Title here

Author 1 *

Department of YYY, University of XXX

and

Author 2

Department of ZZZ, University of WWW

July 29, 2020

Abstract

Friedman & Tukey commented on their intial paper on projection pursuit in 1974 that "the technique use for maximising the projection index strongly influences both the statistical and the computational aspects of the procedure." While many projection pursuit indices have been proposed in the literature, few concerns the optimisation procedure. In this paper, we developed a system of diagnostics aiming to visually learn how the optimisation procedures find its way towards the optimum. This diagnostic system can be applied to more general to help practitioner to unveil the black-box in randomised iteartive (optimisation) algorithms. An R package, ferrn, has been created to implement this diagnostic system.

Keywords: optimisation, projection pursuit, guided tourr, visual, diagnostics, R

1 Introduction

In an optimization problem the goal is to find the best solution within the space of all feasible solutions which typically is represented by a set of constraints. The problem consists on optimizing an objective function $f : S \rightarrow \Re$ with $S \in \Re^n$ in a reduced spaced given by the problem constraints to either minimize or maximize a function.... Gradient based optimization has been typically used to solve such problems. However, there are many situations where derivatives of an objective function are unavailable or unreliable and therefore traditional methods based on derivatives are not the best option to solve an optimization problem.

Derivative free methods provides another option to optimise the objective function without evaluating any gradient or Hessian information and a particular class of methods: direct search, has gained its popularity through its conceptual simplicity. However, the whole searching process in the algorithm remains a black-box. Plots are usually used to evaluate and compare the performance of different algorithms but it can easily become tedious because the code will have to be modified significantly when comparing different parameters in the algorithms. For example, a categorical variable with 5 levels can be easily mapped onto colour while mapping another categorical variable with 30 levels will not make the plot informative. Thus the plot needs to be re-designed to better suits the characteristics of the parameter (whether the parameter is a scalar or a matrix? whether the parameter is quantitative or categorical? If categorical, how many levels does the parameter have?). This motivates the design of a visual diagnostic framework for optimisation algorithms based on the idea of a *global object*.

The remainder of the paper is organised as follows. Section 2 gives a general overview of optimisation methods, specifically derivative free optimisation. Section 3 introduces the optimisation problem in a specific problem: projection pursuit guided tour and existing derivative free algorithms for solving it. Section 4 and 5 together present the new construction of a visual diagnostic framework that unveil the blackbox of optimisation. Section 4 focues on motivational ideas and data structure and Section 5 provides definitions of different diagnostic plots using grammar of graphic and examples. Finally, 6 mentions the R package: ferrn that implements the visual diagnostics described above.

2 Derivative free optimisation

Given an objective function f, one way of optimising it is to equate its gradient to zero. In modern optimisation problems, gradient information can be hard to evaluate or sometimes even impossible and Derivative-Free Optimisation (DFO) methods can be useful to approach these problems. One common class of methods in DFO is *Direct-search methods*. Coined by Hooke & Jeeves (1961), direct search methods don't require any gradient or Hessian information and has gained its popularity through its simplicity in use and reliability in solving complicated practical problems. Depends on whether a random sample is used in the search, this class of methods can be further classified as *stochastic* or *deterministic*. The stochastic version of the direct-search methods will be the main optimisation procedure analysed in this paper.

[How about adding more details into derivative free methods? ppp]

2.1 Difficulties

[Are we using projection pursuit/guided tour to better understand the convergence of optimization algorithms visually in combination with the algorithms discussed below? Or we are focusing on the optimisation problem only within the project pursuit context? Some of the problems listed below are also applicable to optimization problem in general too. ppp]

Below listed several issues in projection pursuit optimisation. Some are general optimisation problems, while others are more specific for PP optimisation.

- *Finding global maximum*: Although finding local maximum is relatively easy with developed algorithms, it is generally hard to guarantee global maximum in a problem where the objective function is complex or the number of decision variables is large. Also, there are discussions on how to avoid getting trapped in a local optimal in the literature.

- *optimising non-smooth function*: When the objective function is non-differentiable, derivative information can not be obtained, which means traditional gradient- or

Hessian- based methods are not feasible. Stochastic optimisation method could be an alternative to solve these problems.

- *computation speed*: The optimisation procedure needs to be fast to compute since tours produces real-time animation of the projected data.

- *consistency result in stochastic optimisation*: In stochastic algorithm, researchers usually set a seed to ensure the algorithm producing the same result for every run. This practice supports reproducibility, while less efforts has been made to guarantee different seeds will provide the same result.

- *high-dimensional decision variable*: In projection pursuit, the decision variable includes all the entries in the projection matrix, which is high-dimensional. Researcher would be better off if they can understand the relative position of different projection matrix in the high-dimensional space.

- *role of interpolation in PP optimisation*: An optimisation procedure usually involves iteratively finding projection bases that maximises the index function, while tour requires geodesic interpolation between these bases to produce a continuous view for the users. It would be interesting to see if the interpolated bases could, in reverse, help the optimisation reach faster convergence.

*Think about how does your package help people to understand optimisation*

- diagnostic on stochastic optim
- vis the progression of multi-parameter decision variable
- understanding learning rate - neighbourhood parameter
- understand where the local & global maximum is found - trace plot - see if noisy function

3 Projection pursuit guided tour

The optimisation problem we're interested in is in the context of projection pursuit guided tour. Coined by Friedman & Tukey (1974), projection pursuit detects interesting structures (i.e. clustering, outliers and skewness) in multivariate data via projecting it in lower dimensions. Let $\mathbf{X}_{n \times p}$ be the data matrix, an n-d projection can be seen as a linear transformation $T : \mathbb{R}^p \mapsto \mathbb{R}^d$ defined by $\mathbf{P} = \mathbf{X} \cdot \mathbf{A}$, where $\mathbf{P}_{n \times d}$ is the projected data and $\mathbf{A}_{p \times d}$ is the projection basis. Define $f : \mathbb{R}^{p \times d} \mapsto \mathbb{R}$ be an index function that maps the projection basis $\mathbf{A}$ onto a scalar index value I, this function is commonly known as the projection pursuit index (PPI) function, or the index function and is used to measure the "interestingness" of the projection. A number of index functions have been proposed in the literature to detect different data structures, including Legendre index (Friedman & 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) and PDA index (Lee & Cook 2010).

In their initial paper, Friedman & Tukey (1974) noted that "..., the technique use for maximising the projection index strongly influences both the statistical and the computational aspects of the procedure." Hence, effective optimisation algorithms are necessary for projection pursuit to find bases that give interesting projections. While we leave the formal construction the optimisation problem and existing algorithms in section 3.1, we want to emphasizes here that given a random starting (current) basis, projection pursuit repetitively searches for candidate bases nearby and outputs a target basis once it finds one with higher index value than the current basis. The target basis becomes the current basis in the next iteration and the searching continues until no better basis can be found or the termination criteria are reached.

A tour method, guided tour is usually used in conjunction with projection pursuit to interactively explore multivariate data. Tour plays an animation on the projected data which is constructed using the bases outputted by projection pursuit and their interpolations. Guided tour constructs the interpolation using the shortest geodesic path between two outputted bases and we refer readers to Buja et al. (2005) for details of the mathematical construction of the interpolation basis Figure 1 shows a sketch of the tour
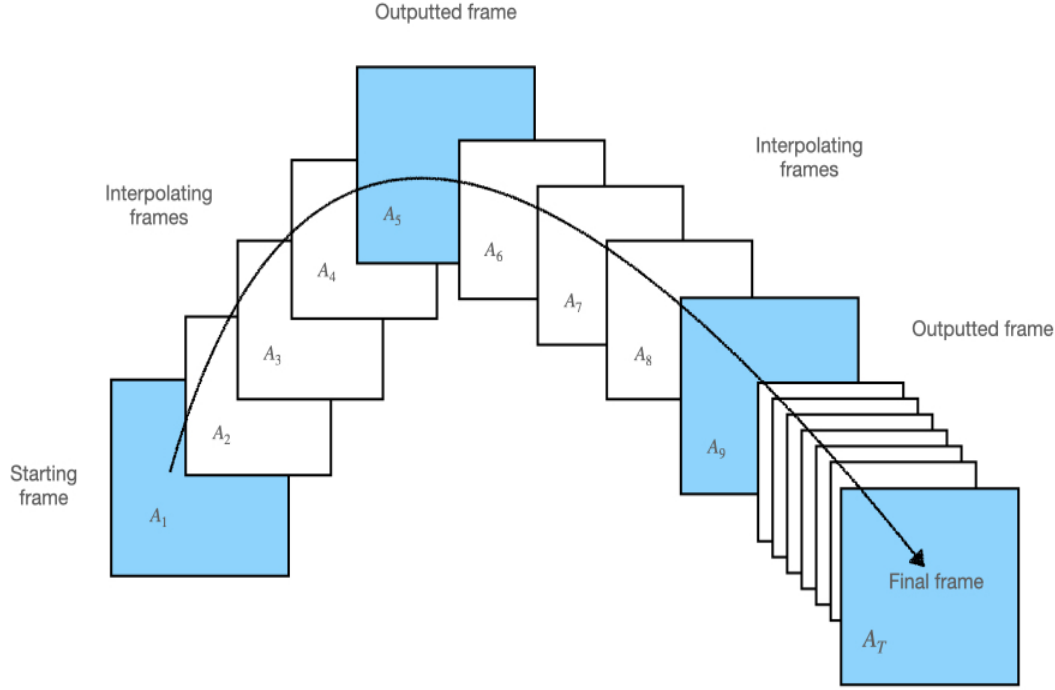
Figure 1: An illustration of the tour path

