Title here

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

September 6, 2020

Abstract

Friedman & Tukey commented on their initial paper on projection pursuit in 1974 that "the technique used 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 more generally to help practitioner to unveil the black-box in randomised iterative (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

Visualisation has been widely used in exploratory data analysis. Presenting information in a graphical format often allows people to see information they would otherwise not see. This motivates our work of creating plots to diagnose optimisation algorithms in the context of projection pursuit guided tour, with the aim to understand and compare features of different existing algorithms.

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 in optimizing an objective function $f : S \to \mathbb{R}$ with $S \in \mathbb{R}^n$ in a reduced space given by the problem constraints to either minimize or maximize a function.

Projection pursuit and guided tour are exploratory data analysis tools that detect interesting structure of high dimensional data through projection on low dimensional space. Optimisation is applied here to search for the low dimensional space that finds the most interesting projection.

The remainder of the paper is organised as follows. Section 2 provides a literature review of optimisation methods, specifically the line search methods used in projection pursuit guided tour. Section 3 reviews projection pursuit guided tour, forms the optimisation problem, and introduces three main existing algorithms. Section 4 presents the new visual diagnostics design, from forming the data object to the definition of different diagnostic plots with some small examples. Section 5 shows the application of how the diagnostic plots designed in section 4 can be used to understand and compare different algorithms and how they contribute to modifications that improve the algorithms. Finally, Section 6 describes the R package: ferrn, that implements all the visual diagnostics above.

2 Optimisation Methods

Given an optimisation problem, two basic approaches find the optimum based on different thinking. An analytical approach aims to find the optimal solution in a finite number of steps, but a potential issue with it is that the closed-form solution may not be available when the problem starts to become complex. An iterative approach, on the other hand, finds the optimum based on the idea of making progressive improvement to the current solution. An iterative method may end up finding a local optimum but the progressive nature of the algorithm allows the practitioner to decide when to stop if a desirable accuracy has been achieved.

A traditional while often used in practice is an iterative method called *line search method* (Fletcher 2013). In a simple one-dimensional problem of finding the x that minimises $f(x)$, line search achieves the goal via an iterative algorithm in the form of Equation 1.

$$x^{(j+1)} = x^{(j)} + \alpha_k * d^{(j)} \tag{1}$$

where $d^{(j)}$ is the searching direction in iteration j, and α_j is the step-size. Strictly speaking, α_k is chosen by another minimisation of $f(x^{(j)} + \alpha * d^{(j)})$ with respect to α and theoretical results have demonstrated the global convergence of the algorithm when the exact minimisation of α_j is attained (Curry 1944). In practice, this second minimisation is rarely implemented due to its computational demanding or even the existence of such a minimisation. A more realistic approach is to impose a mandatory decrease in the objective function for each iteration: $f^{(j+1)} > f^{(j)}$ and despite we lose the guarantee on global convergence, this approach turns out to be efficient in practical problems.

[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 projection pursuit context? Some of the problems listed below are also applicable to optimization problem in general too. ppp]

3 Projection pursuit guided tour

Modern development of the line search methods focuses on proposing different computation on the searching direction: $d^{(j)}$ and various approximations on the step size: α_j catered for practical optimisation problems. The specific problem context we are interested in is called projection pursuit guided tour. Projection pursuit and guided tour are two separate methods in exporatory data analysis focusing on different aspects: coined by Friedman & Tukey (1974), projection pursuit detects interesting structures (i.e. clustering, outliers and skewness) in multivariate data via low dimensions projection; whilst guided tour is a particular variation in a broader class of data visualisation method called tour.

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 $f : \mathbb{R}^{n\times d} \mapsto \mathbb{R}$ to be an index function that maps the projection basis $\mathbf{A}$ onto an 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 a 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 used 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 the bases that give interesting projections. While we leave the formal construction of the optimisation problem and existing algorithms to section 3.1, we outline the general idea here. Given a random starting (current) basis, projection pursuit repeatedly searches for candidate bases nearby until it finds one with higher index value than the current basis. In the second round, that basis becomes the current basis and the repetitive sampling continues. The process ends until no better basis can be found or one of the termination criteria is reached.

Before introducing the guided tour, we shall be familiar with the general tour method (Cook et al. 2008). A tour produces animated visualisation of the high dimensional data via rotating low dimension planes. The smoothness of the animation is ensured by computing a series of intermediate planes between two low dimension planes via geodesic interpolation and we refer readers to Buja et al. (2005) for the mathematical details. Iteratively choosing different low dimension planes and interpolating between them forms a tour path. Different types of tour methods choose the low dimensional planes differently and we mention two other types of tour that are commonly used. A grand tour selects the planes randomly in the high dimensional space and hence serves as an initial exploration of the data. Manual control allows researches to fine-tuning an existing projection by gradually phase in and out one variable.

Guided tour chooses the planes produced by optimising the projection pursuit index function. Figure 1 shows a sketch of the tour path consisting of the blue frames produced by the projection pursuit optimisation algorithm iteratively and the white frames, which are the interpolations between 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).

