Title here

Author 1 [*]
Department of YYY, University of XXX
and
Author 2
Department of ZZZ, University of WWW

July 21, 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

[*] The authors gratefully acknowledge . . .

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 color 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 paper is organised as follows. Section 2 gives a general literature review of optimisation, specifically derivative free optimisation. Section 4 presents the new idea of constructing a systematic visual framework that diagnoses the components of an optimisation procedure (parameters, searching path, etc). The rest of the paper serves as a comprehensive example of using the visual diagnostics on one particular problem: *projection pursuit guided tour*. Some background knowledge of projection pursuit guided tour is provided in Section 3. Section **??** applies the concepts proposed in section 4 in the tour problem and sets up the data. The last section, Section **??**, presents the visual diagnostic plots and explains how they can help to understand different aspects of the optimisation in tour.

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

From Section 3, we presents a comprehensive case study on how to use the visual diagnostics to explore the optimisation in a specific problem: projection pursuit guided tour. Section 3 aims to provide non-experts with an overview of the problem content and the existing optimisation procedures used in projection pursuit guided tour. For those who are already familiar with the techniques, feel free to skip this section.

The optimisation problem we're interested in is in the context of projection pursuit. Coined by Friedman & Tukey (1974), projection pursuit is a method that detects the interesting structure (i.e. clustering, outliers and skewness) of multivariate data via projecting it in lower dimensions. Let $\mathbf{X}_{n \times p}$ be a 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 value I, this function is commonly known as the projection pursuit index (PPI) function, or the index function and can be used to measure the "interestingness" of the projection. A number of indice functions have been proposed in the literature 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).

As Friedman & Tukey (1974) noted "..., the technique use for maximising the projection index strongly influences both the statistical and the computational aspects of the procedure." A suitable optimisation procedure is needed to find the projection angle that maximises the PPI and the quality of the optimisation largely affect the interesting projections one could possibly observe.

Projection pursuit is usually used in conjunction with a tour method called *guided tour*. Tour explores the multivariate data *interactively* via playing a series of projections, that form a *tour path* and guided tour uses the path that is geodesically the shortest. Details of the mathematical construction of a tour path can be found in Buja et al. (2005). Guided tour, along with other types of tour, 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 Notation

For a projection basis, let the three-subscript notation

$$\mathbf{A}_{jlk}$$

represent the projection matrix in iteration j with either a searching index l or an interpolation index k. The placeholder $*$ will be the substitue for the non-applicable index. To give two examples, $\mathbf{A}_{12*}$ denotes the second searching basis in the first iteration; $\mathbf{A}_{1*2}$ denotes the second interpolation basis in the first iteration.

If we are being pedantic, all the target bases, except the starting basis, should have a searching index since it is the last candidate basis that successfully allows the optimisation algorithm to end and output a new basis for interpolation. If we denotes the length of each search (number of basis searched in each iteration) as $l_1, l_2, \cdots, l_J$ and the length of each interpolation is $k_1, k_2, \cdots, k_J$, then the searching index for target basis in each iteration will become $\mathbf{A}_{1l_1*}, \mathbf{A}_{2l_2*}, \cdots, \mathbf{A}_{Jl_J*}$. We simplify this notation for target bases by using a superscript as

$$\mathbf{A}^0, \mathbf{A}^1, \mathbf{A}^2, \cdots, \mathbf{A}^J$$

where $\mathbf{A}^0$ is the starting basis. Notice that in the case when the complex notation can't be avoided, we shall separate each index using a comma and write the index in brackets if necessary. Using the notation above, all the bases on the interpolation path (including target and interpolation bases) can be written as in Equation 1.

$$\{\mathbf{A}^0, \mathbf{A}^1, \mathbf{A}_{1*1}, \mathbf{A}_{1*2}, \cdots, \mathbf{A}^2, \mathbf{A}_{2*1}, \mathbf{A}_{2*2}, \cdots, \mathbf{A}^J, \mathbf{A}_{J*1}, \mathbf{A}_{J*2}, \cdots, \mathbf{A}_{J*K}, \} \tag{1}$$

where K denotes the last interpolating index. A visual representation of the interpolation path (for iteration one) modified from Buja et al. (2005) is shown in Figure 1. If we use a dot to represent a plane in the searching space, the searching points along with the current and target basis in iteration one can sketched in Figure 5.1.1. Combine the two above, we can write all the bases in the first iteration (including the starting basis) in its natural ordering as in Equation 2.

$$\{\mathbf{A}^0, \mathbf{A}_{11*}, \mathbf{A}_{12*}, \cdots, \mathbf{A}^1, \mathbf{A}_{1*1}, \mathbf{A}_{1*2}, \cdots, \mathbf{A}_{1*k_1}\} \tag{2}$$

