is a wrapper for plotting the circular space
- `add_start()` is a wrapper for plotting start bases
- `add_theo()` is a wrapper 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()` for 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

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 orthonormality 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

Bibliography

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`. [p3]

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`. [p2]

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, 2]

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. [p3]

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. [p15]

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. [p2]

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, 3]

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. [p8, 9]

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, 14, 15]

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. [p15]

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. [p15]

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