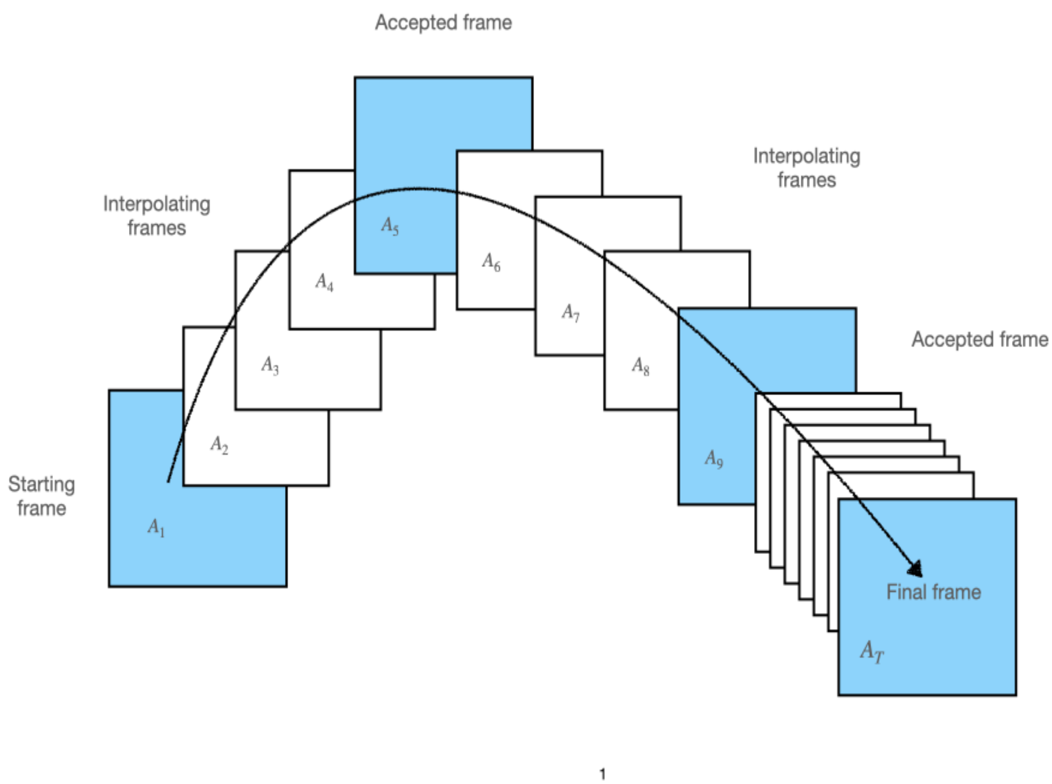

Figure 1: An illustration of the tour path

3.1 Optimisation problem

Now we begin to formulate the optimisation problem. Given a randomly generated starting basis $\mathbf{A}_1$, projection pursuit finds the final projection basis $\mathbf{A}_T$ that satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{X}\cdot\mathbf{A}) \tag{2}$$

$$s.t. \mathbf{A}'\mathbf{A} = I_d \tag{3}$$

where I_d is the d-dimensional identity matrix. The constraint requires the projection basis $\mathbf{A}$ to be an orthogonal matrix with each column vector being orthonormal.

There are several features of this optimisation that are worth noticing. First of all, it is a multivariate constraint optimisation problem. Since the decision variables are the entries of a projection basis, it is required to be orthonormal. It is also likely that the objective function is non-differentiable or the gradient information is simply not available. In this case, we will need to either use some approximation of the gradient or turn to derivative free methods. Given the goal of projection pursuit as finding the basis with the largest index value, the optimisation problem needs to be able to find the global maximum. Along the way, local maximum may also be of our interest since they could present unexpected interesting projections. There is also one computational consideration: the optimisation procedure needs to be easy to compute since the tour animation is played in real-time.

3.2 Existing algorithms

Below we introduce three possible algorithms: `search_better` and `search_better_random` are derivative free methods that sample candidate basis in the neighbourhood whilst `search_geodesic` approximates the gradient information and acts as a non-derivative version of the gradient ascent.

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 $\mathbf{A}_l = \mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{\text{rand}}$, where α controls the radius of the sampling neighbourhood and $\mathbf{A}_{\text{rand}}$ is a randomly generated matrix with the same dimension as $\mathbf{A}_{\text{cur}}$. 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 halving 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 accepted 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. [mention orthonormalise to ensure the constraint is fulfilled; don't use derivative information but a random search]

Simulated annealing (Kirkpatrick et al. 1983, Bertsimas et al. (1993)) modifies `search_better` based on a non-increasing cooling scheme $T(j)$. Given an initial T_0, the temperature at iteration k is defined as $T(j) = \frac{T_0}{\log(j+1)}$. When a candidate basis fails to have an index value larger than the current basis, simulated annealing gives it a second chance to be accepted with probability

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

where $I_{(\cdot)}$ denotes the index value of a given basis. This implementation allows the algorithm to jump out of a local maximum and enables a more holistic search of the whole parameter space. This feature is particularly useful when the dimension of the projected space is smaller than the number of informative variables in the dataset (i.e. a one dimensional projection of the dataset with two informative variables). The algorithm can be written as replacing line 5-11 of Algorithm 1 with Algorithm 2.

Algorithm 1: random search

input : $\mathbf{A}_{\text{cur}}$, f, α, $l_{\max}$

output: $\mathbf{A}_l$

initialisation;

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

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

  Generate $\mathbf{A}_l = \mathbf{A}_{\text{cur}} + \alpha \mathbf{A}_{\text{rand}}$ and orthonormalise $\mathbf{A}_l$;

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

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

    return $\mathbf{A}_l$;

  else

    $c = c + 1$;

  end

  $l = l + 1$;

end

We repeat this loop and if $c > 30$, half the α. Reset c to zero.

Algorithm 2: simulated annealing

Compute $I_j = f(\mathbf{A}_j)$ and $T(k) = \frac{T_0}{\log(j+1)}$;

if $I_j > I_{cur}$ **then**

  return $\mathbf{A}_j$;

else

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

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

  if $P > U$ **then**

    return $\mathbf{A}_j$;

  end

end

Cook et al. (1995) used a gradient ascent algorithm with pseudo-derivative. Instead of computing the actual gradient of the index function with respect to the projection basis matrix, `search_geodesic` samples $2n$ bases that are randomly generated on a uniform open ball with the radius controlled by the δ parameter and finds the most promising one to construct the geodesic direction as an approximation to the gradient. The δ parameter is usually set to be tiny to ensure the geodesic direction is a good local approximation to the gradient. Once the searching direction is determined, the optimiser finds the best projection $\mathbf{A}_{**}$ on the geodesic as the candidate. $\mathbf{A}_{**}$ is outputted for the current iteration if the percentage change in the index value between $\mathbf{A}_{**}$ and $\mathbf{A}_{\text{cur}}$ is greater than a threshold value or the algorithm repeats the above steps until $l_{\max}$ is reached and the searching terminates. Algorithm 3 summarises the steps in geodesic search.