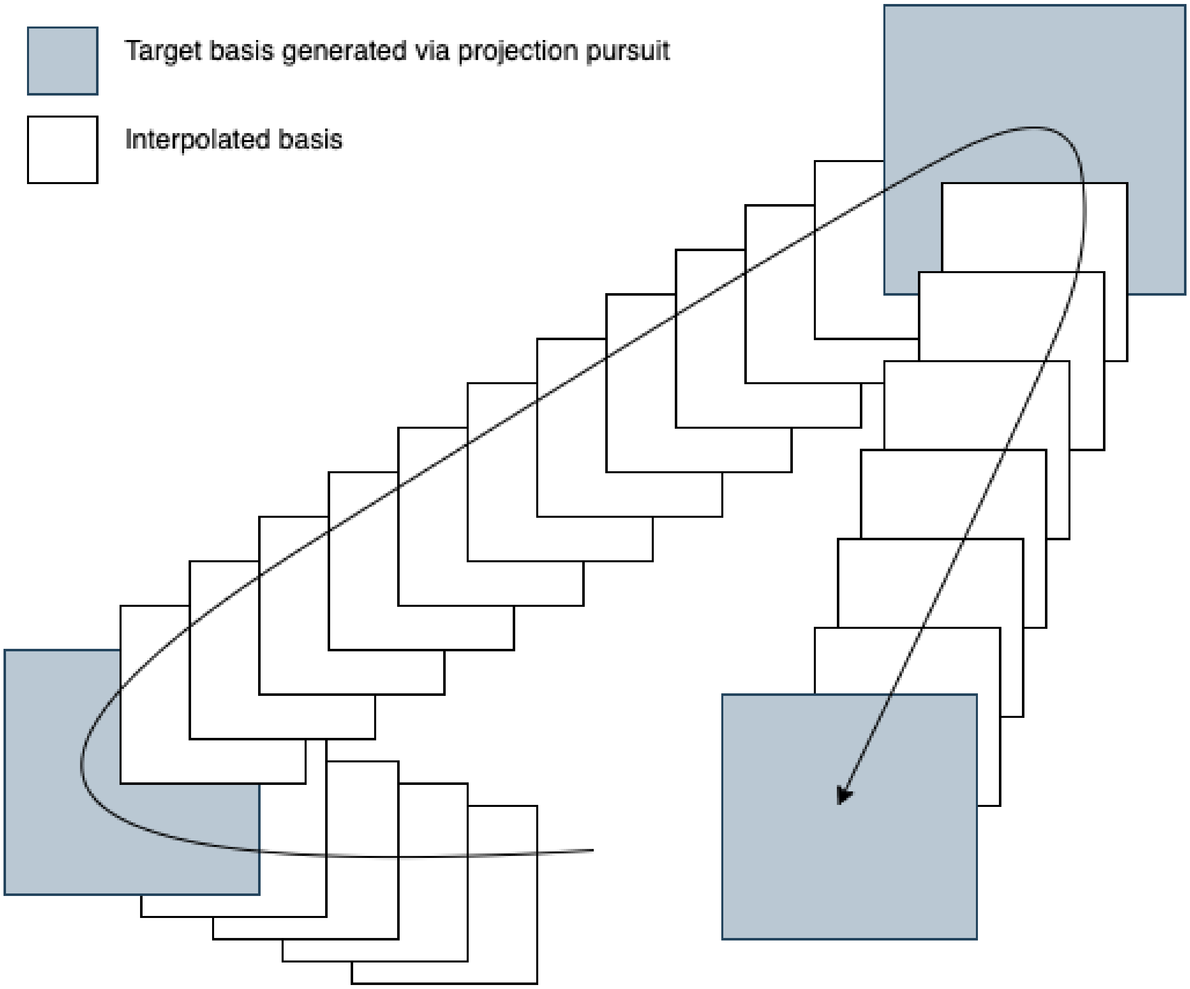

Figure 1: An illustration of the tour path

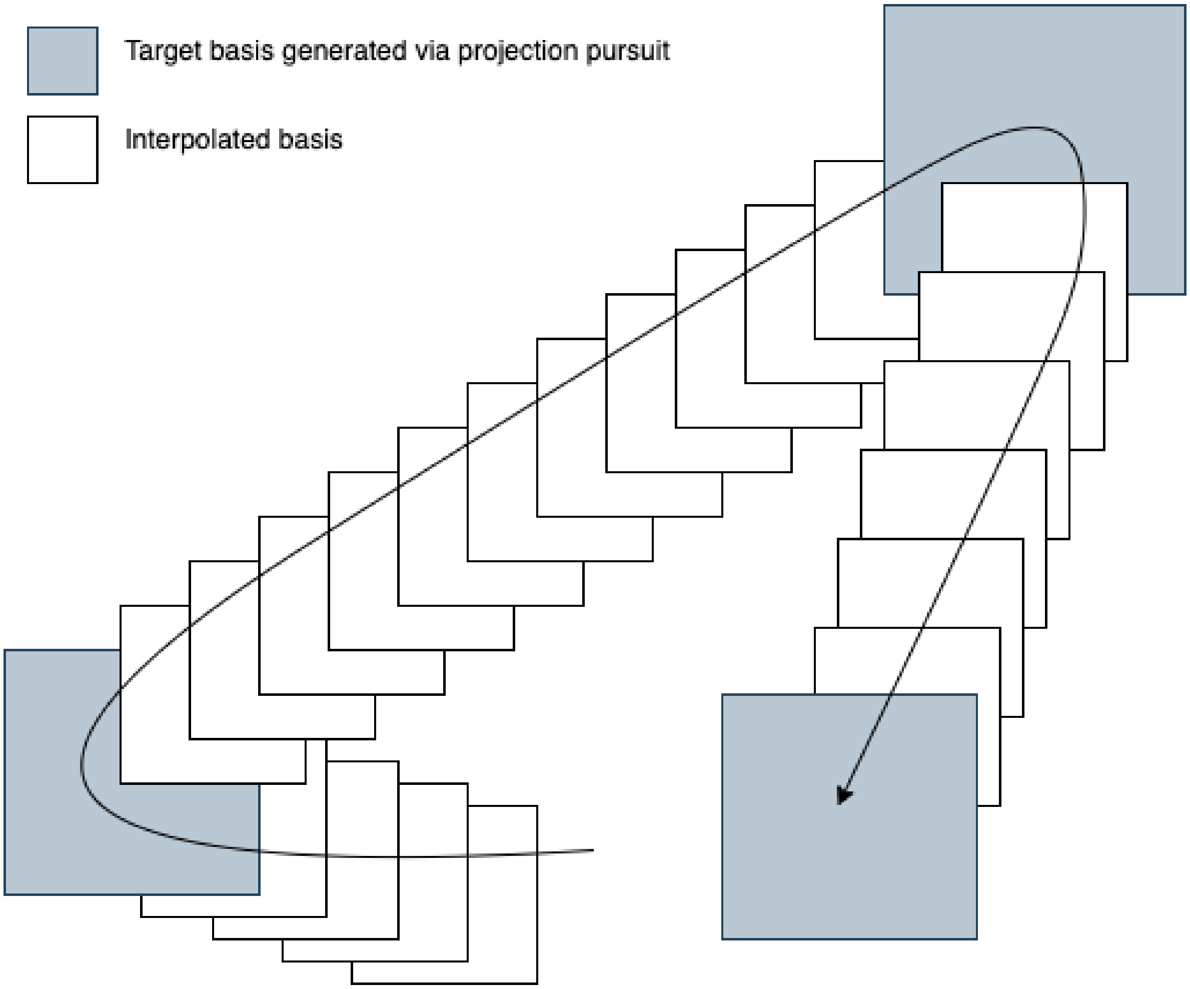

Figure 2: An illustration of the tour path

3.2 Optimisation problem and existing algorithms

Now we begin to formulate the optimisation problem. Given a randomly generated starting basis $\mathbf{A}^0$, projection pursuit finds the final projection basis $\mathbf{A}^J = [\mathbf{a}_1, \cdots, \mathbf{a}_d]$, where $\mathbf{a}_i$ is a $p \times 1$ vector, satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{A}) \tag{3}$$

$$s.t. \mathcal{A} = \{\forall \mathbf{a}_i, \mathbf{a}_j \in \mathbf{A} : \|\mathbf{a}_i\| = 1 \text{ and } \mathbf{a}_i\mathbf{a}_j = 0\} \tag{4}$$

via an iterative search of target basis: $\{\mathbf{A}^0, \mathbf{A}^1, \mathbf{A}^2, \cdots\cdots \mathbf{A}^J\}$.

There are three existing methods for optimisating PPI function and we review them below.

Posse (1995) proposed a stochastic direct search method, a random search algorithm. Given a current basis $\mathbf{A}^{j-1}$, a candidate basis $\mathbf{A}_{jl*}$ is sampled in the neighbourhood defined by the radius of the p-dimensional sphere, α of the current basis $\mathbf{A}^j$ by $\mathbf{A}_{jl*} = \mathbf{A}^{j-1} + \alpha\mathbf{A}_{rand}$. If the candidate basis has a higher index value than the current basis, it is outputed as the target basis $\mathbf{A}^j$ along with other metadata and the current iteration stops. If no basis is found to have higher index value after the maximum number of tries $l_{\max}$, the algorithm stops with nothing outputted. c is the halfing parameter and when $c > 30$, the nieghbourhood parameter α in the next iteration will be reduced by half. The algorithm for a random search is summarised in Algorithm 1.

Cook et al. (1995) explained the use of a gradient ascent optimisation with the assumption that the index function is continuous and differentiable. Since some indices could be non-differentiable, the computation of derivative is replaced by a pseudo-derivative of evaluating five randomly generated directions in a tiny nearby neighbourhood. Taking a step on the straight derivative direction has been modified to maximise the projection pursuit index along the geodesic direction. See Algorithm 2 for a summarisation of the steps.

