Title here

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

June 9, 2020

Abstract

Keywords: 3 to 6 keywords, that do not appear in the title

*The authors gratefully acknowledge ...

1 Introduction

1.1 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. The two most common methods in DFO are *direct-search methods* and *model-based methods* and this paper dedicates to the discussion of direct-search methods, which gains its popularity due to its simplicity in use and reliability in complicated practical problem. Coined by Hooke & Jeeves (1961), direct search methods don't require any gradient or Hessian information and evaluate f directly. A well-known example of it is the Nelder-Mead algorithm (Nelder & Mead 1965) and it enjoys the popularity due to its simplicity and reliability. [feel like this sentence can be expanded to include more information. xxx]. Further, direct search methods can be classified as *stochastic* and *deterministic* depends on whether a random sample is used in the search. The stochastic version of direct-search method is the focus of this paper.

1.2 Projection pursuit guided tour

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. A Projection Pursuit Index (PPI) is a function of the projection matrix (a.k.a projection angle, projection basis) measuring the "interestingness" of data and we refer the value of PPI function as *index value*. In the literature, the indices being proposed include 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). [Any literature on holes index? xxx]

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*, like movie frames in real time. Starting from a randomly generated projection basis, projection pursuit searches for a target basis with higher index value and guided tour interpolates geodesically between the initial basis and the target basis. This series of bases forms a tour path and the projection of the data on these bases is played in real time. This search-and-interpolate process is repeated until no basis can be found with higher index value. Modified from Buja et al. (2005), Figure 1 vividly depicts the tour path in guided tour. 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).

[doubt if I need the following. xxx]

1.3 Optimisation in projection pursuit

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.

[thinking if the following three should be presented as algorithms or plain description is fine. xxx]

There are three existing methods for optimisating PPI function and we review them below. Posse (1995) used a stochastic direct search method, a random search algorithm to