Algorithm 3: search geodesic

input : $\mathbf{A}_{\text{cur}}$, f, $j_{\max}$, n, δ

output: $\mathbf{A}_{**}$

initialisation;

Set $j = 1$;

while $j < j_{\max}$ **do**

  Generate $2n$ bases in n random directions: $\mathbf{A}_j : \mathbf{A}_{j+9}$ within a small neighbourhood δ;

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

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

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

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

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

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

  end

  $j = j + 1$;

end

4 Visual diagnostics

To be able to make diagnostic plot, the optimisation algorithm should populate a data structure that contains the key elements of the algorithm. When the algorithm runs, key information regarding the decision variable, objective function and hyper-parameters needs to be recorded and stored as a data object for future analysis.

4.1 Data structure for diagnostics

In the optimisation algorithms for projection pursuit, three main elements to record are 1) projection bases: $\mathbf{A}$, 2) index values: I, and 3) State: S, which labels the observation with detailed stage in the optimisation. Multiple iterators are also 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 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 the algorithm has 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, the neighbourhood parameter that controls the size in sampling candidate bases. A matrix notation of the data structure is presented in Equation 4.

$$
\left[\begin{array}{c|ccc|cc|cc}
t & \mathbf{A} & I & S & j & l & V_1 & V_2 \\
\hline
1 & \mathbf{A}_1 & I_1 & S_1 & 1 & 1 & V_{11} & V_{12} \\
\hline
2 & \mathbf{A}_2 & I_2 & S_2 & 2 & 1 & V_{21} & V_{22} \\
3 & \mathbf{A}_3 & I_3 & S_3 & 2 & 2 & V_{31} & V_{32} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & l_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & 2 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & 2 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & k_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T & \mathbf{A}_T & I_T & S_T & J & l_J & V_{T1} & V_{T2} \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & J & k_J & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J+1 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T' & \mathbf{A}_{T'} & I_{T'} & S_{T'} & J+1 & l_{J+1} & V_{T'1} & V_{T'2}
\end{array}\right]
=
\left[\begin{array}{c}
\text{column name} \\
\hline
\text{search (start basis)} \\
\hline
\text{search} \\
\text{search} \\
\vdots \\
\text{search (accepted basis)} \\
\hline
\text{interpolate} \\
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\vdots \\
\hline
\text{search} \\
\vdots \\
\text{search (final basis)} \\
\hline
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\text{search (no output)} \\
\vdots \\
\text{search (no output)}
\end{array}\right]
\tag{4}
$$

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

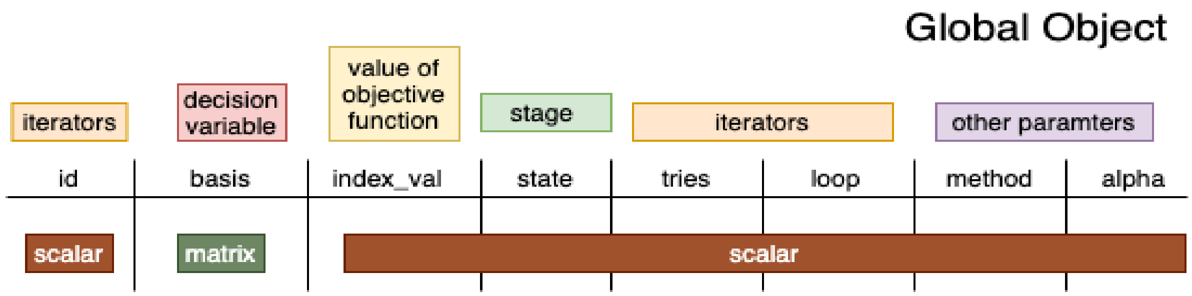

Figure 2: The data structure in projection pursuit guided tour.

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

It is worth noticing that the data structure constructed above meets the tidy data principle (Wickham et al. 2014) that states

1) each observation forms a row,
2) each variable forms a column, and
3) each type of observational unit forms a table

The wrangling and visualisation of tidy data have been greatly simplified by the well-known dplyr(Wickham et al. 2020) and ggplot2(Wickham 2016) package.

With a constructed data object, the construction of 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 does not 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.

4.2 Check how hard the optimiser is working

A primary interest of diagnosing an optimisation algorithm is to study how it progressively finds its optimum. One way of doing it is to plot the index value across its natural ordering t, however, it usually takes the algorithm much longer to find a better basis than the current one towards the end of the search. When plotting the searching observations as points in a plot, the space each iteration takes will be proportional to the number of points in that iteration.

Another option is to use summarisation for each iteration. Boxplot is a suitable candidate that provides five points summary of the data, however, there are two pieces of information missing from the boxplot: 1) It does not report the number of points, and 2) the position of the last basis. This could be remedied by adding more layers using the concept of grammar of graphics. A label geometry is added at the bottom of the plot to show the number of points in each iteration and a line geometry links the last basis in each iteration. Further, an option to switch between displaying points and boxplot geometry is helpful since point geometry can be more intuitive for the iteration with few observations. This is achieved via a `cutoff` parameter. Below we define the searching point plot based on four different component layers

- Layer 1: boxplot geom
  - data: group by j and filter the observations in the groups that have count greater than `cutoff = 15`.
  - x: j is mapped to the x-axis
  - y: the statistical transformed index value: $Q_{I'_t}(q)$ is mapped to the y-axis where $Q_X(q)$, $q = 0, 25, 50, 75, 100$ finds the qth-quantile of X and I'_t denotes the index value of all the searching bases defined in Matrix 4.
- 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

- Layer 4: label geom
    - data: A count table of the number of observation in each iteration
    - x: j is mapped to the x-axis
    - y: the y-axis for each label is $0.99 * \text{MIN}$, where MIN is the smallest index value in the whole data object
    - label: the number of observation is mapped to the label

Figure 3 presents a comparison between the two plotting options discussed above. Using the natural order to plot searching points distorts the point from the first few iterations and over-emphasizes the searches in the last few iterations. The second plot design evenly presents the summarised information for each iteration while allowing for a switch to the full information for the iteration with small number of observations.