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: $\mathbf{A}^{j-1}$; The index function: $f$
output: The global object; The target basis: $\mathbf{A}^{j}$
initialisation;
while $l < l_{\max}$ do
    Generate $\mathbf{A}_{jl*} = \mathbf{A}^{j-1} + \alpha\mathbf{A}_{\text{rand}}$ ensuring $\mathbf{A}_{j,l,*}$ is orthonormal;
    Compute $I_{jl*} = f(\mathbf{A}_{jl*})$;
    if $I_{jl*} > I^{j-1}$ then
        $A^{j} = A_{jl*}$, $I^{j} = I_{jl*}$;
    else
        $c = c + 1$;
    end
    $l = l + 1$;
end
```

basis A_{rand}, a candidate basis is constructed as $\mathbf{A}_{jl*} = (1-\alpha)\mathbf{A}^{j} + \alpha\mathbf{A}_{rand}$. If the index value of the candidate basis is larger than the one of the current basis, the candidate basis becomes the target basis. If it is smaller, the candidate is accepted with probability $P = \min\left\{\exp\left[\frac{I^{j-1}-I_{jl*}}{T(l)}\right], 1\right\}$ where I^{j-1} and I_{jl*} are the index value of the current and candidate basis respectively. The algorithm can be summarised as in Algorithm 3.

Below listed several features characterise the optimisation procedures needed in projection pursuit

- *Being able to handle non-differentiable PPI function*: The PPI function could be non-differentiable, thus derivative free methods are preferred.

- *Being able to optimise with constraints*: The constraint comes from projection matrix being an orthonormal matrix.

- *Being able to find both local and global maximum*: Although the primary interest is to find the global maximum, local maximum could also reveal structures in the data that are of our interest.

Algorithm 2: search geodesic

```
input  : The current projection basis: $\mathbf{A}^{j-1}$; The index function: $f$
output: The global object; The target basis: $\mathbf{A}^{j}$
initialisation;
while $l < l_{\max}$ do
    Generate ten bases in five random directions: $\mathbf{A}_{jl*} : \mathbf{A}_{j,(l+9),*}$ within a small
     neighbourhood dist;
    Find the direction with the largest index value: $\mathbf{A}_{j,l_d,*}$;
    Construct the geodesic from $\mathbf{A}^{j-1}$ to $\mathbf{A}_{j,l_d,*}$;
    Find $\mathbf{A}_{j,l_g,*}$ on the geodesic that has the largest index value ;
    Compute $I_{j,l_g,*} = f(\mathbf{A}_{j,l_g,*})$, $p_{\text{diff}} = (I_{j,l_g,*} - I^{j-1})/I_{j,l_g,*}$;
    if $p_{diff} > 0.001$ then
        $A^j = A_{j,l_g,*}$, $I^j = I_{j,l_g,*}$;
    end
    $l = l + 1$;
end
```

Algorithm 3: simulated annealing