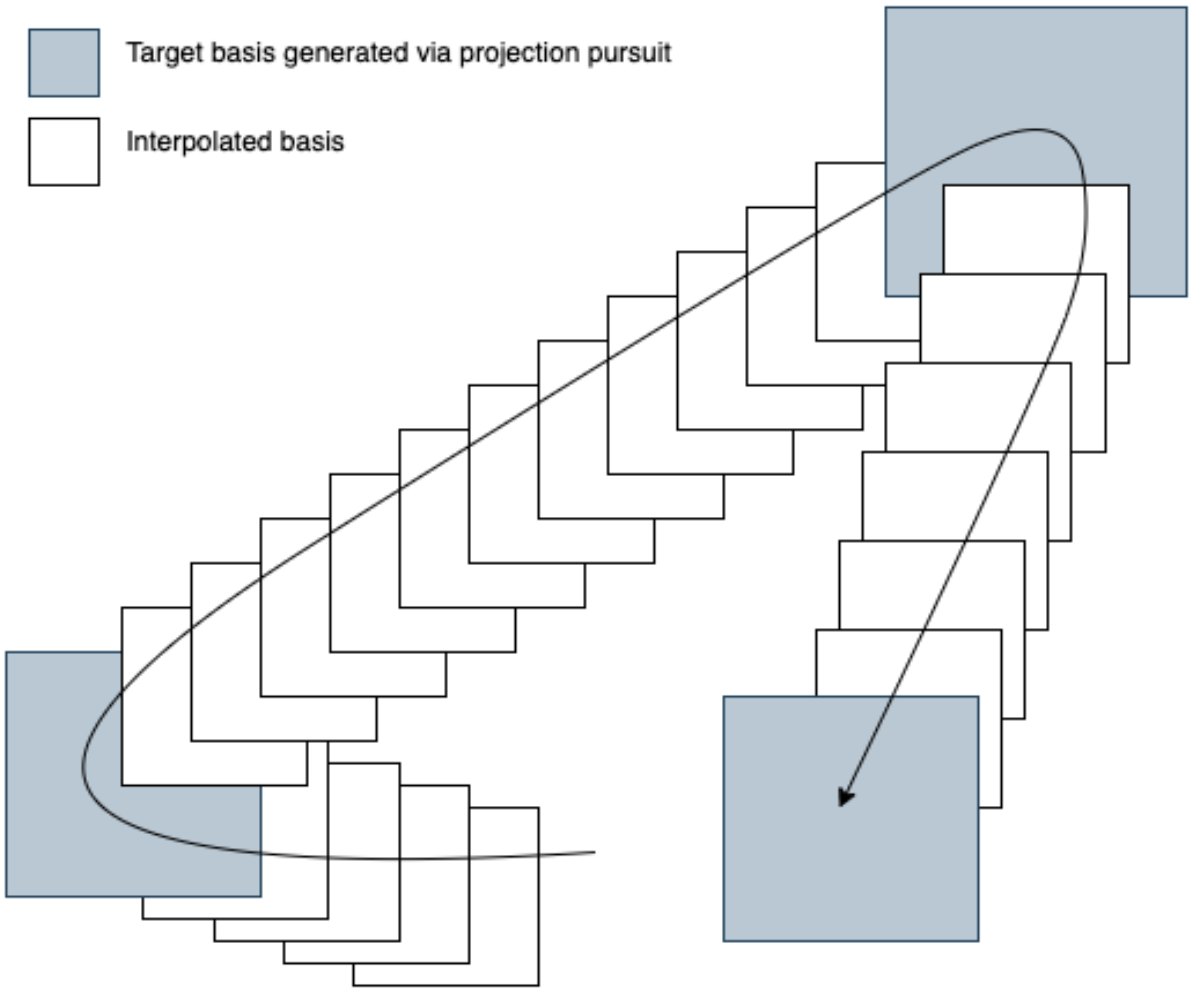

Figure 1: An illustration of the tour path

sample new bases in the neighbourhood of the current basis defined by the radius of the p-dimensional sphere, c. The new basis is taken as the target basis if it has higher index value, or the sampling continues. If no basis is found to have higher index value after a certain number of tries n, the radius c is halved. The algorithm stops when the maximum number of iteration is attained or the radius c is less than a pre-determined number.

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.

Simulated annealing (Bertsimas et al. 1993, Kirkpatrick et al. (1983)) is a non-derivative procedure based on a non-increasing cooling scheme $T(i)$. Given an initial T_0, the temperature at iteration i is defined as $T(i) = \frac{T_0}{log(i+1)}$. The simulated annealing algorithm works as follows. Given a neighbourhood parameter α and a randomly generated orthonormal basis B, a candidate basis is constructed as $B_j = (1-\alpha)B_i + \alpha B$ where B_i is the current basis. 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 $A = \min\left(\exp(-\frac{I(B_j)-I(B_i)}{T(i)}), 1\right)$ where $I(.)$ is the index function.