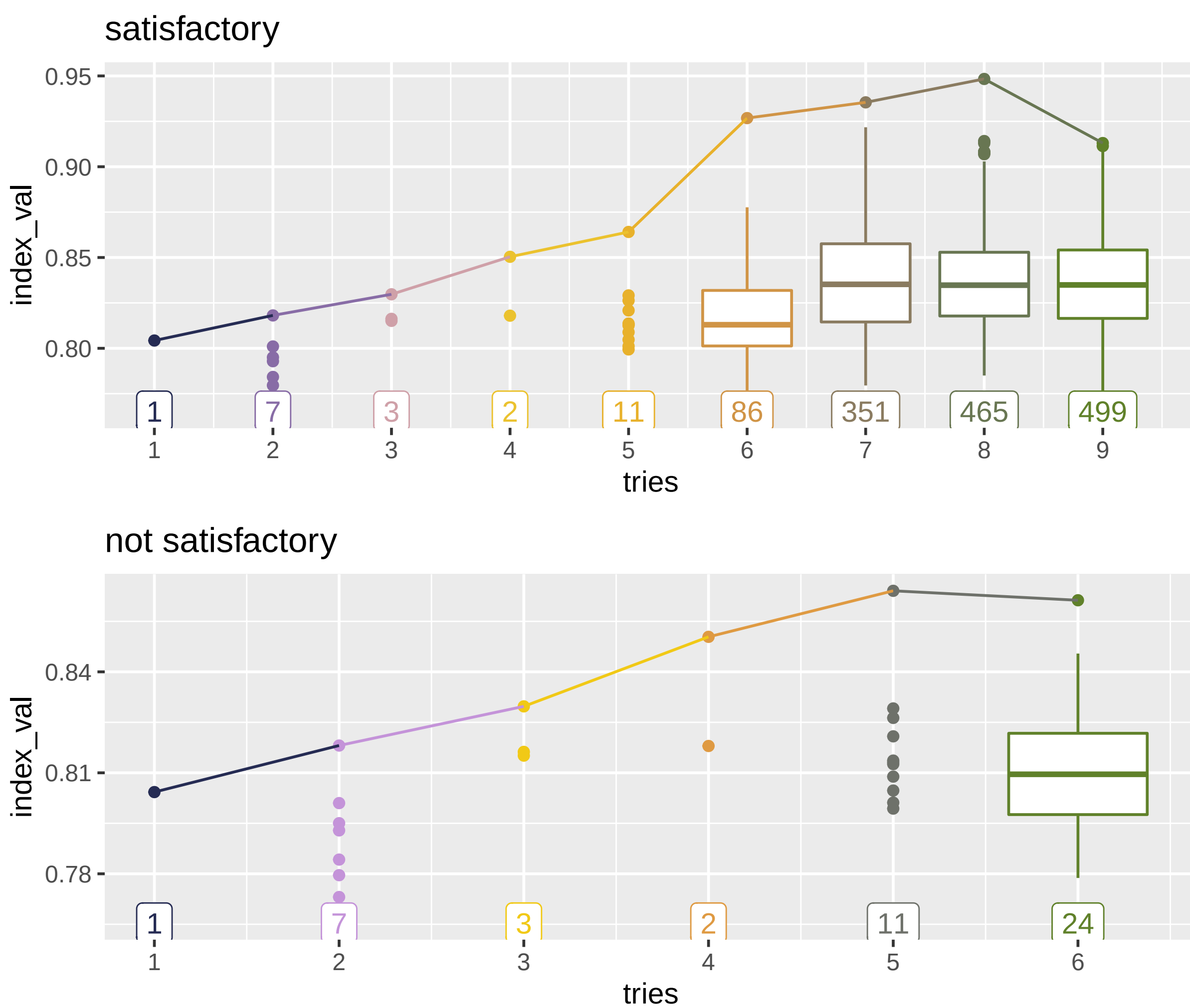

Figure 3: A comparison of plotting the same search points with different plot designs. In the upper plot, points from the last three iterations span the vast majority of the plotting space leaving the first few iterations being squeezed together whilst the lower plot spaces each iteration evenly and presents summary information easy to read.

4.3 Examining the optimisation progress

Sometimes, we may be interested in exploring the points on the interpolation path since these are the points that will be played by the tour animation. The plot definition is as follows:

- 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 4 presents the interpolation of three different tour paths. The upper plot shows a desirable interpolation in each iteration with the index value being progressively and monotonically increasing. While in the middle plot, the increases towards the target basis is not monotonical in the last two iterations and interpolated basis with higher index value can be found on the tour path. The lower plot is constructed using `search_better_random`, where a basis with smaller index value has a probabilistic chance of being accepted and we can observe a much more involved pattern. In iteration three, the probabilistic acceptance produces a monotonically decreasing interpolation whilst in iteration five, six and seven, with also a probabilistic acceptance, the interpolation is now non-monotonical. When the target basis has a higher index value, iteration eight reaches this basis by first decreasing the index value. While the interpolation plot shows different possibilities of the change in index value on the tour path, the diagnosis of the validity of each pattern and the related optimisation algorithms will be postponed to Section 5.