```
input  : The current projection basis: $\mathbf{A}^{j-1}$; The index function: $f$
output: The global object; The target basis: $\mathbf{A}^j$
initialisation;
while $l < l_{\max}$ do
    Generate $\mathbf{A}_{jl*} = (1-\alpha)\mathbf{A}^{j-1} + \alpha\mathbf{A}_{rand}$ ensuring $\mathbf{A}_{jl*}$ is orthonormal ;
    Commpute $I_{jl*} = f(\mathbf{A}_{jl*})$ and $T(l) = \frac{T_0}{\log(l+1)}$;
    if $I_{jl*} > I^{j-1}$ then
        $A^j = A_{jl*}$, $I^j = I_{jl*}$;
    else
        Compute $P = \min\left\{\exp\left[\frac{I^{j-1}-I_{jl*}}{T(l)}\right], 1\right\}$;
        Draw $D$ from a uniform distribution: $D \sim \text{Unif}(0,\ 1)$;
        if $P > D$ then
            $A^j = A_{jl*}$, $I^j = I_{jl*}$;
        end
    end
    $l = l + 1$;
end
```

4 Visual diagnostic system

[I would expand this section more as the core contribution. ppp]

4.1 Motivation

Random search methods has a black-box mechanism and focuses solely on finding the global maximum point, while the projection pursuit problem we have aims at *exploring* the data and thus is interested in how the algorithm finds its maximum. This motivates us **to develop a visual diagnostic system for exploring the optimisation searching path**.

The necessity of developing such a system rather than simply producing different diagnostic plots is because the diagnostics of each variable requires a different function and these functions can't be scaled to other problems. Thus we want to establish a set of rules that can generalise the diagnostic of iterative algorithms.

The idea of generalising all the diagnostic plots under one framework 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". By this design, ggplot doesn't need to develop a gazillion of functions that each produces a different type of plot. Instead, it aesthetically maps the variables to the geometric objects and builds the plot through different layers.

4.2 Global object

Ggplot requires a data frame that contains all the variables to plot and a *global object* is constructed as the data frame supplied to the visual diagnostic plots to better suit the characters of iterative optimisation algorithms. Given an optimisation algorithm, two primary variables of interest are the *decision variable:* $\mathbf{A}$ and the *value of the objective function:* I. To further simplify the notation for $\mathbf{A}$, we write the bases as a column vector denoting by a single subscript by their natural ordering as in Equation 5.

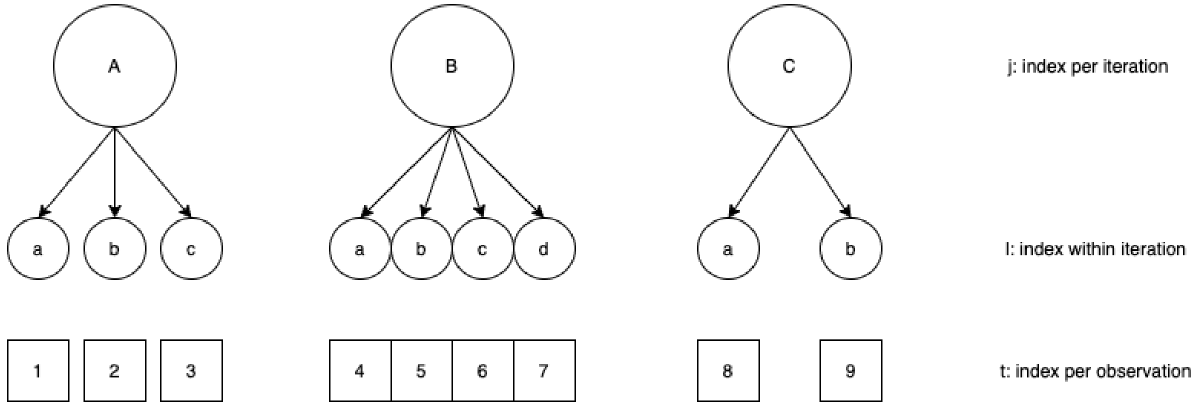

Figure 3: A sketch of the design of iterators in iterative algorithms.

$$\{\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_T\} \tag{5}$$

where T is the total number of plane serached by the optimisation algorithm including all the target planes, all the interpolating planes and all the candidate planes but NOT the starting plane.

Iterators indexes the data collected and has a time series feature that prescribes the its natural ordering of the decision varible in the searching. The simpliest iterator, *index per observation:* t, is a unique identifier for each observation in the data. For each level of iteration, we design two types of indices: *index per iteration:* j is fixed for each observations in one iteration and has an increment of one once a new iteration starts. *index within iteration:* l has an increment of one for each observation in an iteartion and starts over from one once a new iteration starts. A sketch of the difference between these three iterators is provided in Figure 3. In projection pursuit optimisation algorithms, there is one level of iteartion and hence exists three iterators: `id` indices each observation by a unique number; `tries` is the index per iteration iterator that gets updated once a search-and-interpolate step is finished; and `loop` is the index within iteration iterator and starts over from one at the beginning of a new `tries`. We give the interpolating basis a different index k for projection pursuit guided tour. It is similar in nature to l but is specific for interpolating bases, which is usually not part of the optimisation.

There could exist other parameters that are also of our interest but can't be classified as one of the three categories. They are defined under the fourth category: *other parameter of interest: S*. In projection pursuit, three other parameters of our interest includes `method`, `alpha` and `info`. `method` identifies the name of the searching method used and we are interested in comparing the performance between different algorithms under direct search. The neighbourhood parameter `alpha` controls the size of the sampling space and we are interested to understand how the searching space shrinks as the algorithm progresses. `info` labels different stages in the searching process. A sketch of the global object for projection pursuit guided tour is presented in Figure 4. The full data structure can thus be shown as in Equation 6.

$$\left[\begin{array}{c|cc|ccc|ccc}
t & \mathbf{A} & I & j & l & k & S_1 & \cdots & S_p \\
\hline
1 & \mathbf{A}_0 & I_0 & 0 & * & * & * & \cdots & * \\
\hline
2 & \mathbf{A}_1 & I_1 & 1 & 1 & * & S_{1,1} & \cdots & S_{1,p} \\
3 & \mathbf{A}_2 & I_2 & 1 & 2 & * & S_{2,1} & \cdots & S_{2,p} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & 1 & l_1 & * & \vdots & \cdots & \vdots \\
\hline
\vdots & \vdots & \vdots & 1 & * & 1 & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & 1 & * & 2 & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & 1 & * & k_1 & \vdots & \cdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
T+1 & \mathbf{A}_T & I_T & J & l_J & * & S_{T,1} & \cdots & S_{T,p}
\end{array}\right] \tag{6}$$

4.3 Simulated data

We first simulate some random variables of size 1000 with different structures. `x1`, `x8`, `x9` and `x10` are simulated from normal distribution iwth zero mean and variance of one as in equation 7. 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 distribution with different weights and locations. Equation 8 to 13 show the distribution of each variables and Figure 5 shows

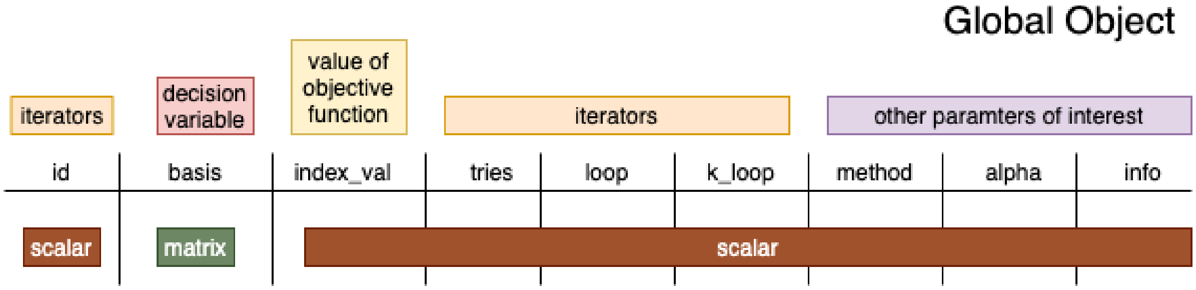

Figure 4: The global object in projection pursuit guided tour.

the histogram of each variable except `x3`. All the variables are then scaled to ensure the mixture has variance of one.

$$x_1 \overset{d}{=} x_8 \overset{d}{=} x_9 \overset{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{7}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{8}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{10}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{11}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{12}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{13}$$

We form our first dataset using variable `x1`, `x2`, `x8`, `x9` and `x10` and supply it to one of the above algorithm, say `search_geodesic` in the `tourr` package. 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 `verbose = TRUE`. Below shows the first five rows of the global object. The global object meets the definition of tidy data introduced in (Wickham et al. 2014) where each observation forms a row, each variable forms a column and each type of observational unit forms a table. The adoption of tidy data makes it easier for data wrangling and visualisation, which powers the visual diagnostics system introduced in this paper.

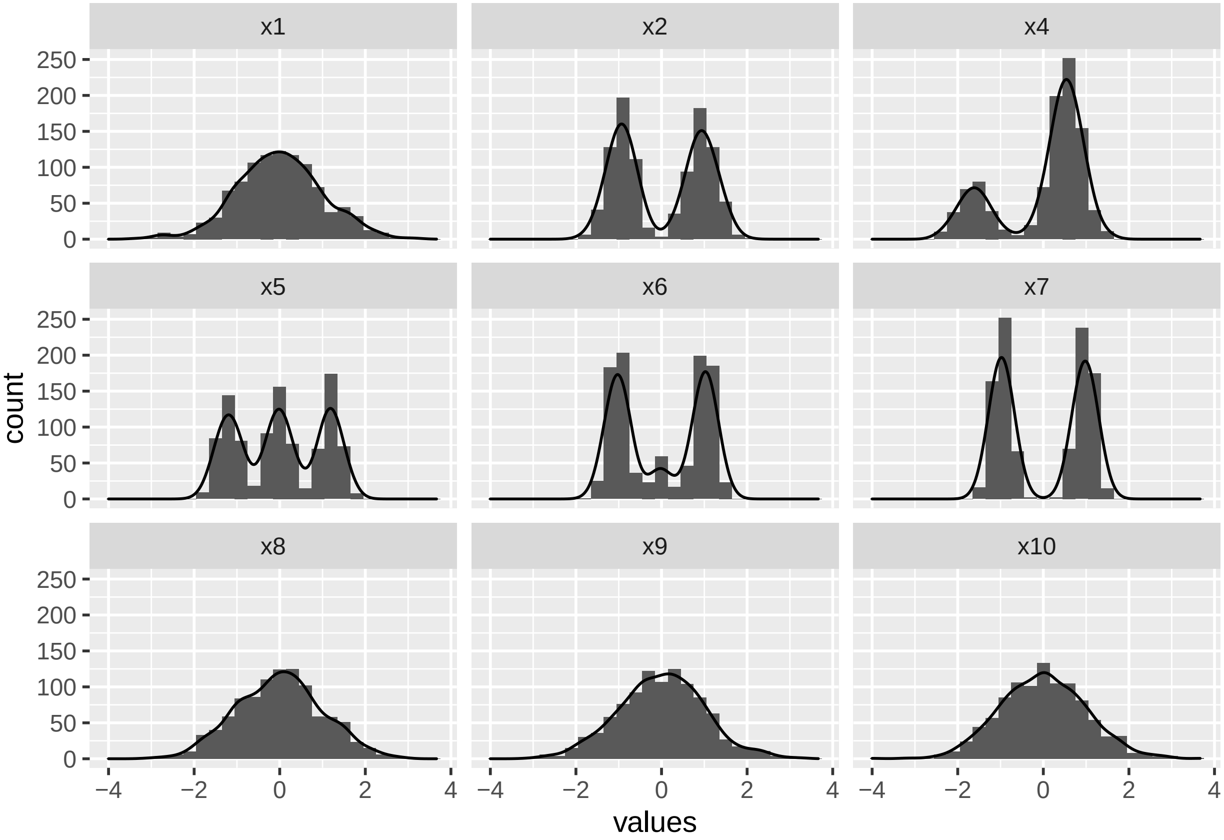

Figure 5: The distribution of simulated data except x3