path consisting of the blue frames produced by projection pursuit and the white frames, which interpolates two blue frames. The tour method has been implemented in the *tourr* package in R, available on the Comprehensive R Archive Network at `https://cran.r-project.org/web/packages/tourr/` (Wickham et al. 2011).

3.1 Optimisation problem and existing algorithms

Now we begin to formulate the optimisation problem. Given a randomly generated starting basis $\mathbf{A}_1$, projection pursuit finds the final projection basis $\mathbf{A}_T = [\mathbf{a}_1, \cdots, \mathbf{a}_d]$, where $\mathbf{a}_i \in \mathbb{R}^p$, satisfies the following optimisation problem:

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

$$s.t.\mathcal{A} = \{\forall \mathbf{a}_i, \mathbf{a}_j \in \mathbf{A} : \|\mathbf{a}_i\| = 1 \text{ and } \mathbf{a}_i\mathbf{a}_j = 0\} \tag{2}$$

There are three existing algorithms.

Posse (1995) proposed a random search algorithm that samples a candidate basis $\mathbf{A}_l$ in the neighbourhood of the current basis $\mathbf{A}_{\text{cur}}$ by $\mathbf{A}_l = \mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{rand}$, where α controls the radius of the sampling neighbourhood and $\mathbf{A}_{rand}$ is a randomly generated matrix with the same dimension as $\mathbf{A}_{\text{cur}}$. The optimiser keeps sampling bases near the current basis until it finds one with higher index value than the current basis and then outputs it for guided tour to construct the interpolation path. A new round of search continues to find a better basis after the interpolation finishes. The halfing parameter c with default value of 30 is designed to adjust the searching neighbourhood α. When the search needs to sample more than c number of basis to find an outputted basis, the neighbourhood parameter α will be reduced by half in the next iteration. If the optimiser can't find a better basis within the maximum number of tries l_{max}, the algorithm stops. The algorithm is summarised in Algorithm 1 for one iteration.

Cook et al. (1995) explained the use of a gradient ascent optimisation with an assumption that the index function is continuous and differentiable. Since some indices could be non-differentiable, the derivative is replaced by a pseudo-derivative of evaluating five randomly generated directions in a tiny nearby neighbourhood. Once the best direction is determined, the optimiser finds the best projection $\mathbf{A}_{**}$ on the geodesic between the current basis $\mathbf{A}_{\text{cur}}$ and the best directional basis $\mathbf{A}_*$. If the percentage change in the index value between $\mathbf{A}_{**}$ and $\mathbf{A}_{\text{cur}}$ is greater than a threshold value, $\mathbf{A}_{**}$ is outputted for the current iteration to construct the interpolation or the algorithm repeats the above steps until l_{max} is reached and the searching terminates. Algorithm 2 summarise the steps in geodesic search.

Simulated annealing (Bertsimas et al. 1993, Kirkpatrick et al. (1983)) is a non-derivative procedure based on a non-increasing cooling scheme $T(l)$. Given an initial T_0, the temperature at iteration l is defined as $T(l) = \frac{T_0}{\log(l+1)}$. The simulated annealing algorithm works as follows. Given a neighbourhood parameter α and a randomly generated orthonormal

---
Algorithm 1: random search

    input : The current projection basis: $\mathbf{A}_{\mathrm{cur}}$; The index function: f

    output: The target basis: $\mathbf{A}_l$

1 initialisation;

2 Set $l = 1$ and $c = 0$;

3 **while** $l < l_{\max}$ **do**

4   Generate $\mathbf{A}_l = \mathbf{A}_{\mathrm{cur}} + \alpha \mathbf{A}_{\mathrm{rand}}$ ensuring $\mathbf{A}_l$ is orthonormal;

5   Compute $I_l = f(\mathbf{A}_l)$;

6   **if** $I_l > I_{cur}$ **then**

7   **return** $\mathbf{A}_l$;

8   **else**

9 $c = c + 1$;

10   **end**

11 $l = l + 1$;

12 **end**

---

basis A_{rand}, a candidate basis is sampled by $\mathbf{A}_{t+1} = (1 - \alpha)\mathbf{A}_t + \alpha\mathbf{A}_{rand}$. If the index value of the candidate basis is larger than the one of the current basis, the candidate basis is outputted. While if it is smaller, the candidate has another chance of being accepted with probability $P = \min\left\{\exp\left[\frac{I_{cur} - I_l}{T(l)}\right], 1\right\}$ where $I.$ denotes the index value of a given basis. The algorithm can be summarised as in Algorithm 3.

Notice that the three algorithms above constitute one search. The outputted basis is then sent to construct the interpolation and one such search-and-interpolate step is one round.

Algorithm 2: search geodesic

---

input : The current projection basis: $\mathbf{A}_{\mathrm{cur}}$; The index function: f

output: The target basis: $\mathbf{A}_{**}$

1 initialisation;

2 Set $l = 1$;

3 **while** $l < l_{\mathrm{max}}$ **do**

4   Generate ten bases in five random directions: $\mathbf{A}_l : \mathbf{A}_{l+9}$ within a small neighbourhood dist;

5   Find the direction with the largest index value: $\mathbf{A}_*$ where $l < * < l + 9$;

6   Construct the geodesic $\mathcal{G}$ from $\mathbf{A}_{\mathrm{cur}}$ to $\mathbf{A}_*$;

7   Find $\mathbf{A}_{**}$ on the geodesic $\mathcal{G}$ that has the largest index value ;

8   Compute $I_{**} = f(\mathbf{A}_{**})$, $p_{\mathrm{diff}} = (I_{**} - I_{\mathrm{cur}})/I_{**}$;

9   **if** $p_{diff} > 0.001$ **then**

10   **return** $\mathbf{A}_{**}$;

11   **end**

12 $l = l + 1$;

13 **end**

Algorithm 3: simulated annealing

input : The current projection basis: $\mathbf{A}_{\text{cur}}$; The index function: f; Initial termperature: T_0

output: The global object; The target basis: $\mathbf{A}_l$

1 initialisation;

2 Set $l = 1$;

3 **while** $l < l_{\max}$ **do**

4 $\quad$ Generate $\mathbf{A}_l = (1 - \alpha)\mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{rand}$ ensuring $\mathbf{A}_l$ is orthonormal ;

5 $\quad$ Compute $I_l = f(\mathbf{A}_l)$ and $T(l) = \frac{T_0}{\log(l+1)}$;

6 $\quad$ **if** $I_l > I_{cur}$ **then**

7 $\quad\quad$ | **return** $\mathbf{A}_l$;

8 $\quad$ **else**

9 $\quad\quad$ Compute $P = \min\left\{\exp\left[\frac{I_{\text{cur}} - I_l}{T(l)}\right], 1\right\}$;

10 $\quad\quad$ Draw D from a uniform distribution: $D \sim \text{Unif}(0, 1)$;

11 $\quad\quad$ **if** $P > D$ **then**

12 $\quad\quad\quad$ | **return** $\mathbf{A}_l$;

13 $\quad\quad$ **end**

14 $\quad$ **end**

15 $\quad$ $l = l + 1$;

16 **end**

---

4   Visual diagnostic system

4.1   Motivation

We can see from Figure 1 that only the data projected using the outputted bases and inter-polated bases are animated by tour; none of the searching bases are presented in teh tour animation. The optimisation algorithms in projection pursuit, although has been demonstrated to find the global maximum in various literature we referred above, focuses solely on finding the global maximum and remains as black-box machanisms. There has been fewer tools available to explore how each algorithm iteratively finds its maximum and compare how different algorithms searches the parameter space. Visualisation has been widely used for exploring and understanding data. Presenting information in a graphical manner often allows people to see information they would otherwise not seen. This motivates us **to develop a visual framework to diagnose optimisation algorithms in projection pursuit with a aim to understand and compare different existing algorithms.** This study is meaningful because **as more complex index functions are proposed, only an appropriate optimisation algorithm allows us to find the projection basis that presents interesting underlying structure in the data**.