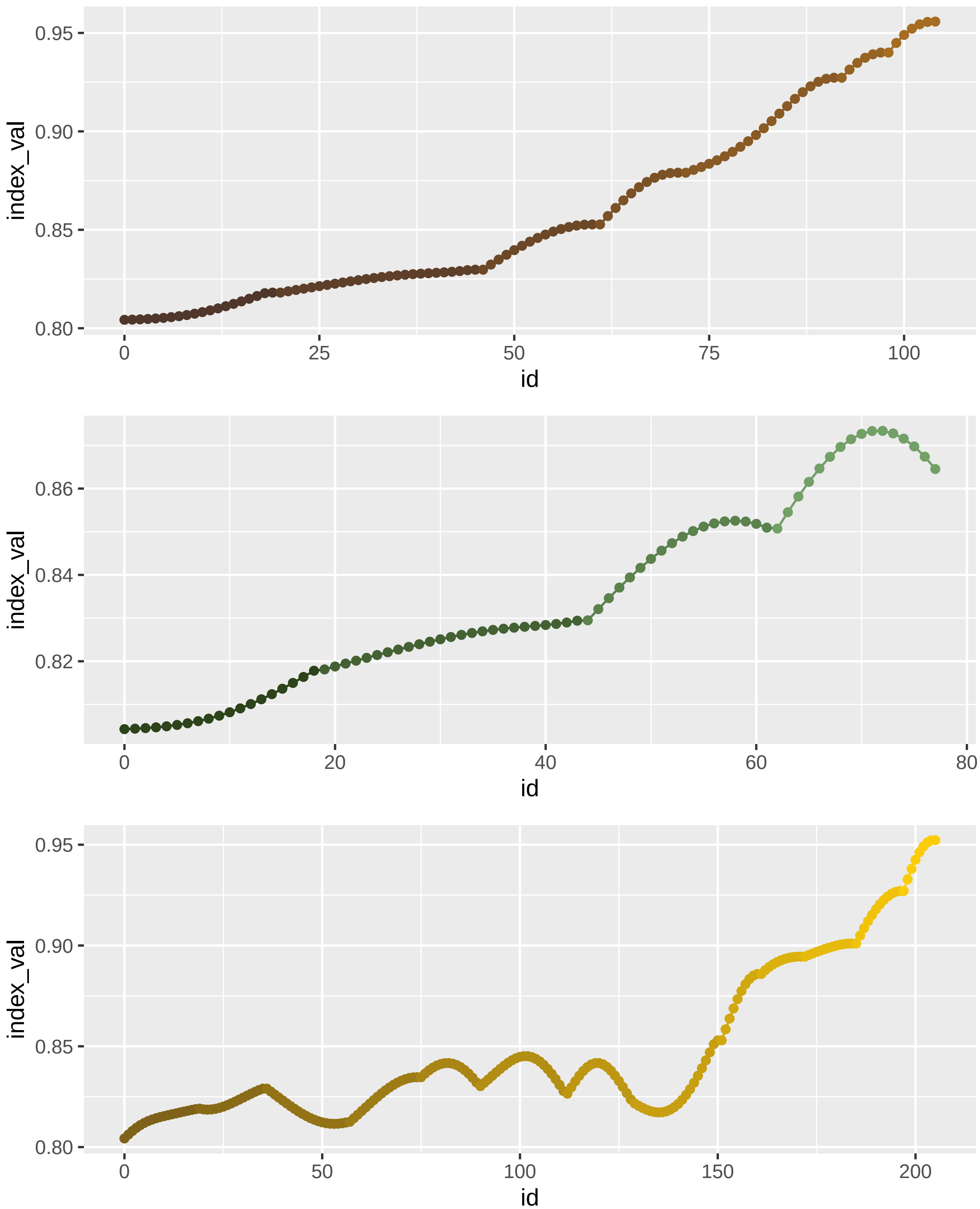

Figure 4: This is a toy example of plotting interpolated points

4.4 Understanding the optimiser's coverage of the search space

In projection pursuit, the projection bases $\mathbf{A}_{p\times d}$ are of dimension $p \times d$ with p usually greater than two 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 5 shows the searching space plot of a geodesic search, colored by different searching stages. Starting from the start basis, a directional search is conducted in a narrow neighbourhood on five random directions. The best one is picked and an exact search finds the stepsize that maximises the index value and outputs the corresponding projection basis.

4.5 Animating the diagnostic plots

all the plots above can be animated if wanted - importance is how the data is moving around

show an example of some snap shots of the animation and provide a link... for animated pca plot

4.6 The tour looking at itself

show the path of the tour on a sphere. (full data dimension animation...)

include the tour animation (geozoo)

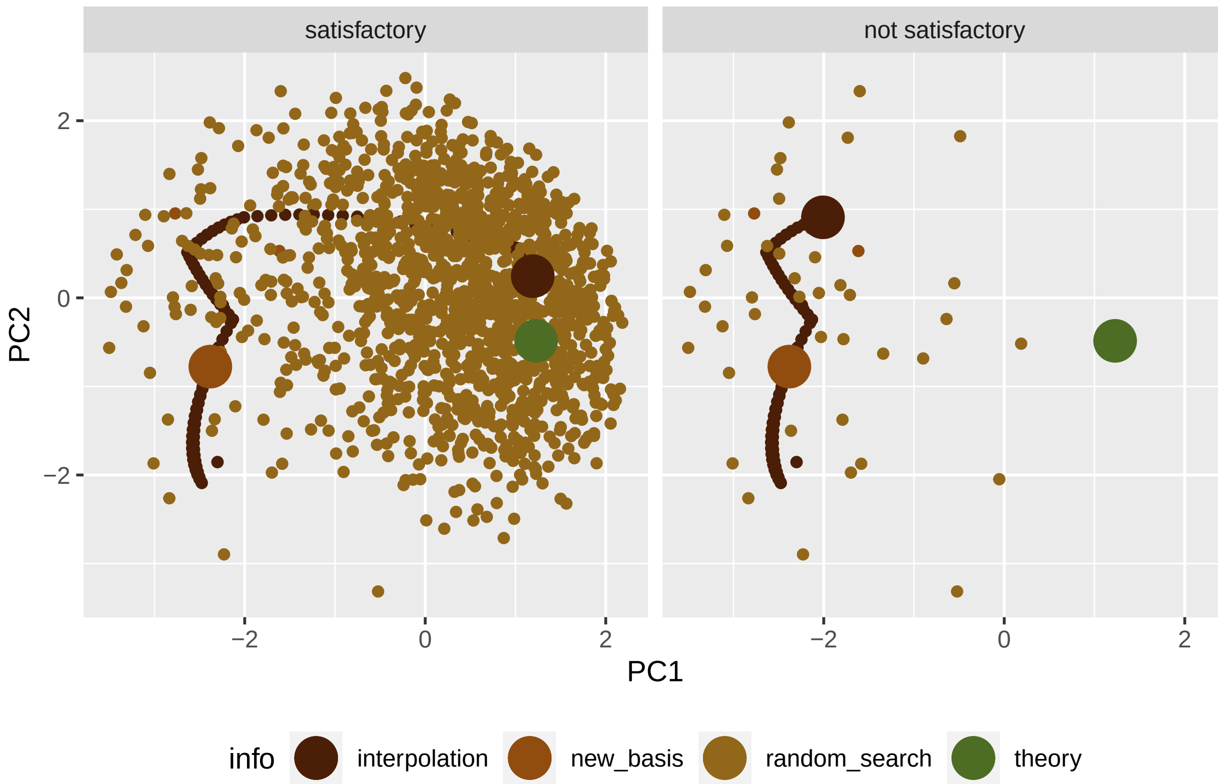

Figure 5: PCA plot of search geodesic colouring by info allows for better understanding of each stage in the geodesic search

5 Diagnosing an optimiser

The three algorithms introduced in Section 3 have experienced several practical issues in application. [initial problems with tourr optim, 2-3 sentences]. These issues can be studied and investigated using the diagnostic plots constructed above.