```
## # A tibble: 5 x 8
##   basis             index_val tries info          loop method      alpha    id
##   <list>                <dbl> <dbl> <chr>        <dbl> <chr>       <dbl> <int>
## 1 <dbl[,1] [5 x 1]>     0.749     1 start           NA <NA>          0.5     1
## 2 <dbl[,1] [5 x 1]>     0.730     1 random_search    1 search_bett~   0.5     2
## 3 <dbl[,1] [5 x 1]>     0.743     1 random_search    2 search_bett~   0.5     3
## 4 <dbl[,1] [5 x 1]>     0.736     1 random_search    3 search_bett~   0.5     4
## 5 <dbl[,1] [5 x 1]>     0.747     1 random_search    4 search_bett~   0.5     5
```

5 Visual diagnostics

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. Remeber 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 preceive 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 `tries` and filter the observations in the group that have count greater than `cutoff = 15`.
  - x: `tries` is mapped to the x-axis
  - y: the index value after statistical transformation $y = \{Q_q(I_{jlk} \cdot \mathbb{1}(j = x, k = *))\}$ is mapped to the y-axis where q takes one of $0, 25, 50, 75, 100$ and $Q_q(x)$ finds the qth-quantile for the vector x. $\mathbb{1}(.)$ is the identity operator.

- Layer 2: point geom
  - data: group by `tries` and filter the observations in the group that have count less than `cutoff = 15`.
  - x: `tries` is mapped to the x-axis
  - y: `index_val` is mapped to the y-axis

- Layer 3: line geom
  - data: filter the points with the highest index value in each `tries`
  - x: `tries` is mapped to the x-axis
  - y: `index_val` 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 6. 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.