The idea of generalised framwork for diagnostic plots is inspired by the concept of grammar of graphic (Wickham 2010), which powers the primary graphical system in R, ggplot2 (Wickham 2016). In grammar of graphic, plots are not defined by its appearance (i.e. boxplot, histogram, scatter plot etc) but by "stacked layers". Using this design, ggplot doesn't have to develop a gazillion of functions that each produces a different type of plot from a different data structure. Instead, it aesthetically maps variables (and its statistical transformation) in a dataset to different geometric objects (points, lines, box-and-whisker etc) and builds the plot through overlaying different layers. There are different ways to represent the same data in rectangular form. Certain cleaning steps are needed, before using ggplot2, to bring the data into a tidy data format (Wickham et al. 2014) where 1) each observation forms a row; 2) each variable forms a column and 3) each type of observational unit forms a table. In a tidy format, data wrangling and visualisation are greatly simplified.

Hence the study **contributes to set up a tidy data structure for collecting observations from optimisation algorithms and prescribe a set of rules for generalising diagnostic plots for optimisation algorithms, specifically those used for projection pursuit.**

4.2 Global object

For our visual diagnostics, global object is a such data construction with key components in optimisation algorithms and is designed for making diagnostic plots easier. In the optimisation algorithms for projection pursuit, two key elements are the projection basis: $\mathbf{A}$ and the index value: I. State: S labels the observation with detailed stage of searching or interpolation.

Multiple iterators are needed to index the data collected at different levels. t is a unique identifier that prescribes the natural ordering of each observation; j is the counter for each search-and-interpolate round, which remains the same for within one round and has an increment of one once a new round starts. l is the counter for each search/interpolation allowing us to know how many basis has the algorithm searched before finding one to output.

There are other parameters that are of our interest and we denote them as V_p. In projection pursuit, this includes $V_1 =$ method, which tags the name of the algorithm used and $V_2 =$ alpha, which is the neighbourhood parameter that controls the size in sampling candidate bases. The data structure can thus be shown as in Equation 3.

$$
\begin{array}{c|ccc|cc|cc}
t & \mathbf{A} & I & S & j & l & V_1 & V_2 \\
\hline
1 & \mathbf{A}_1 & I_1 & S_1 & 1 & 1 & V_{11} & V_{21} \\
2 & \mathbf{A}_2 & I_2 & S_2 & 2 & 1 & V_{12} & V_{22} \\
3 & \mathbf{A}_3 & I_3 & S_3 & 2 & 2 & V_{13} & V_{23} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & l_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & 2 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & 2 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & k_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T & \mathbf{A}_T & I_T & S_T & J & l_J & V_{1T} & V_{2T} \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & J & k_J & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J+1 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T' & \mathbf{A}_{T'} & I_{T'} & S_{T'} & J+1 & l_{J+1} & V_{1T'} & V_{2T'}
\end{array}
=
\begin{array}{c}
\text{column name} \\
\hline
\text{search (start basis)} \\
\text{search} \\
\text{search} \\
\vdots \\
\text{search (outputed basis)} \\
\hline
\text{interpolate} \\
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\vdots \\
\text{search} \\
\vdots \\
\text{search (final basis)} \\
\hline
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\text{search (no output)} \\
\vdots \\
\text{search (no output)}
\end{array}
\tag{3}
$$

where $T' = T + k_J + l_{J+1}$. Note that we deliberately denote the last round of search as $j = J + 1$ and in that round there is no output/interpolation basis and the algorithm terminates. This notation allows us to denote the last complete search-and-interpolate round as round J and hence the final basis is A_T and highest index value found is I_T.

[outside the paper: I find the notation of current/target basis is confusing because the target basis in round j becomes the current basis in round $j + 1$. Also, when we start to have polish, the target basis may not be the current basis in the next round... The place where current/target is most appropriate is probably when describing the interpolation where the first one is always the current basis and the last is always the target basis. I
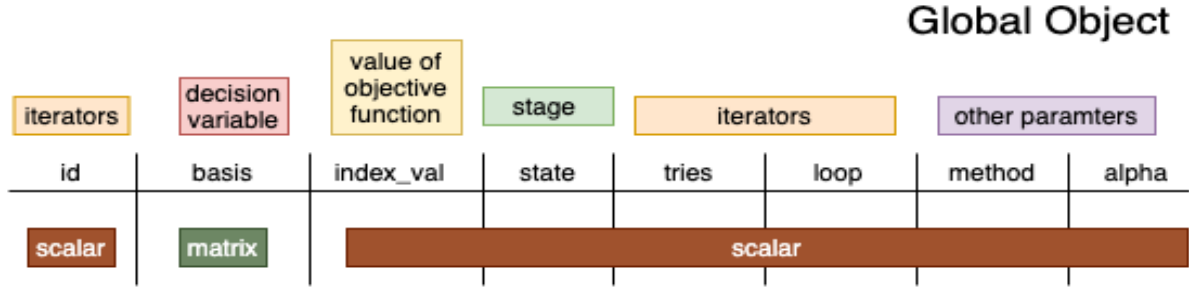
13

Figure 2: The global object in projection pursuit guided tour.

think it is better to leave this language in the code]

A sketch of the global object for projection pursuit guided tour is presented in Figure 2.

[I feel this sketch was initially useful but now since we have the data matrix and the printed output of the global object, it doesn't any additional information. I'm still keeping it here but we may need to remove it if it's not useful :(]

4.3   Simulated data

We simulate some random variables of size 1000 with different structures. `x1`, `x8`, `x9` and `x10` are simulated from normal distribution with zero mean and variance of one as in equation 4. When using projection pursuit to explore the data structure based on its departure from normality, the entry in the projection basis for these variables should be close to zero in theory. `x2` to `x7` are mixture of normal distributions with different weights and locations. Equation 5 to 10 outlines the distribution where each variable is simulated from and Figure 3 shows the histogram of each variable except `x3`. All the variables are then scaled to have variance of one before running the projection pursuit.
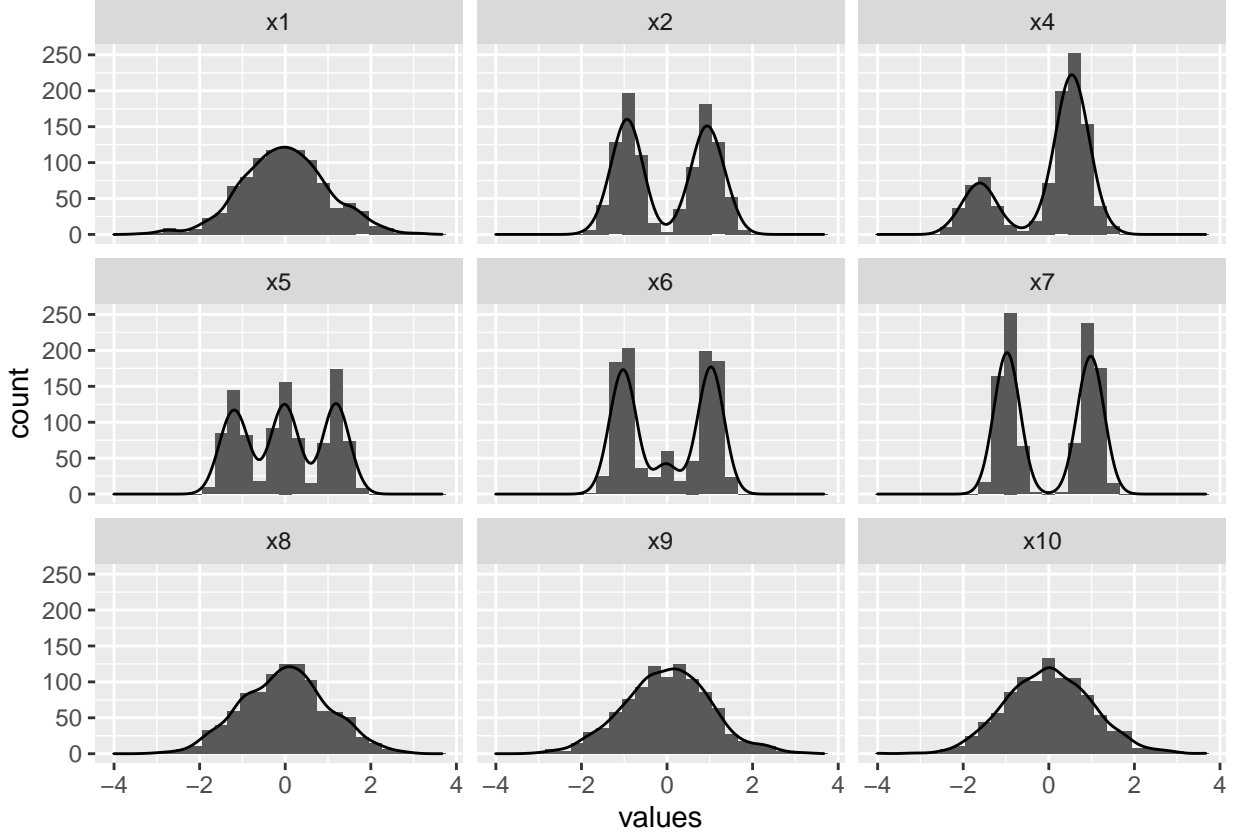
Figure 3: The distribution of simulated data except x3