5.1 Simulation setup

Random variables with different structures has been simulated and the distribution of each is presented in Equation 5 to 11. Variable `x1`, `x8`, `x9` and `x10` are normal distributed with zero mean and unit variance and `x2` to `x7` are mixture of normal distributions with different weights and locations. Figure 6 plots the histogram of each variable except `x3`. The mixture variables are scaled to have unit variance before running the projection pursuit.

$$x_1 \overset{d}{=} x_8 \overset{d}{=} x_9 \overset{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{5}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{6}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{8}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{9}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{10}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{11}$$

5.2 A problem of not monotonic

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 the `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 a higher index value than the target basis on the interpolation path. In this sense,

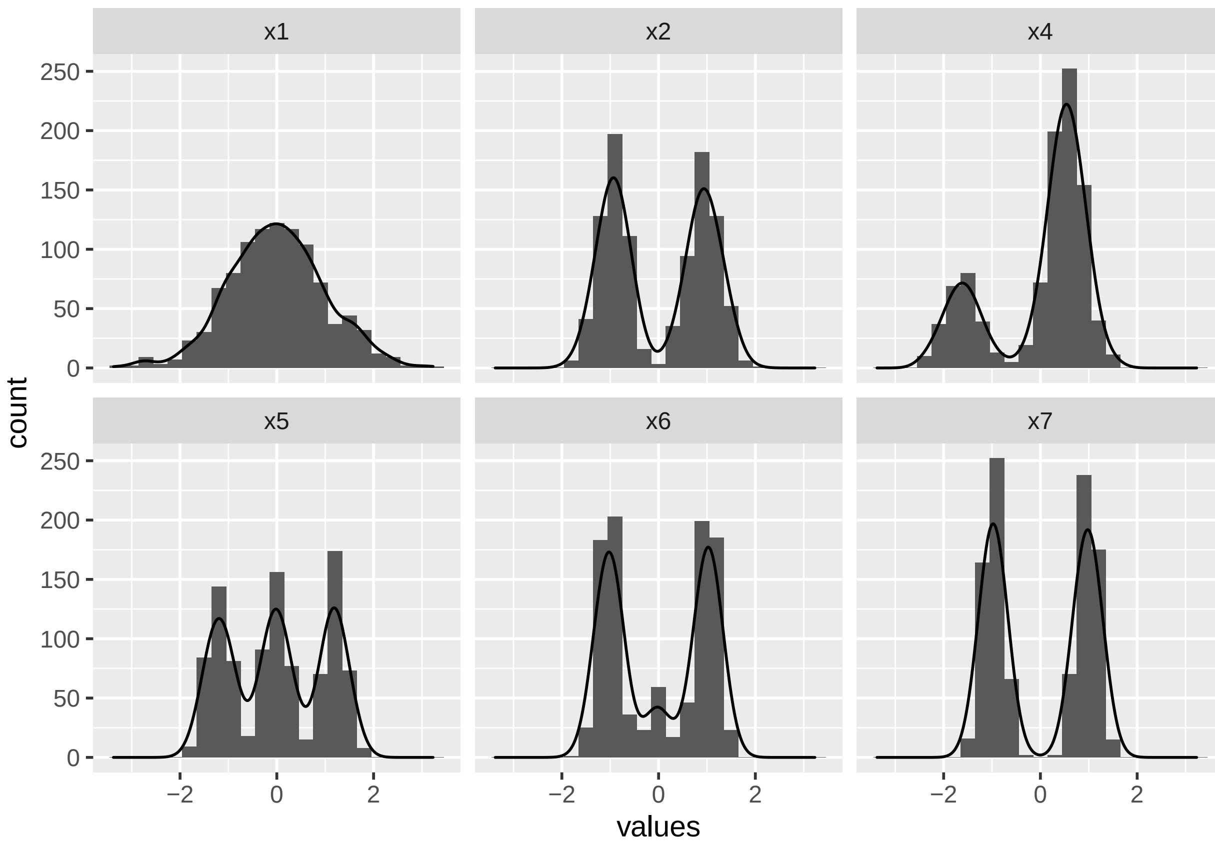

Figure 6: The histogram of simulated selected variables.

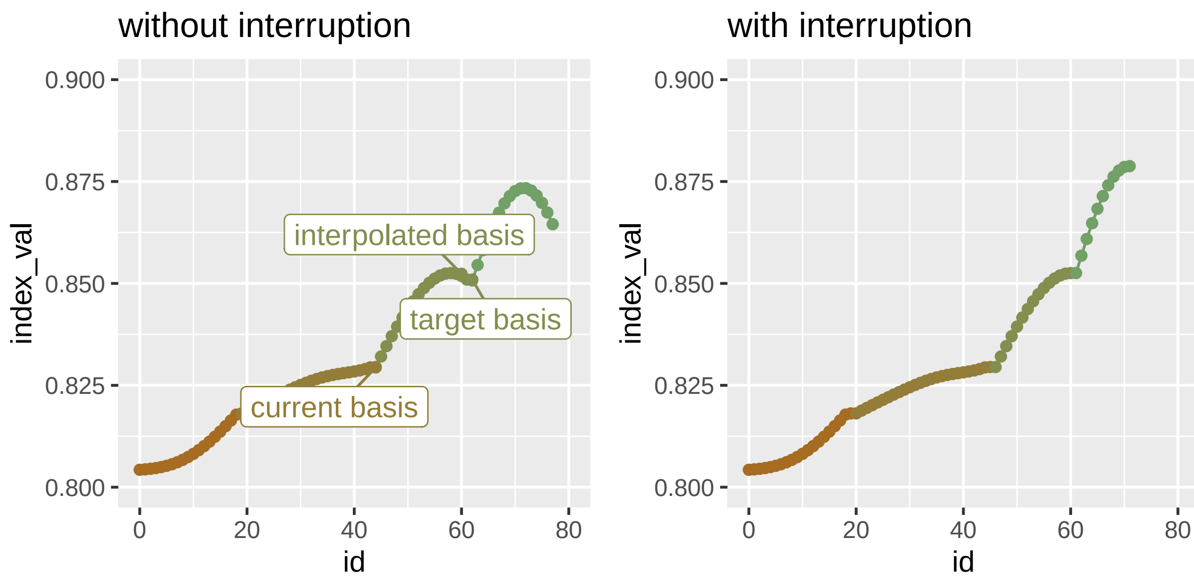

Figure 7: Trace plots of the interpolated basis with and without interruption. The interruption stops the interpolation when the index value starts to decrease at id = 60. The implementation of the interuption finds an ending basis with higher index value using fewer steps.