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:

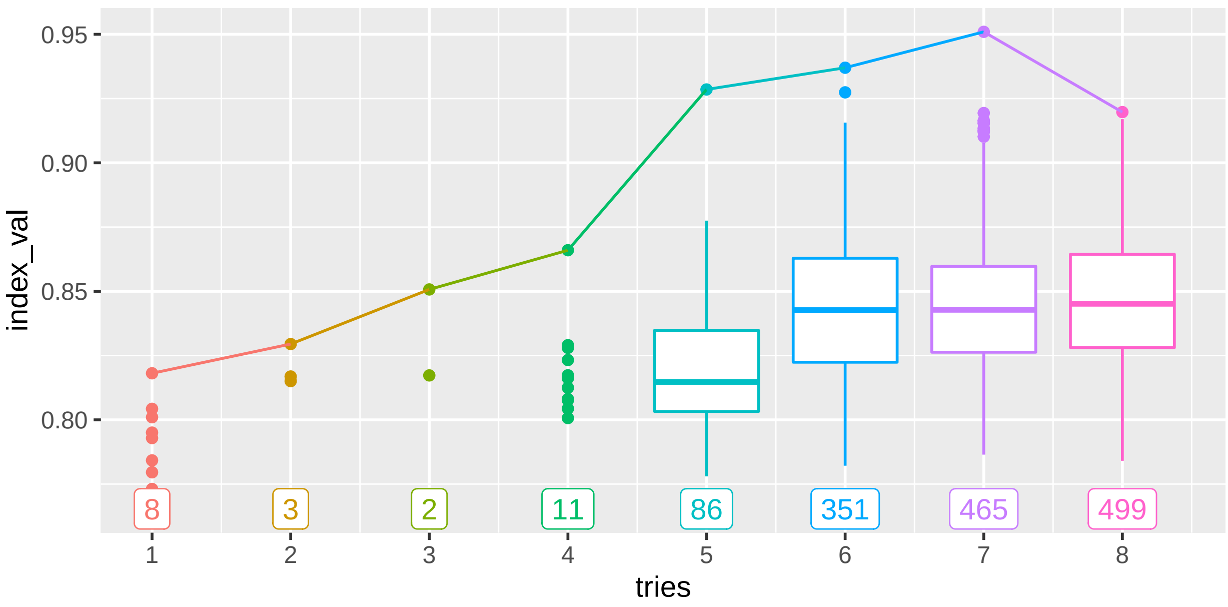

Figure 6: A comparison of plotting the same search points with different plot designs. The left plot doesn't efficiently use the plot space to convey information from the plot while the right plot provides good summarisation of data and number of points in each tries.

- Layer 1: point geom
  - data: filter the observations with `info` being `interpolation` and mutate `id` to be the row number of the subsetted tibble
  - x: `id` is mapped to the x-axis
  - y: `index_val` is mapped to the y-axis
- Layer 2: line geom
  - using line goemetry for the same data and aesthetics

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 7, 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

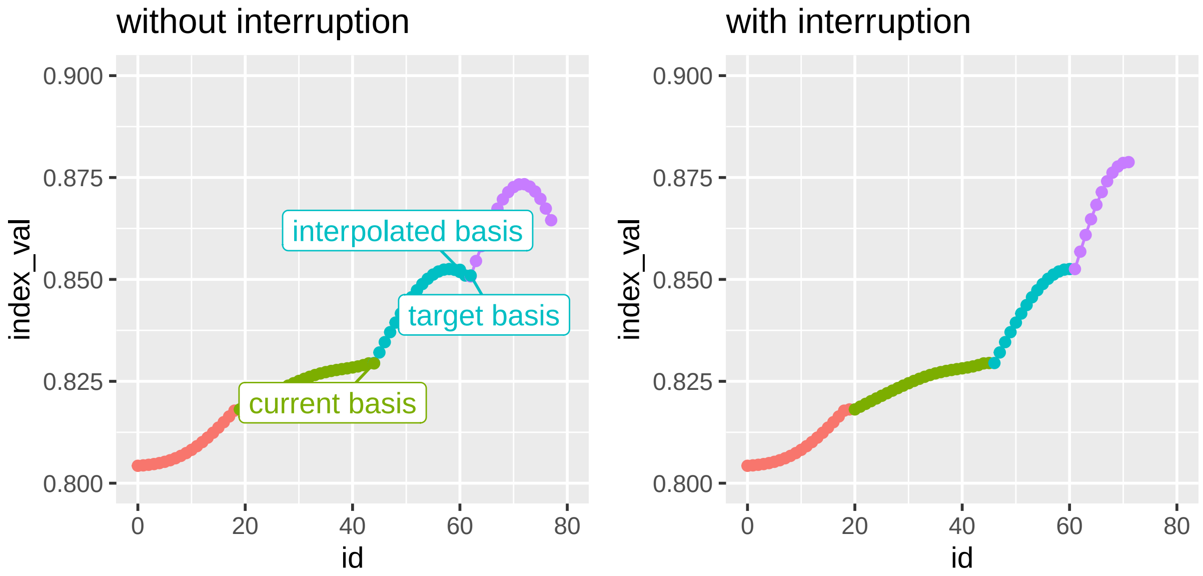

Figure 7: Trace plots of the interpolated basis with and without the interruption. The interruption stops the interpolation when the index value starts to decrease at id = 60. The implementation of the interuption finds an ending basis with higher index value using fewer steps.

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 7.

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 `info` being `interpolation`; bind the global object from optimisation and interpolation and form polishing; mutate `id` to be the row number of the binded tibble.
    - x: `id` is mapped to the x-axis
    - y: `index_val` is mapped to the y-axis
    - color: `method` is mapped to the color aesthetic
- Layer 2: line geom
    - using line goemetry for the same data and aesthetics