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

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

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

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

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

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

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

We form our first dataset using variable `x1`, `x2`, `x8`, `x9` and `x10` and run the guided

tour with optimiser `search_better`. When the optimisation ends, the global object will be stored and printed (it can be turned off by supplying argument `print = FALSE`). Additional messages during the optimisation can be displayed by argument `verbose = TRUE`. Below shows the first ten rows of the global object. Notice that the tibble object allows the list-column `basis` to be printed out nicely with the dimension of the projection basis readily available.

[I notice that in the implementation the loop index for interpolation starts from 0 rather than 1. Need more investigation to see if it is easy to change it to 1 or we will have to change the definition of the data structure.]

```
## # A tibble: 10 x 8
##       id basis              index_val info          tries  loop method       alpha
##    <int> <list>                 <dbl> <chr>         <dbl> <dbl> <chr>         <dbl>
## 1      1 <dbl[,1] [5 x 1]>      0.749 new_basis         1     1 <NA>            0.5
## 2      2 <dbl[,1] [5 x 1]>      0.730 random_sear~      2     1 search_bett~    0.5
## 3      3 <dbl[,1] [5 x 1]>      0.743 random_sear~      2     2 search_bett~    0.5
## 4      4 <dbl[,1] [5 x 1]>      0.736 random_sear~      2     3 search_bett~    0.5
## 5      5 <dbl[,1] [5 x 1]>      0.747 random_sear~      2     4 search_bett~    0.5
## 6      6 <dbl[,1] [5 x 1]>      0.725 random_sear~      2     5 search_bett~    0.5
## 7      7 <dbl[,1] [5 x 1]>      0.752 new_basis         2     6 search_bett~    0.5
## 8      8 <dbl[,1] [5 x 1]>      0.749 interpolati~      2     0 search_bett~     NA
## 9      9 <dbl[,1] [5 x 1]>      0.750 interpolati~      2     1 search_bett~     NA
## 10    10 <dbl[,1] [5 x 1]>      0.750 interpolati~      2     2 search_bett~     NA
```

5   Visual diagnostic plots

Below we will present several examples of diagnosing different aspects of the projection pursuit optimisation. We will present:

1) static plots to explore the index value,

2) animated plots to explore the projection basis and,

3) a self-contained example to optimise a complex index function.

In the first two sections, we will first provide a toy example that is easy to grasp and then more examples that can help us to understand the algorithm and parameter choice. Remember the research question we raised earlier, the purpose of visual diagnostics is to understand:

- Whether the algorithm has successfully found the maximum and how the index value changes throughout the algorithm?

- How does the searching space look like, that is, geometrically, where are the projection bases located in the space?

The first question can be answered using a static plot with x-axis showing the progression of the optimisation and y-axis showing the value of the objective function. The second question can be addressed via visualising the rotating high dimensional space or its projection on the reduced 2D space. Thus animated visualisation is needed to perceive the optimisation path in the searching space.

Since the global object is already tidy, not further tidying steps is needed, while certain data wrangling steps (Wickham & Grolemund 2016) are still needed to transform the global object into desirable format for one particular visualisation. To emphasize on this good practice of data analysis, we will describe the transformation steps needed for each diagnostic plots before stepping into visualisation.

5.1   Explore the value of objective function

5.1.1   Searching points

A primary interest of diagnosing an optimisation algorithm is to study how it finds its optimal progressively. We could plot the index value across its natural ordering, however, different iterations may have different number of points and, towards the end of the search there could easily be hundreds of bases being tested before the target basis is found. In the plot, points from those iterations towards the end will occupy the vast majority of the plot space. This motivates to use summarisation. Rather than knowing the index value of *every* basis, we are more interested to have a general summary of all the index value in that iteration and more importantly, the basis with the largest index value (since it prescribes the next geodesic interpolation and future searches).

Boxplot is a suitable candidate that provides five points summary of the data, however it has one drawback: it doesn't report the number of point in each box. We may risk losing information on how many points it takes to find the target basis by displaying the boxplot alone for all `tries`. Thus, the number of point in each iteration is displayed at the bottom of each box and we provide options to switch iteration with small number of points to a

point geometry, which is achieved via an `cutoff` argument. A line geometry is also added to link the points with the largest index value in each iteration. This helps to visualise the improvement made in each iteration. Using the concept of *gramma of graphics* (Wickham 2010), the plot for exploring index value can be defined in three layers as:

- Layer 1: boxplot geom

  - data: group by j and filter the observations in the group that have count greater than `cutoff = 15`.
  - x: j is mapped to the x-axis
  - y: the statistical transformed index value: $Q_q(I'_t)$ is mapped to the y-axis where $Q_q(x)$ finds the $0, 25, 50, 75, 100$th-quantile for the vector x and I'_t is the index value of all the searching bases defined in Matrix 3.

- Layer 2: point geom

  - data: group by j and filter the observations in the group that have count less than `cutoff = 15`.
  - x: j is mapped to the x-axis
  - y: I is mapped to the y-axis

- Layer 3: line geom

  - data: filter the points with the highest index value in group j
  - x: j is mapped to the x-axis
  - y: I is mapped to the y-axis

Toy example: exploring searching points   We choose variable `x1`, `x2`, `x3`, `x8`, `x9` and `x10` to perform a 2D projection with tour. Parameter `search_f = tour::search_better` and `max.tries = 500` is used. The index value of the searching points are shown in Figure 4. Label at the bottom indicates the number of observations in each iteration and facilitates the choice of `cutoff` argument (by default `cutoff = 15`). We learn that the `search_better` quickly finds better projection basis with higher index value at first and the takes longer to find a better one later.
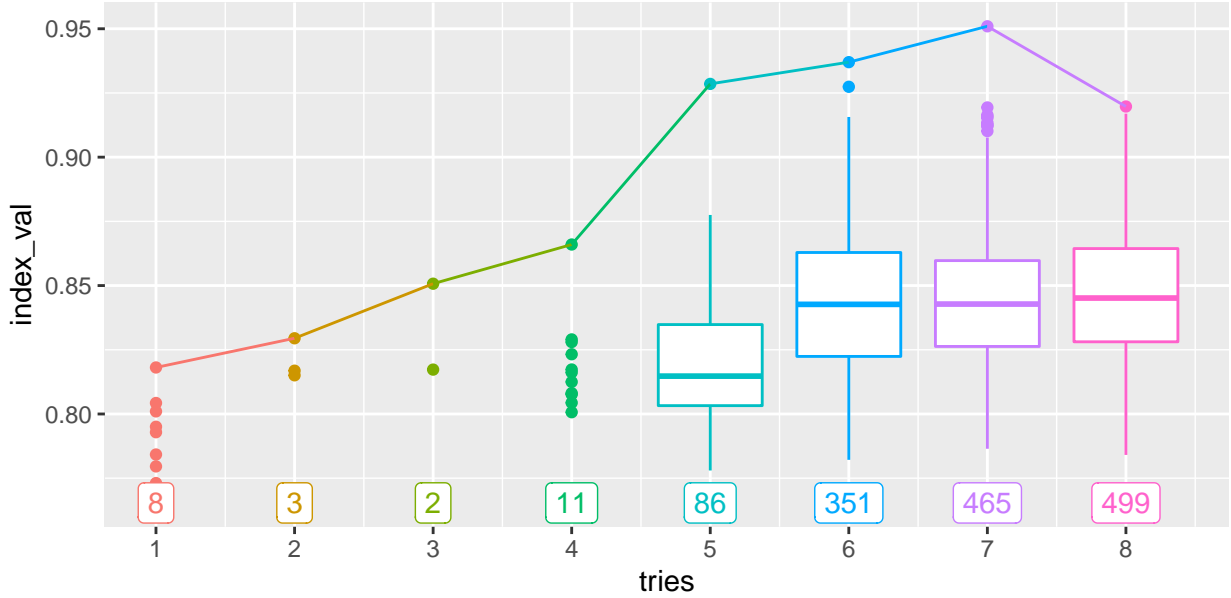
Figure 4: A comparison of plotting the same search points with different plot designs. The left plot doesn't efficiently use the plot space to convey information from the plot while the right plot provides good summarisation of data and number of points in each tries.