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 7.

5.3 Close but not close enough

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 `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)
```

A slight variation of the 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 data 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

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 sufficiently explore the parameter space.

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]

5.4 Seeing the signal in the noise

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 in-

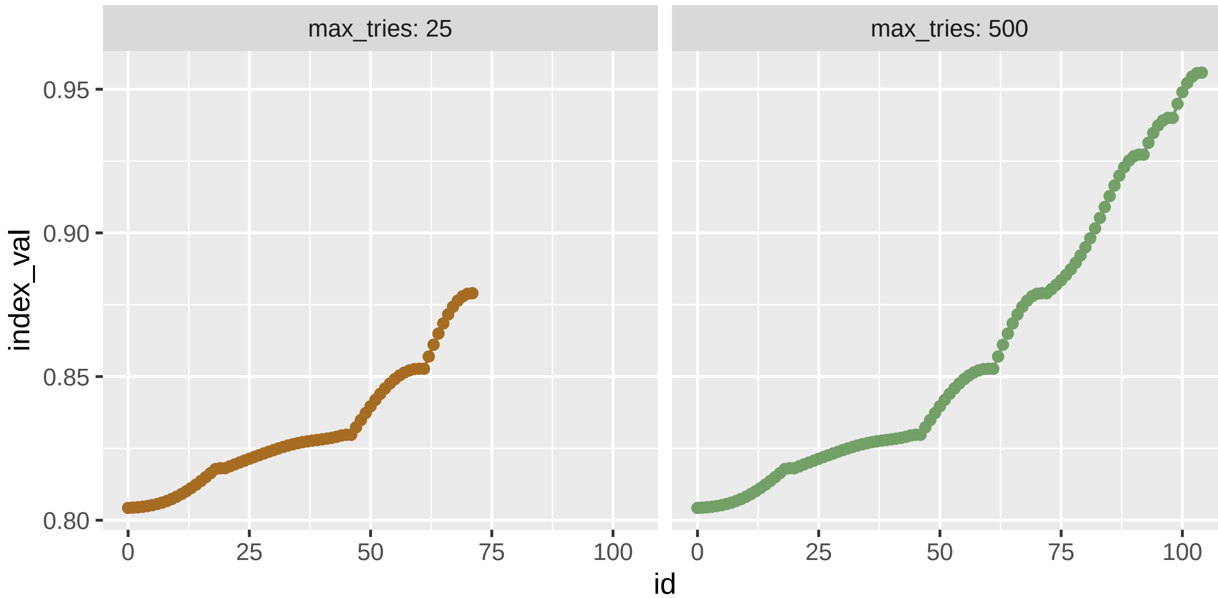

Figure 8: Breakdown of index value when using different max.tries in search better in conjunction with search polish. Both attain the same final index value after the polishing while using a `max.tries` of 25 is not sufficient to find the true maximum.

dex 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.

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

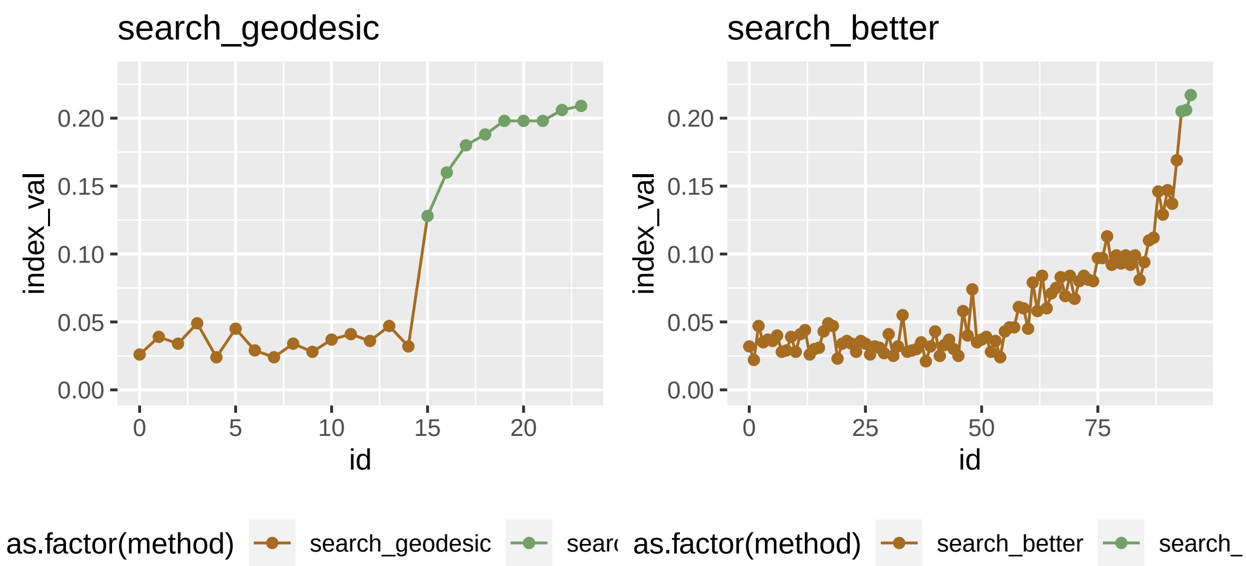

Figure 9: Comparison of two different searching methods: search_geodesic and search_better on 1D projection problem for a noisier index: kol_cdf. The geodesic search rely heavily on the polishing step to find the final index value while search better works well.

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 maxima appear with 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 maxima d 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 a Metropolis-hasting random search as shown in Figure 11, although at a higher cost of number of points to evaluate.

We can also plot the searching points of all three algorithms in the searching space and explore their relative position against each other using principal components. As shown in Figure **??**, 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 find x2 in the end.