1.4 problems and difficulties in PP optimisation

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

2 Visual diagnostic system

2.1 Visual diagnostics

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 creates aesthetic mapping from the variables to the geometric objects and builds the plot through different layers.

[Do I need this paragraph to details the grammar of graphic? xxx]

2.2 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* and the *value of the objective function*. *iterators* indexes the data collected and this order has time series feature that prescribes the progression of the optimisation. Given the complexity of the algorithm, there could be multiple iterators. There are other parameters that can't be classified as one of the three categories but are also of our interest. They are defined under the fourth category:

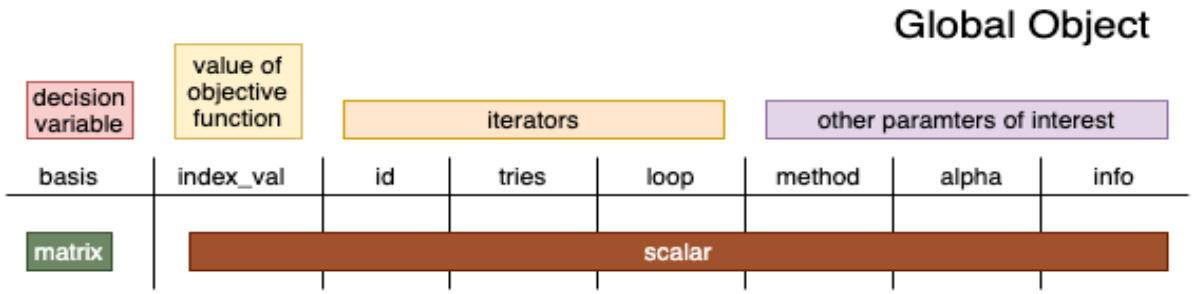

Figure 2: The global object in projection pursuit guided tour.

other parameter of interest. This could include variables that are specific for one particular algorithm. Below we show an example of defining the global object of projection pursuit guided tour.