5.1.2 Interpolating points

Sometimes, rather than explore the searching points, we may be interested in explore the points on the interpolation path (target and interpolating points) since these points will be played by the tour animation. Since interpolation paths are geodesically the shortest, a summarisation using boxplot geometry is no longer needed. The slightly modified plot definition is shown below:

- Layer 1: point geom
    - data: filter the observations with $S =$ interpolation and mutate t to be the row number of the subsetted tibble
    - x: t is mapped to the x-axis
    - y: I is mapped to the y-axis

- Layer 2: line geom
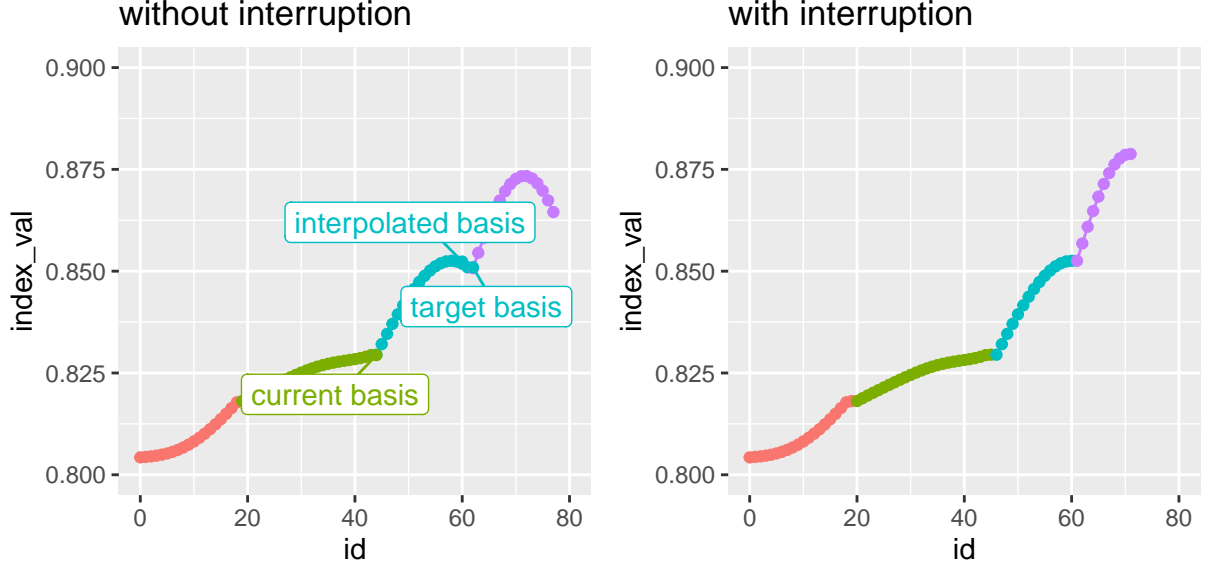   - using line geometry for the same data and aesthetics

Figure 5: Trace plots of the interpolated basis with and without the interruption. The interruption stops the interpolation when the index value starts to decrease at id = 60. The implementation of the interuption finds an ending basis with higher index value using fewer steps.

A more complex example: Interruption We use the same dataset as the toy example above to explore the search function `search_better` and we want to learn how the index value changes on the interpolation path for `holes` index. From the left panel of Figure 5, we observe that when interpolating from the current basis to the target basis, the index value may not be monotone: we could reach a basis with higher index value than the target basis on the interpolation path. In this sense, we would be better off using the basis with the highest index value on the interpolation path as the current basis for the next iteration (rather than using the target basis). Hence, an interruption is constructed to accept the interpolating bases only up to the one with the largest index value. After implementing this interruption, the search finds higher final index value with fewer steps as shown in the right panel of Figure 5.

5.1.3 Polishing points

In principle, all the optimisation routines should result in the same output for the same problem while this may not be the case in real application. This motivates the creation of a polishing search that polishes the final basis found and achieves unity across different methods.

search_polish takes the final basis of a given search as a start and uses a brutal-force approach to sample a large number of basis (n_sample) in the neighbourhood. Among those sampled basis, the one with the largest index value is chosen to be compared with the current basis. If its index value is larger than that of the current basis, it becomes the current basis in the next iteration. If no basis is found to have larger index value, the searching neighbourhood will shrink and the search continues. The polishing search ends when one of the four stopping criteria is satisfied:

1) the chosen basis can't be too close to the current basis
2) the percentage improvement of the index value can't be too small
3) the searching neighbourhood can't be too small
4) the number of iteration can't exceed the max.tries

The usage of search_polish is as follows. After the first search, the final basis from the interpolation is extracted and supplied to the second search as the start argument. search_polish is used as the search function. All the other arguments should remain the same.

```
set.seed(123456)
holes_2d_geo <- animate_xy(data_mult[,c(1,2, 7:10)],tour_path =
                          guided_tour(holes(), d = 2,
                                      search_f = tourr:::search_geodesic),
                          rescale = FALSE, verbose = TRUE)


last_basis <- holes_2d_geo %>% filter(info == "interpolation") %>%
  tail(1) %>% pull(basis) %>% .[[1]]
```

```
set.seed(123456)
holes_2d_geo_polish <- animate_xy(data_mult[,c(1,2, 7:10)], tour_path =
                                     guided_tour(holes(), d = 2,
                                       search_f = tourr:::search_polish),
                                 rescale = FALSE, verbose = TRUE,
                                 start = last_basis)
```

Slight variation of plot definition due to the addition of polishing points is as follows:

- Layer 1: point geom

    - data: filter the observations with $S =$ interpolation; bind the global object from optimisation and interpolation and form polishing; mutate t to be the row number of the binded tibble.
    - x: t is mapped to the x-axis
    - y: I is mapped to the y-axis
    - colour: V is mapped to the colour aesthetic

- Layer 2: line geom

Another example: Polish   Again using the same data, we are interested to compare the effect of different `max.tries` in the 2D projection setting. `max.tries` is a hyperparameter that controls the maximum number of try before the search ends. The default value of 25 is suitable for 1D projection while we suspect it may not be a good option for the 2D case and hence want to compare it with an alternative, 500. As shown in Figure 6, both trials attain the same index value after polishing while the small `max.tries` of 25 is not sufficient for search_better to find its global maximum and we will need to adjust the `max.tries` argument for the search to sufficiently explore the parameter space.
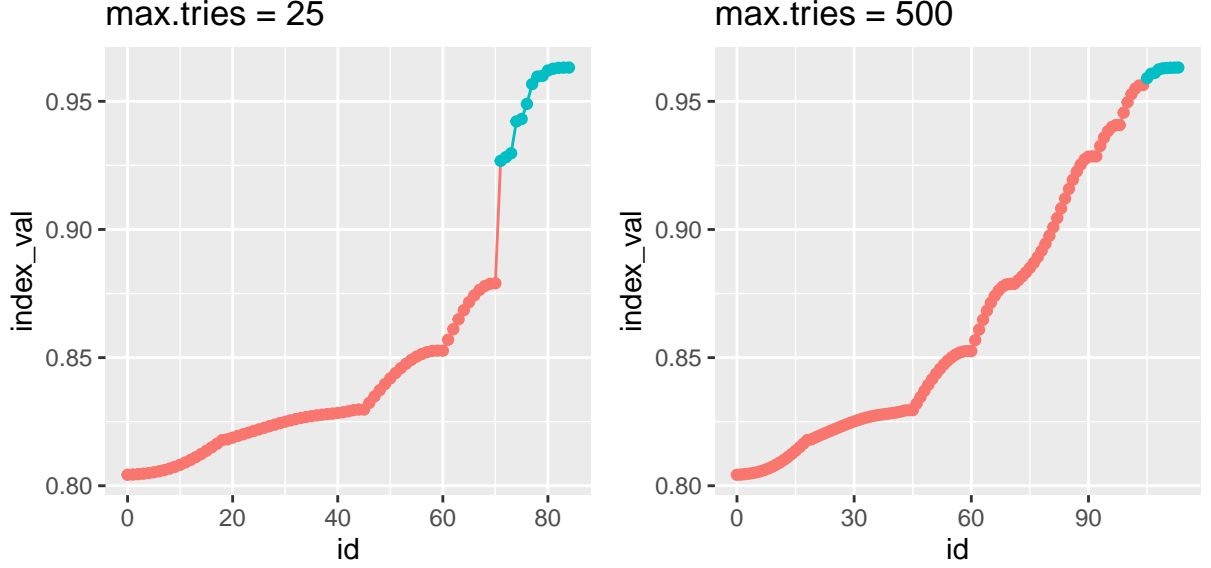
Figure 6: Breakdown of index value when using different max.tries in search better in conjunction with search polish. Both attain the same final index value after the polishing while using a max.tries 25 is not sufficient to find the ture maximum.