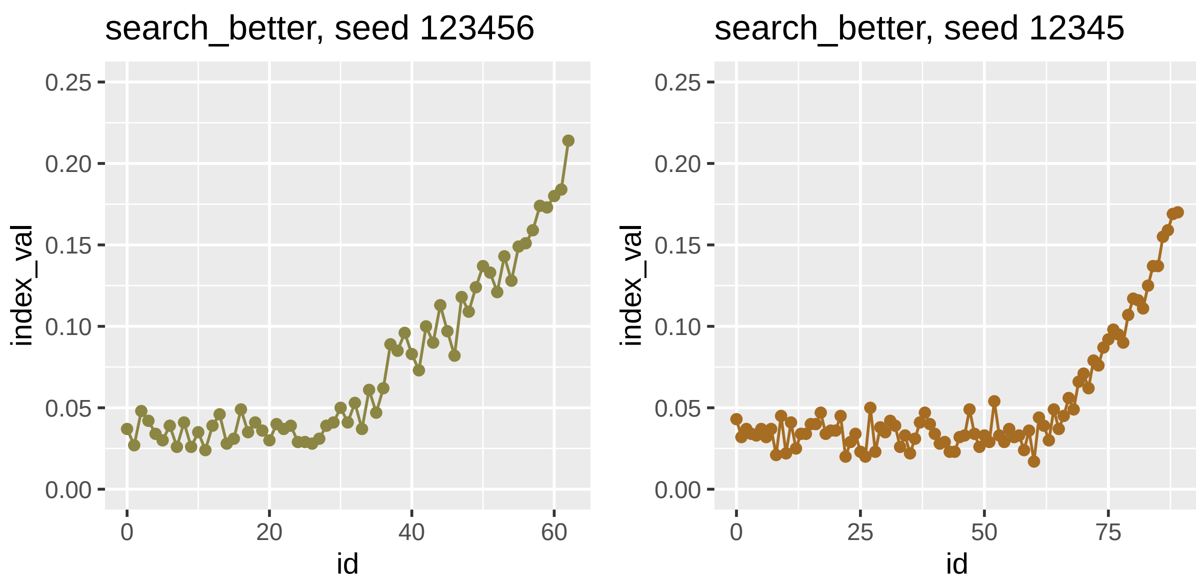

Figure 10: The trace plot search better in a 1D projection problem with two informative variables using different seeds (without polishing). Since there are two informative variables, setting different value for seed will lead search better to find either of the local maximum.

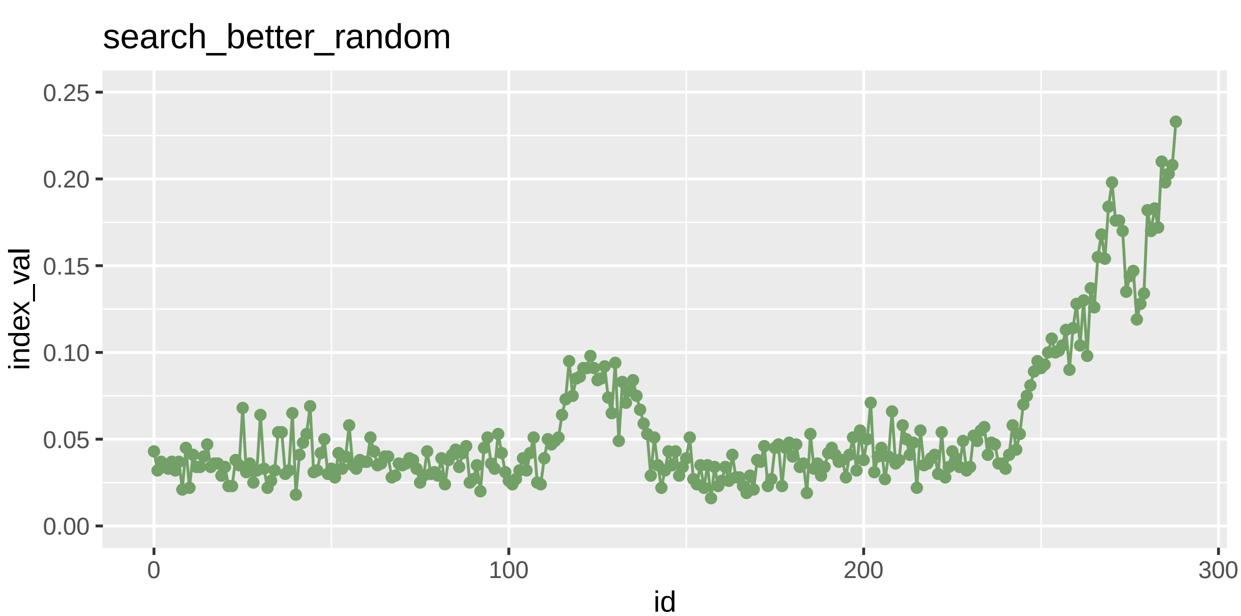

Figure 11: Using search better random for the problem above will result in finding the global maximum but much larger number of iteration is needed.

6 Implementation: Ferrn package

Everything is coded up in a package. Package structure

When the optimisation ends, the data 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`. Notice that the tibble object allows the list-column `basis` to be printed out nicely with the dimension of the projection basis readily available.

We form our first dataset using variables `x1`, `x2`, `x8`, `x9` and `x10`, run the guided tour with optimiser `search_better` and below shows the first ten rows of the data object.

```
## # 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
```

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.

Cook, D., Buja, A., Lee, E.-K. & Wickham, H. (2008), Grand tours, projection pursuit guided tours, and manual controls, *in* 'Handbook of data visualization', Springer, pp. 295–314.

Curry, H. B. (1944), 'The method of steepest descent for non-linear minimization problems', *Quarterly of Applied Mathematics* **2**(3), 258–261.

Fletcher, R. (2013), *Practical methods of optimization*, John Wiley & Sons.

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.

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., François, R., Henry, L. & Müller, K. (2020), *dplyr: A Grammar of Data Manipulation*. R package version 1.0.0.
URL: *https://CRAN.R-project.org/package=dplyr*

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