In the projection pursuit guided tour, the decision variable is the projection matrix, the value of objective function is the index value. There are three iterators: `id` is the smallest ordering unit that increases by one for each observation; `tries` is updated once a search-and-interpolate step is finished. A third iterator `loop` is used to record the number of repetition in the search step and starts over from one at the beginning of a new `tries`. 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 2.

3 Diagnostics plots

3.1 Simulated data

Two set of simulated data are used in the demo of the visual diagnostics in projection pursuit guided tour. A small dataset consists of 1000 randomly simulated observations

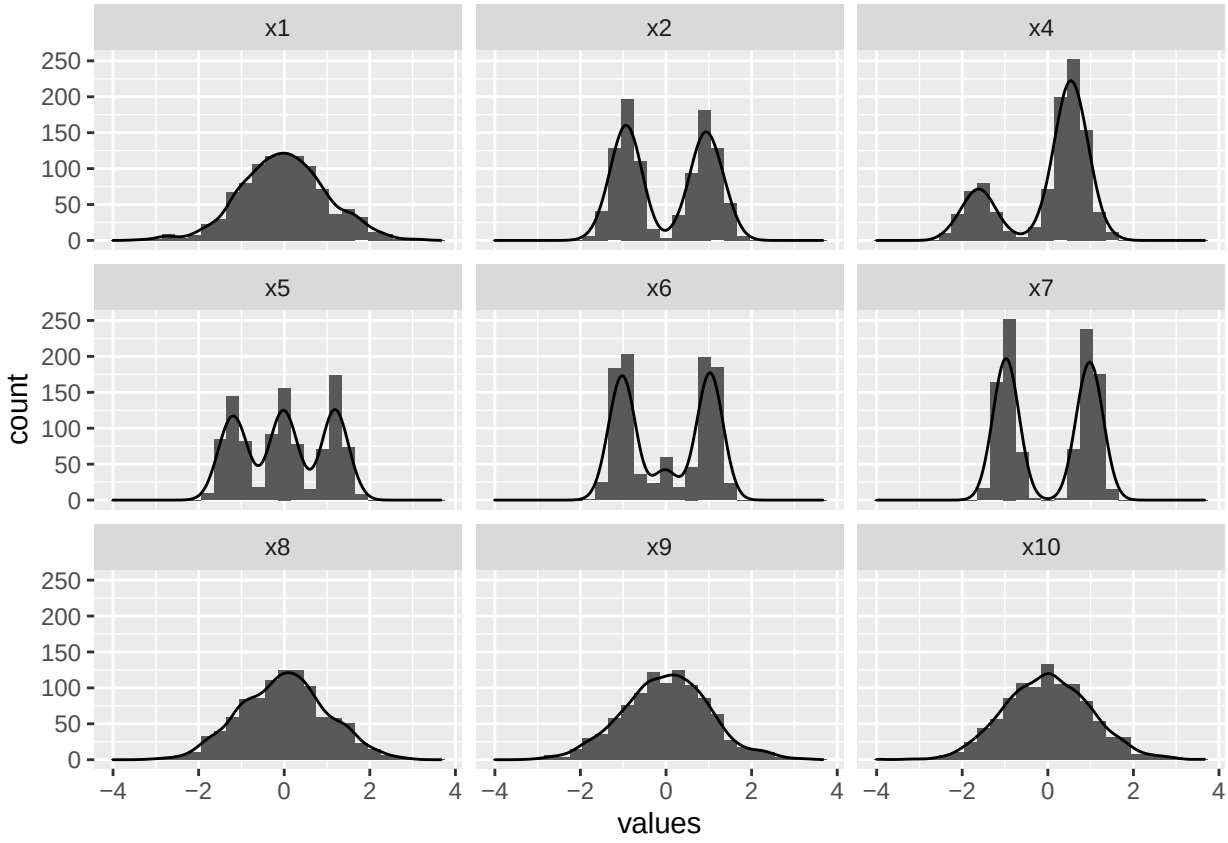

Figure 3: The distribution of simulated data except x3

of five variables (`x1`, `x2`, `x8`, `x9`, `x10`). `x2` is the informative variable simulated from two bi-modal normal distribution centred at -3 and 3 with variance of one and the other four are simulated from non-informative standard random normal distributions. The data has been scaled to ensure `x2` has variance of 1. The goal is to find the 1D projection of bi-modal shape, which corresponds to the projection matrix (vector) of `[0, 1, 0, 0, 0]`.

A larger dataset contains more informative variables (`x3` to `x7`) of different types. The distribution of all the variables except `x3` is plotted in Figure 3 and `x3` takes 500 positive one and 500 negative one. [should I introduce the dist for each var? xxx] [also feel like we don't really use `x3` to `x6`, should we not mention about these? xxx]

In tour, when the optimisation ends the global object will be stored and printed and can be turned off via `print = FALSE`. Additional messages during the optimisation can be displayed via `verbose = TRUE`. Below presented the global object of the 1D projection

using the small dataset. [the printing is not ideal as it doesn't show all the columns. xxx]

```
## # 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 ~     0.749     1 start             NA <NA>            0.5     1
## 2 <dbl[,1] [5 x ~     0.749     1 direction_sea~     1 search_geode~  NA       2
## 3 <dbl[,1] [5 x ~     0.749     1 direction_sea~     1 search_geode~  NA       3
## 4 <dbl[,1] [5 x ~     0.749     1 direction_sea~     1 search_geode~  NA       4
## 5 <dbl[,1] [5 x ~     0.749     1 direction_sea~     1 search_geode~  NA       5
```

3.2 Explore scalar parameters

In tour, all the points recorded in the global object can be divided into two broad categories: searching points and interpolating points

- *Searching points* include the observations recorded in the searching algorithm to find the target basis. The points for target bases is also included in the searching points and there is one such point per `tries`.
- *interpolating points* exist in the guided tour to produce continuous animated view from one target basis to another and it doesn't have `loop` value.

3.2.1 Explore searching points

The largest difficulties of exploring searching points is its unknown number of observations per `tries`. Mapping `id` on the x-axis will leave the `tries` with few observations a small space in the plot, while those `tries` with large number of search points (usually towards the end) occupy the vast majority of the plot space. This motivates to summarise the points in each `tries`. At each iteration, rather than knowing the index value of *every* points, we are more interested to know a general summary of all the points and more importantly, the point with the largest `index_val` since it prescribes the geodesic interpolation and future searches.