5.2 Explore searching space

In projection pursuit, the projection bases $\mathbf{A}_{p \times d}$ are usually of dimension $p \times d$ and hence can't be visualised in a 2D plot. An option to explore the searching space of these bases is to explore a reduced space via principal component analysis (PCA). The visualisation can thus be defined as

- Layer 1: point geom

  - data: subset the basis of interest and arrange into a matrix format; perform PCA on the basis matrix and compute the projected basis on the first two principal components; bind the variables from the original global object and form a tibble
  - x: the projected basis on the first principal component
  - y: the projected basis on the second principal component
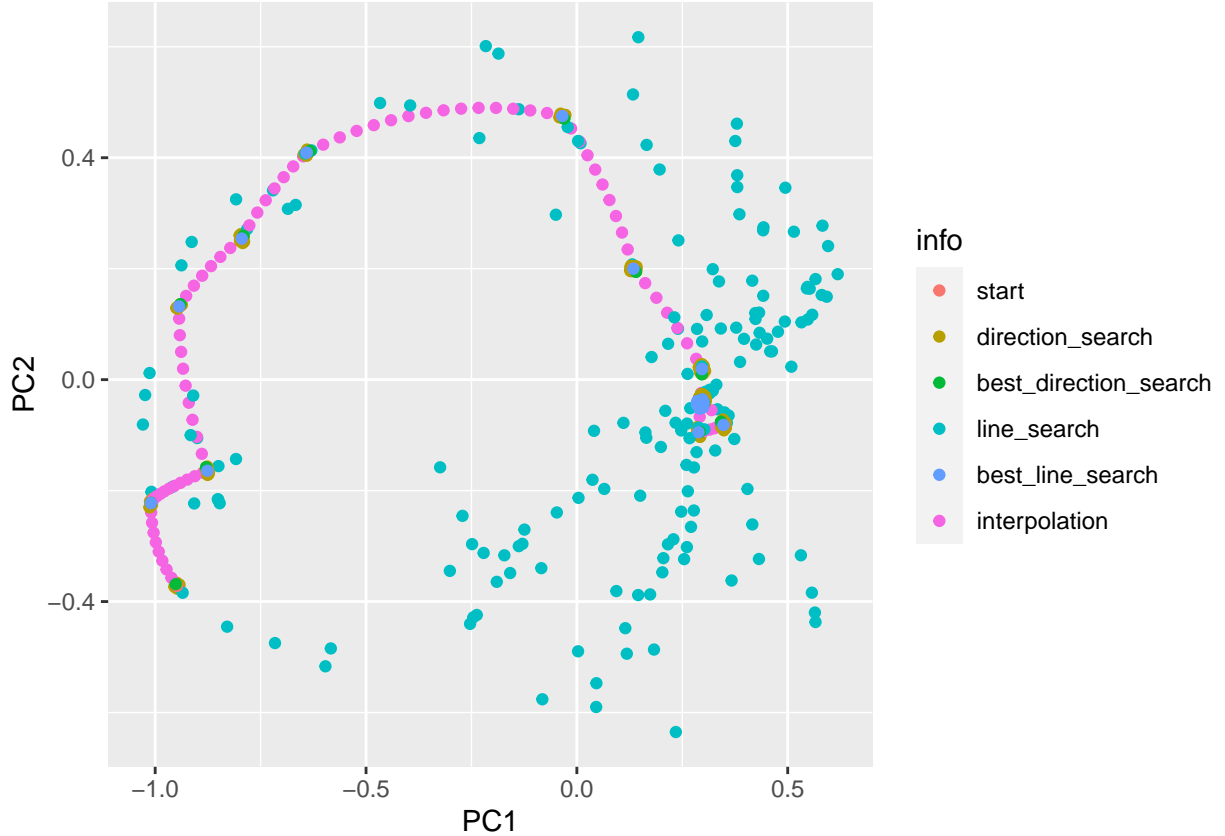 - colour: V is mapped to the colour aesthetic

Figure 7: PCA plot of search geodesic colouring by info allows for better understanding of each stage in the geodesic search

5.2.1 A toy example: understand different stage of search_geodesic

Example: understand search_geodesic [feel like this example is merely explaining search geodesic algorithm, so maybe introduce the animated plot here? xxx] `search_geodesic` is a two-stage ascending algorithm with four different stages in the search and a PCA plot useful to understand how the algorithm progresses and the relative position of each basis in the PCA projected 2D space. Starting from the start basis, a directional search is conducted in a narrow neighbourhood on five random directions. The best one is picked and a line search is then run on the geodesic direction to find the target basis. The starting and target bases are then interpolated. In the next iteration, the target basis becomes the current basis and then procedures continues.

5.2.2 A more complex example: Choosing the initial value for polishing parameter

Example: initial value for polishing alpha `search_polish` is a brutal-force algorithm that evaluate 1000 points in the neighbourhood at each loop. Setting an appropriate initial value for polish_alpha would avoid wasting search on large vector space that are not likely to produce higher index value. The default initial value for polishing step is 0.5 and we are interested in whether this is an appropriate initial value to use after `search_geodesic`. The problem is a 1D projection of the small dataset using `search_geodesic` and followed by `search_polish`. The top-left panel of Figure 8 displays all the projection bases on the first two principal components, coloured by the `polish_alpha`. We can observe that rather than concentrating on the ending basis from `search_geodesic` as what polishing step is designed, `search_polish` searches a much larger vector space, which is unnecessary. Thus a customised smaller initial value for `polish_alpha` would be ideal. One way to do this is to initialised `polish_alpha` as the projection distance between the last two target bases. The top-right panel of Figure 8 shows a more desirable concentrated searching space near the ending basis. Both specifications of initial value allow the searches to reach the same ending index values.

While explore the reduced space is an initial attempt to understand the searching space, there are existing technology for rotating a higher dimensional space for visualisation. Geozoo is an option. It generates random points on the high dimensional space and we can overlay it with the points on the optimisation path to visualise the spread of it on the high-D sphere.
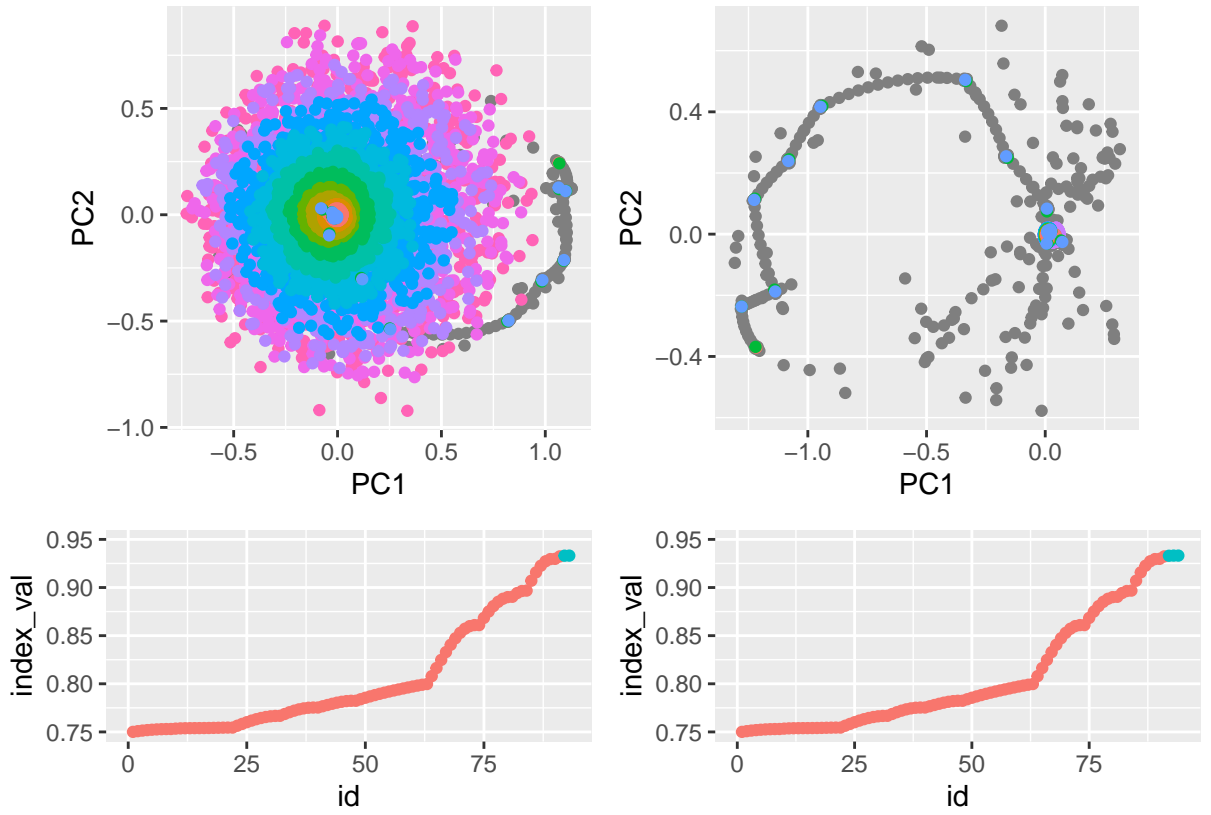
[add example from geozoo]