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 8, 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 succiciently explore the parameter space.

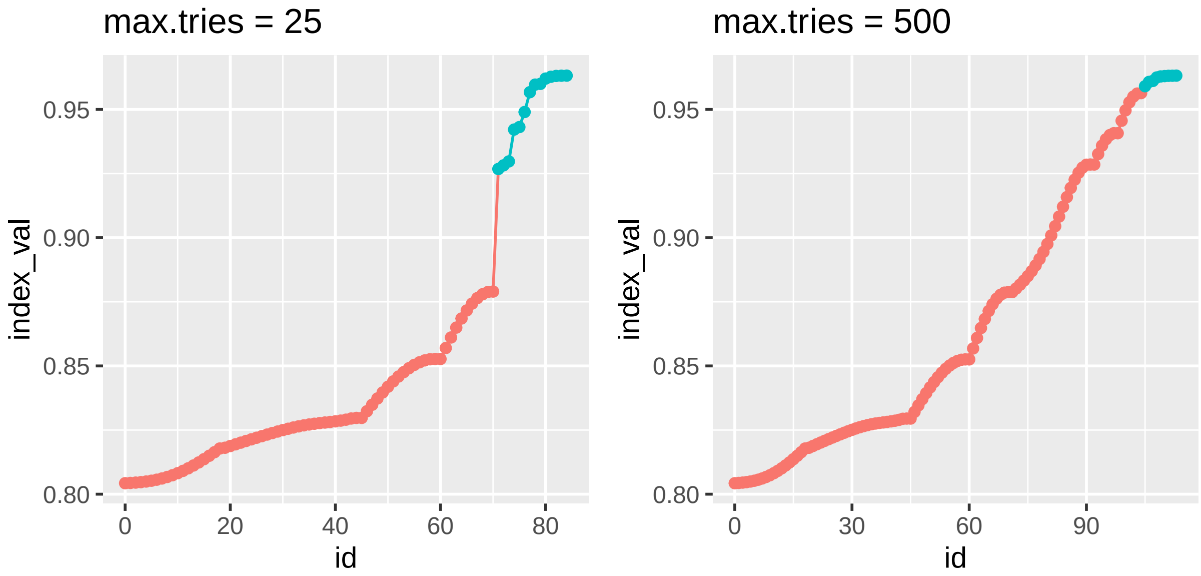

Figure 8: Breakdown of index value when using different max.tries in search better in conjunction with search polish. Both attain the same final index value after the polishing while using a max.tries 25 is not sufficient to find the ture maximum.

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
  - color: an variable of interest is mapped onto color

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.

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 10 displays all the projection bases on the first two principal components, colored 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 10 shows a more desirable concentrated searching space near

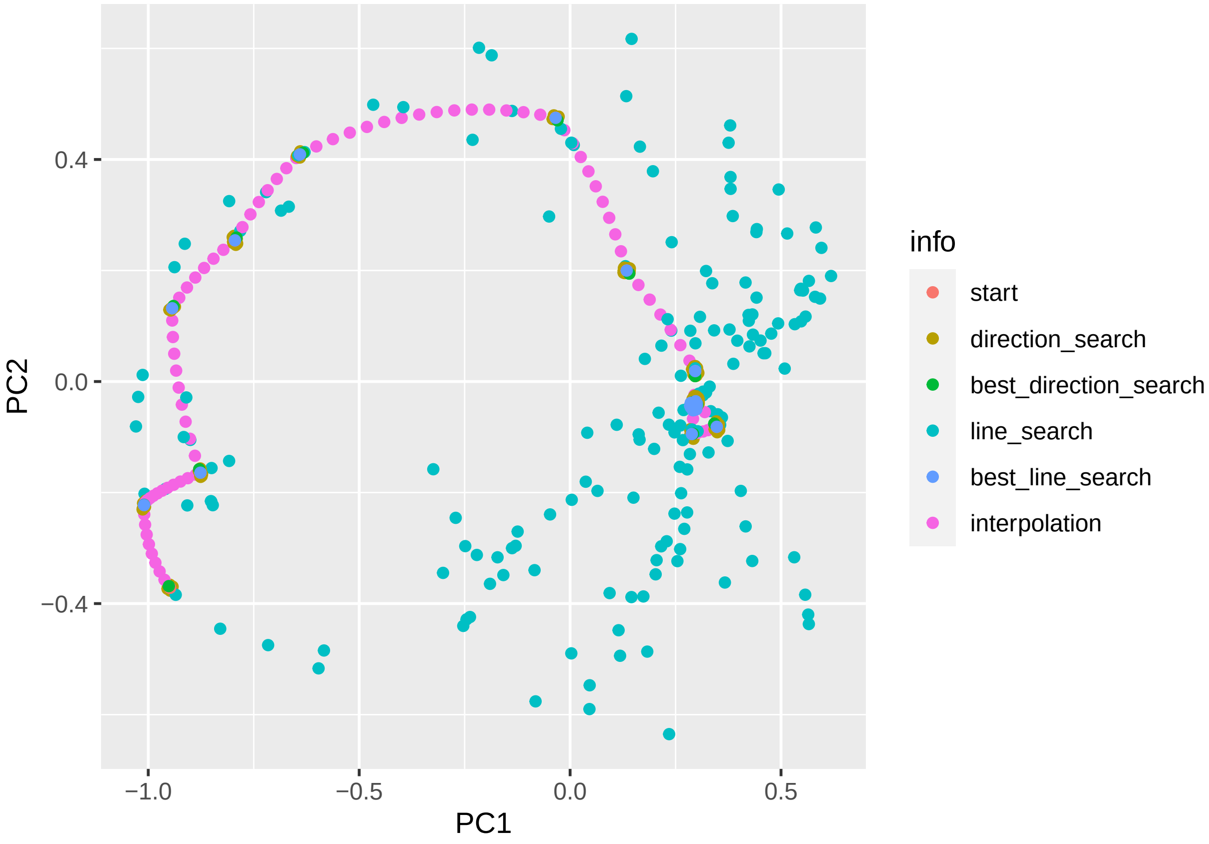

Figure 9: PCA plot of search geodesic Coloring by info allows for better understanding of each stage in the geodesic search

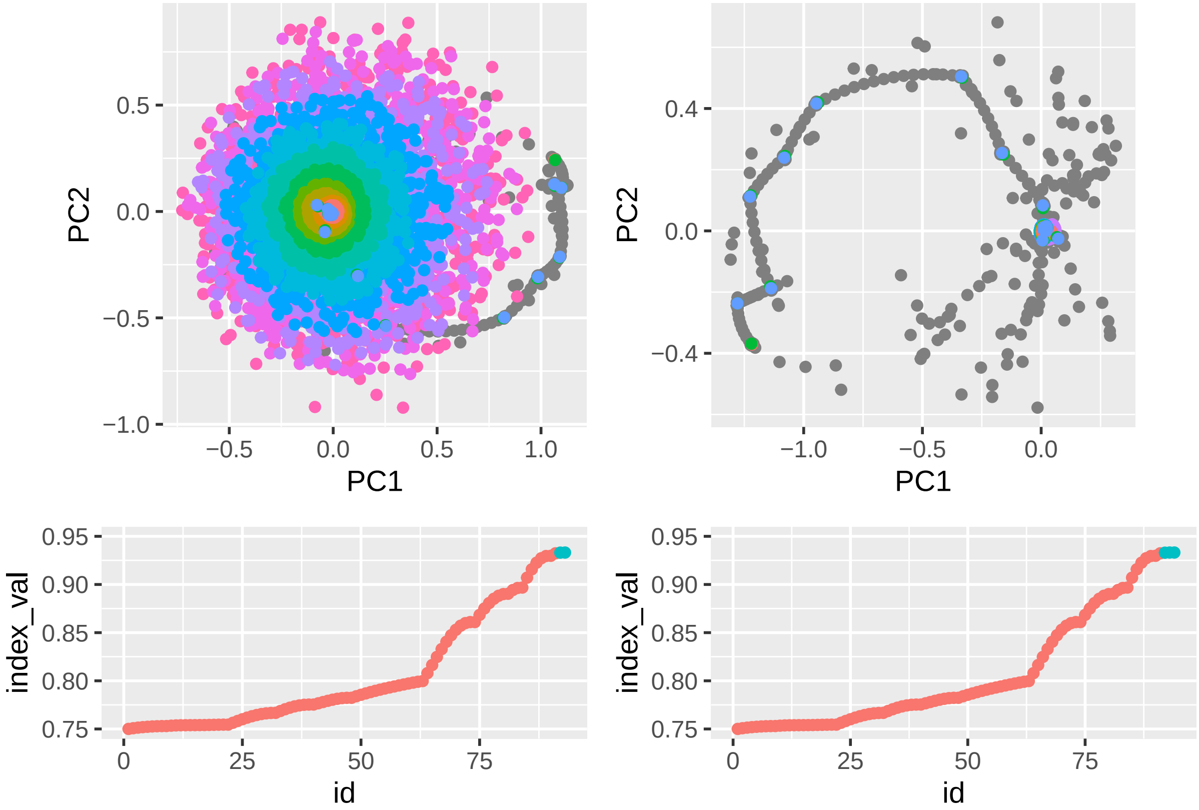

Figure 10: PCA plot of two different polish alpha initialisations. A default polish alpha = 0.5 searches a larger space that is unncessary while a small customised initial value of polish alpha will search near the ending basis. Both intialisations reach the same ending index values.

the ending basis. Both specifications of initial value allow the searches to 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 7, 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 incidating the variable, the Kolmogorove 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 11 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 ub `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 12. 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