Figure 8: PCA plot of two different polish alpha initialisations. A default polish alpha = 0.5 searches a larger space that is unncessary while a small customised initial value of polish alpha will search near the ending basis. Both intialisations reach the same ending index values.

5.3 A comprehensive example of diagnosing a noisy index function

The interpolation path of holes index, as seen in Figure 5, is smooth, while this may not be the case for more complicated index functions. `kol_cdf` index, an 1D projection index function based on Kolmogorov test, compares the difference between the 1D projected data, $\mathbf{P}_{n \times 1}$ and a randomly generated normal distribution, y_n based on the empirical cumulated distribution function (ECDF). Denotes the ECDF function as $F(u)$ with subscript indicating the variable, the Kolmogorov statistics defined by

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

can be seen as a function of the projection matrix $\mathbf{A}_{p \times 1}$ and hence a valid index function.

5.3.1 Explore index value

Figure 9 compares the tracing plot of the interpolating points when using different optimisation algorithms: `search_geodesic` and `search_better`. One can observe that

- The index value of `kol_cdf` index is much smaller than that of holes index
- The link of index values from interpolation bases are no longer smooth
- Both algorithms reach a similar final index value after polishing

Polishing step has done much more work to find the final index value in `search_geodesic` than `search_better` and this indicates `kol_cdf` function favours of a random search method than ascent method.

Now we enlarge the dataset to include two informative variables: `x2` and `x3` and remain 1D projection. In this case, two local maximum appear when projection matrix being $[0, 1, 0, 0, 0, 0]$ and $[0, 0, 1, 0, 0, 0]$.

Using different seeds in `search_better` allows us to find both local maximum as in Figure 10. Comparing the maximum of both, we can see that the global maximum happens when `x2` is found. It is natural to ask then if there is an algorithm that can find the global maximum without trying on different seeds? `search_better_random` manages to do it via
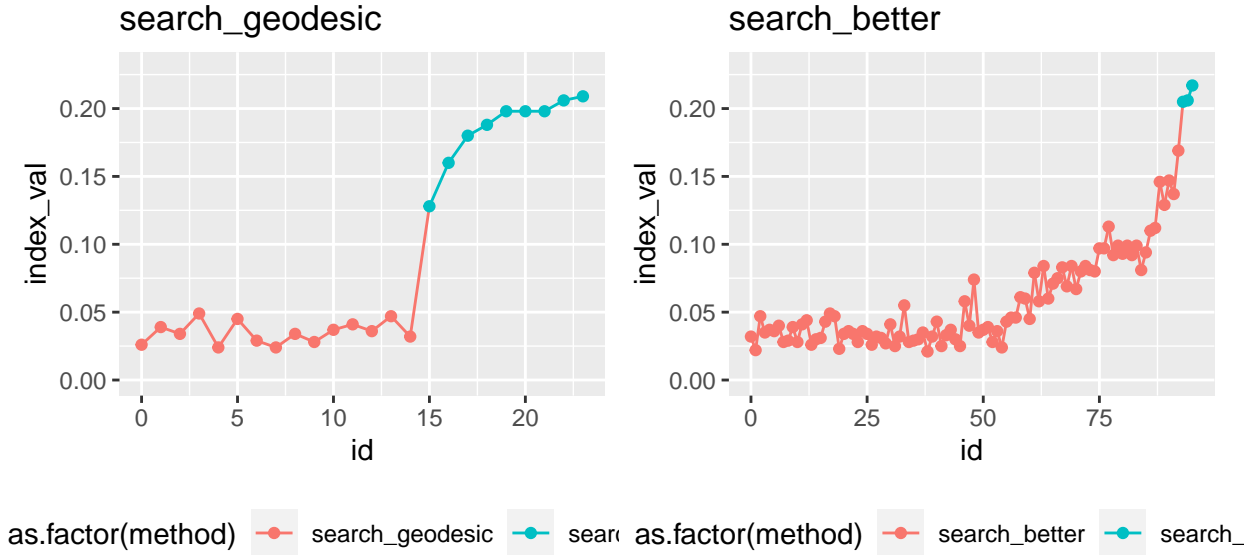
Figure 9: Comparison of two different searching methods: search_geodesic and search_better on 1D projection problem for a noisier index: kol_cdf. The geodesic search rely heavily on the polishing step to find the final index value while search better works well.

a metropolis-hasting random search as shown in Figure 11, although at a higher cost of number of points to evaluate.

5.3.2 Explore searching space

We can also plot the searching points of all the three algorithms in the searching space and explore their relative position against each other using principal components. As shown in Figure 12, the bases from better1 and better2 only search a proportion of the searching space while better_random produces a more exhaustive search. The large overlapping of better1 and better_random is explained by the fact that both algorithms finds x2 in the end.

6 Implementation: Ferrn package

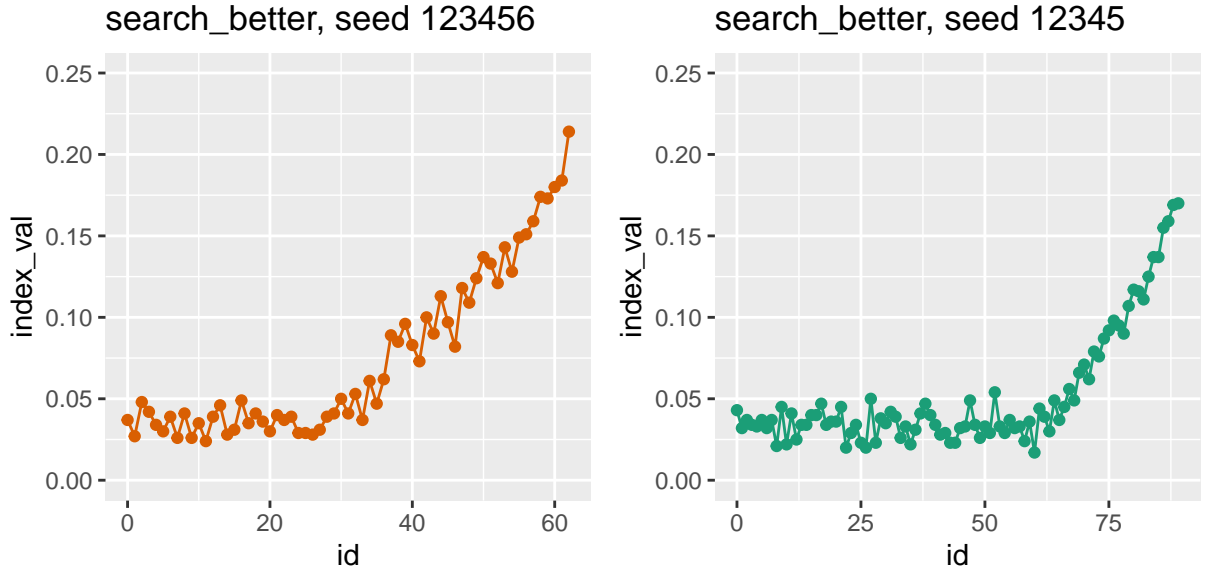Everything is coded up in a package. Package structure