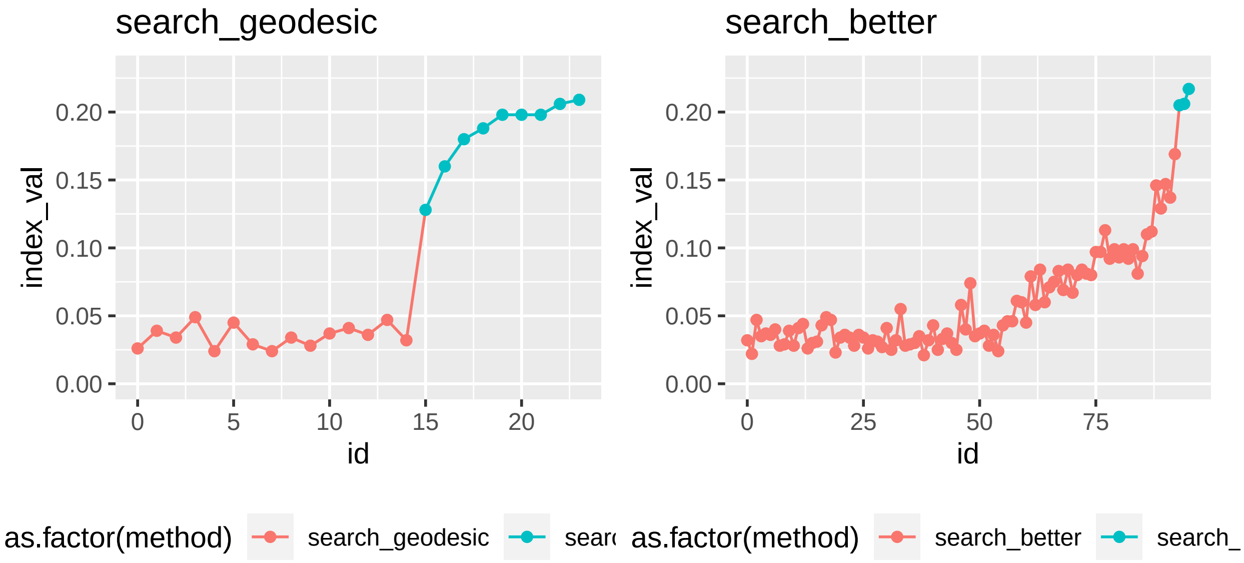

Figure 11: Comparison of two different searching methods: search_geodesic and search_better on 1D projection problem for a noisier index: kol_cdf. The geodesic search rely heavily on the polishing step to find the final index value while search better works well.

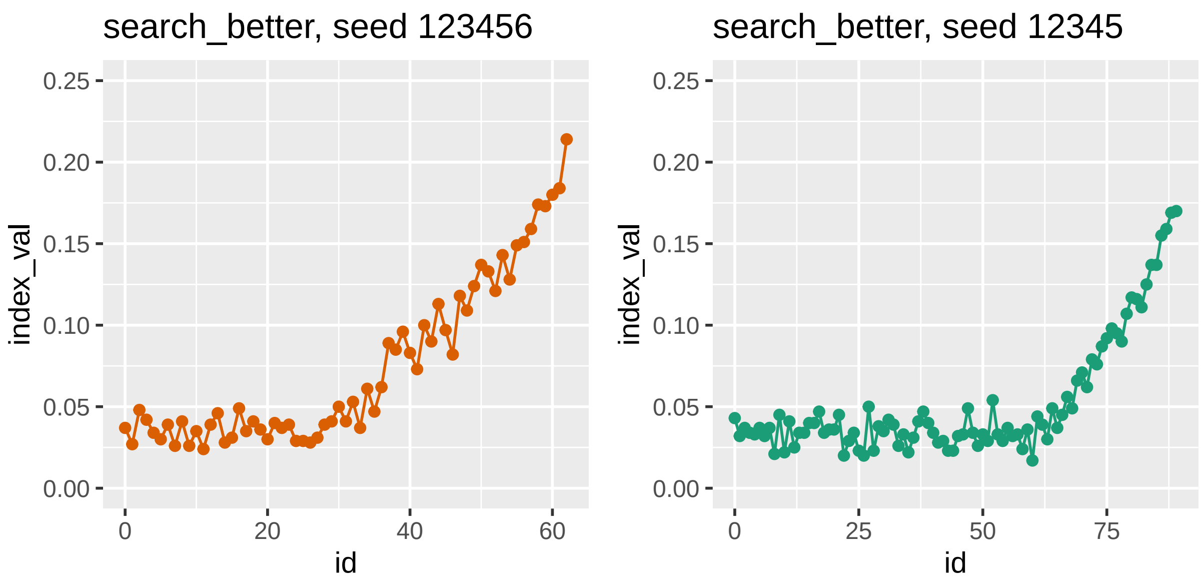

Figure 12: The trace plot search better in a 1D projection problem with two informative variables using different seeds (without polishing). Since there are two informative variables, setting different value for seed will lead search better to find either of the local maximum.

a metropolis-hasting random search as shown in Figure 13, 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 14, 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.

6 Implementation: Ferrn pacakge

Everything is coded up in a package. Package structure

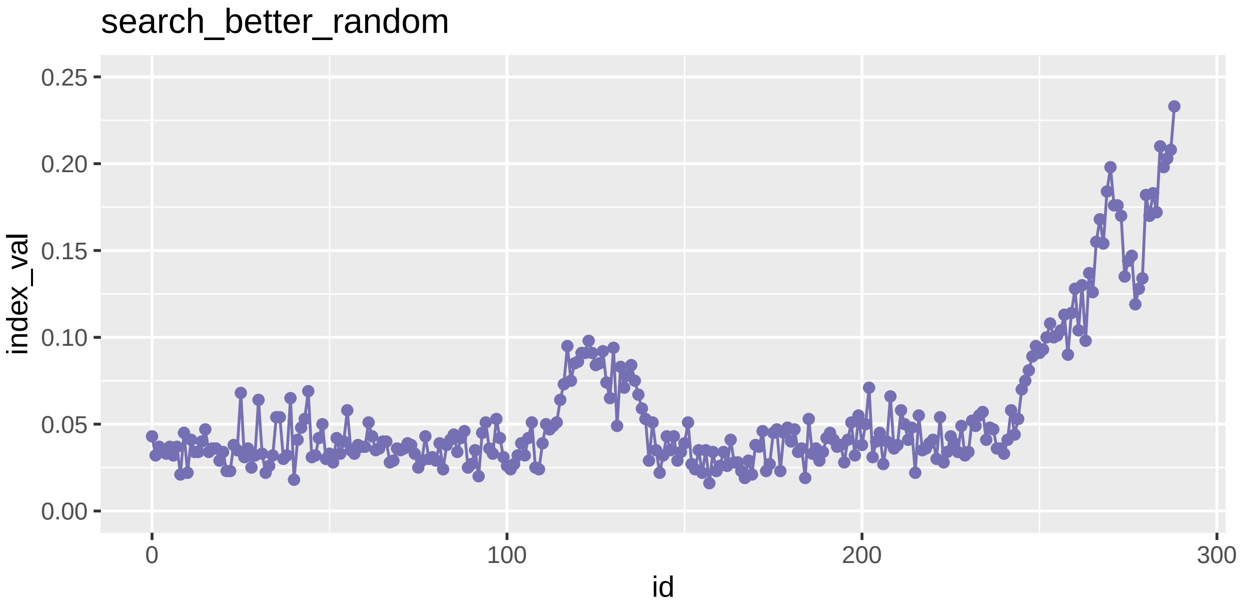

Figure 13: Using search better random for the problem above will result in finding the global maximum but much larger number of iteration is needed.

7 Conclusion

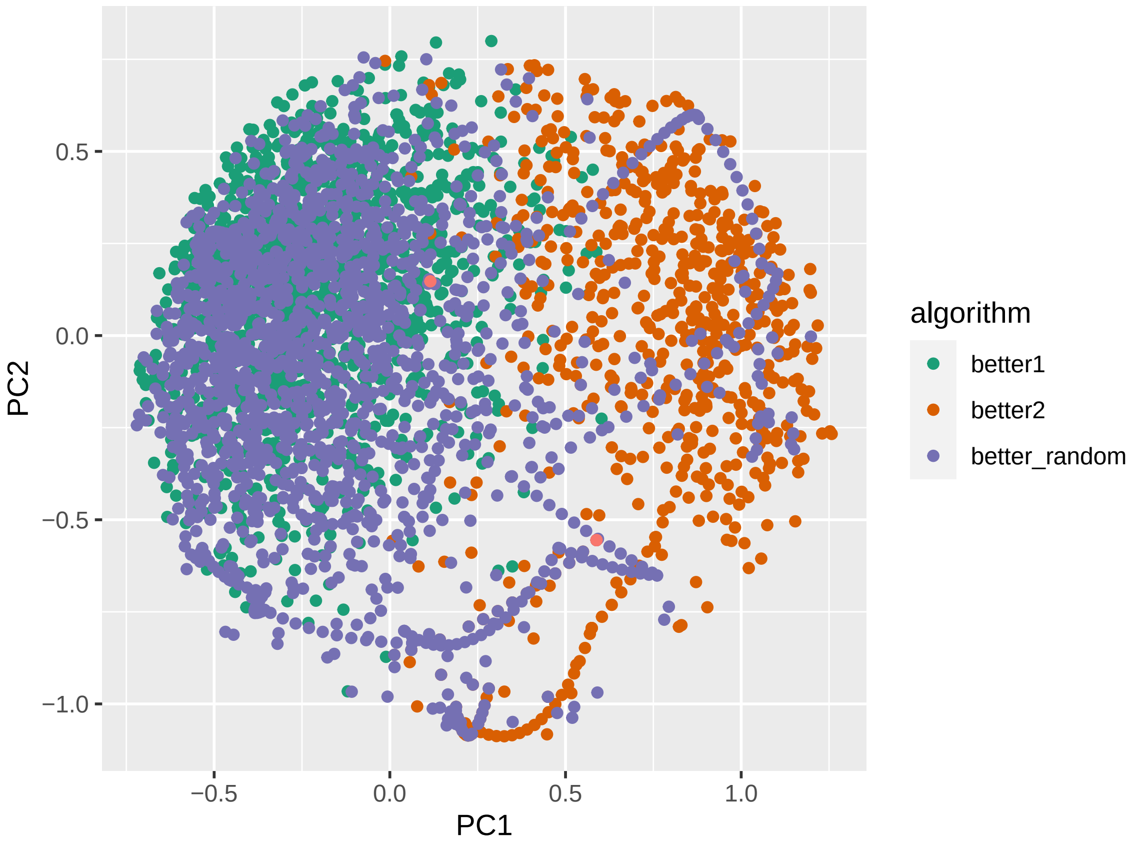

Figure 14: The projected projection basis using principal components. The bases from better1 and better2 only search a proportion of the searching space while better_random produces a more exhausive search. The large overlapping of better1 and better_random is explained by the fact that both algorithms finds x2 in the end.

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.