Figure 10: The trace plot search better in a 1D projection problem with two informative variables using different seeds (without polishing). Since there are two informative variables, setting different value for seed will lead search better to find either of the local maximum.
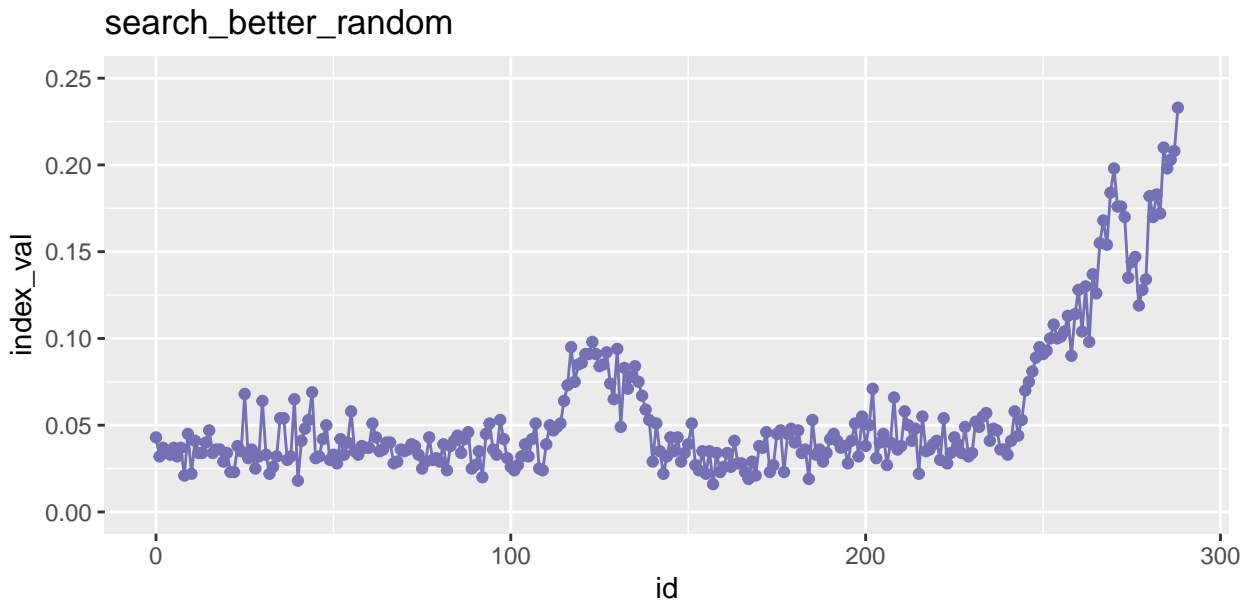

Figure 11: Using search better random for the problem above will result in finding the global maximum but much larger number of iteration is needed.
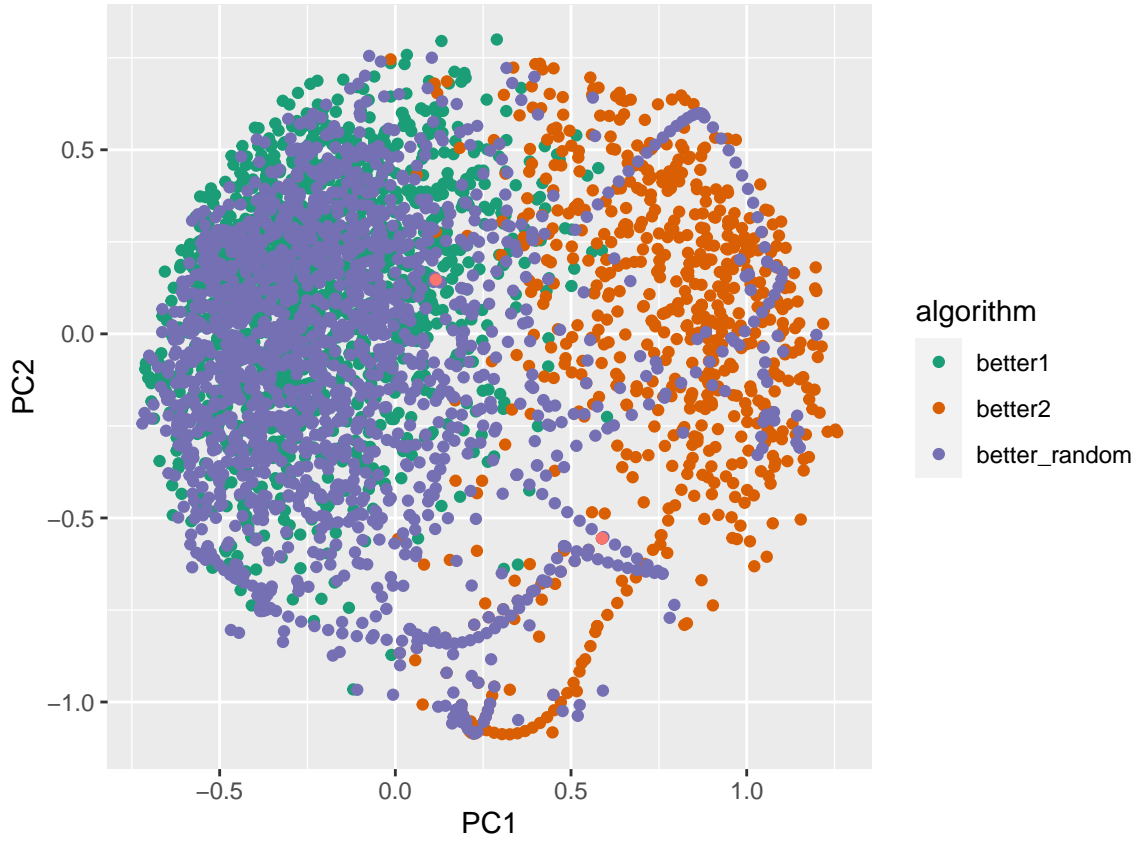
Figure 12: The projected projection basis using principal components. The bases from better1 and better2 only search a proportion of the searching space while better_random produces a more exhausive search. The large overlapping of better1 and better_random is explained by the fact that both algorithms finds x2 in the end.

7   Conclusion

References

Bertsimas, D., Tsitsiklis, J. et al. (1993), 'Simulated annealing', *Statistical science* **8**(1), 10–15.

Buja, A., Cook, D., Asimov, D. & Hurley, C. (2005), 'Computational methods for high-dimensional rotations in data visualization', *Handbook of statistics* **24**, 391–413.

Cook, D., Buja, A. & Cabrera, J. (1993), 'Projection pursuit indexes based on orthonormal function expansions', *Journal of Computational and Graphical Statistics* **2**(3), 225–250.

Cook, D., Buja, A., Cabrera, J. & Hurley, C. (1995), 'Grand tour and projection pursuit', *Journal of Computational and Graphical Statistics* **4**(3), 155–172.

Friedman, J. H. & Tukey, J. W. (1974), 'A projection pursuit algorithm for exploratory data analysis', *IEEE Transactions on computers* **100**(9), 881–890.

Hall, P. et al. (1989), 'On polynomial-based projection indices for exploratory projection pursuit', *The Annals of Statistics* **17**(2), 589–605.

Hooke, R. & Jeeves, T. A. (1961), '"direct search"solution of numerical and statistical problems', *Journal of the ACM (JACM)* **8**(2), 212–229.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *science* **220**(4598), 671–680.

Lee, E., Cook, D., Klinke, S. & Lumley, T. (2005), 'Projection pursuit for exploratory supervised classification', *Journal of Computational and graphical Statistics* **14**(4), 831–846.

Lee, E.-K. & Cook, D. (2010), 'A projection pursuit index for large p small n data', *Statistics and Computing* **20**(3), 381–392.

Posse, C. (1995), 'Projection pursuit exploratory data analysis', *Computational Statistics & data analysis* **20**(6), 669–687.

Wickham, H. (2010), 'A layered grammar of graphics', *Journal of Computational and Graphical Statistics* **19**(1), 3–28.

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.
URL: *https://ggplot2.tidyverse.org*

Wickham, H., Cook, D., Hofmann, H. & Buja, A. (2011), 'tourr: An R package for exploring multivariate data with projections', *Journal of Statistical Software* **40**(2), 1–18.
URL: *http://www.jstatsoft.org/v40/i02/*

Wickham, H. & Grolemund, G. (2016), *R for data science: import, tidy, transform, visualize, and model data*, " O'Reilly Media, Inc.".

Wickham, H. et al. (2014), 'Tidy data', *Journal of Statistical Software* **59**(10), 1–23.