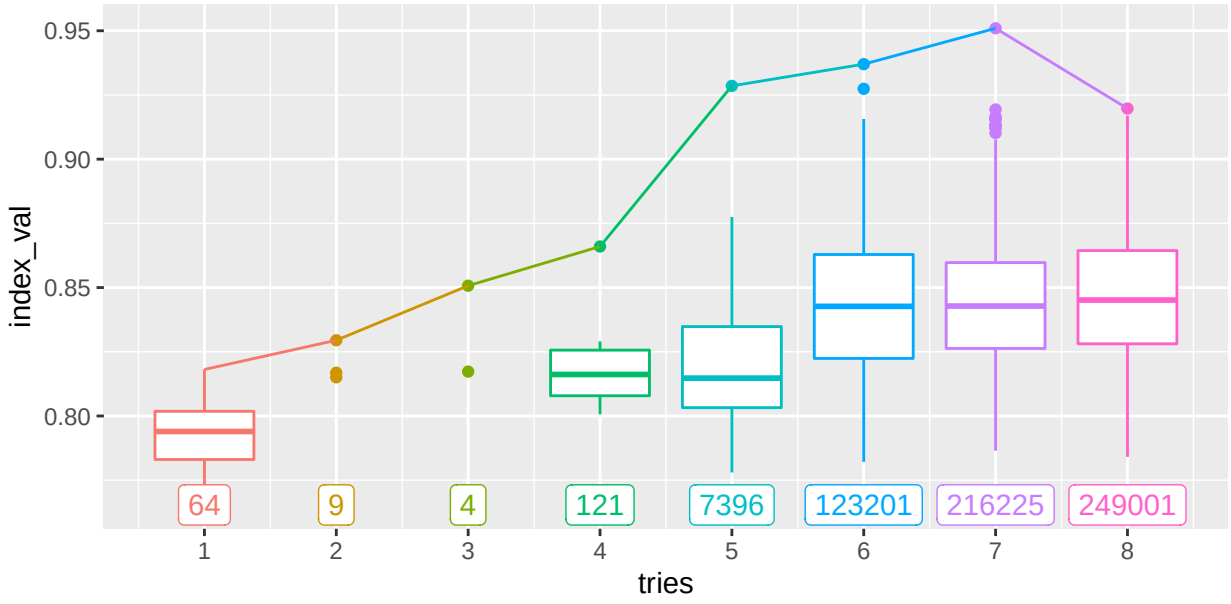

Figure 4: A comparison of plotting the same search points with different plot designs. The left plot doesn't efficiently use the plot space to convey information from the plot while the right plot provides good summarisation of data and number of points in each tries.

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 for each `tries` is displayed at the bottom of each box and we provide options to switch `tries` with small number of points to a point geometry, which is achieved via the `cutoff` argument. A line geometry is also added to link the points with the largest index value in each `tries`. This helps to visualise the improvement made by each `tries`.

Example: exploring searching points The larger dataset is used for 2D projection with `search_better` and `max.tries = 500`. In Figure 4, the searching points with `id` and `tries` on the x-axis, colored by `tries`. Label at the bottom indicates the number of observations in each tries and facilitates the choice of cutoff to switch from point geometry to boxplot geometry (`cutoff = 15`). The line geometry suggests the largest improvement happens at `tries = 5`.

3.2.2 Explore interpolating points

Plotting the interpolating points as time series data allows us to diagnose characteristics of different configurations and index functions. Here we present two examples of using plots to diagnose tour algorithms and different index functions.

Example: Interruption This examples uses `search_better` for a 2D projection on the larger dataset using the `holes` index. In the interpolation stage, continuous frames will be constructed to bridge the project on the current basis to the target basis. The target basis will become the current basis in the next iteration and the searching stage continues to find the next target basis. In the left panel of Figure 5, we observe that there are interpolating bases with higher index values than the target basis and these bases could be used to search for new basis in the next searching stage.

An interruption is then constructed to accept the interpolating bases up to the one with the largest index value and use that basis as the current basis for the next searching stage. After implementing this interruption, the tracing plot with the same configuration is shown on the right panel of Figure 5. In the third `tries`, rather than interpolating fully to the target basis at `id = 62`, it stops before the index value starts to decrease at `id = 60`. This implementation results in a higher index value in the end with fewer steps.

Example: Noisy index function The interpolation path of holes index, as seen in Figure 5, is smooth, while this may not be the case for more complicated index functions. `kol_cdf` index, an index function based on Kolmogorov test, compares the difference between a projection matrix and a randomly generated normal distribution based on cumulated distribution function (CDF). Several diagnostic visualisations below show the characteristic of this index function.

Figure 6 compares the tracing plot of the interpolating points for `search_geodesic` and `search_better` on 1D projection for `kol_cdf` index and the interpolation path shows a zig-zag pattern. Polishing step has done much more work to reach the final index value for `search_geodesic` than `search_better` and this indicates that noisy index's favour of a random search method than ascent method [the order of the examples needs to be rearranged since I haven't introduced polishing here. xxx].

The second example for `kol_cdf` index uses 1D projection on the larger dataset. Since

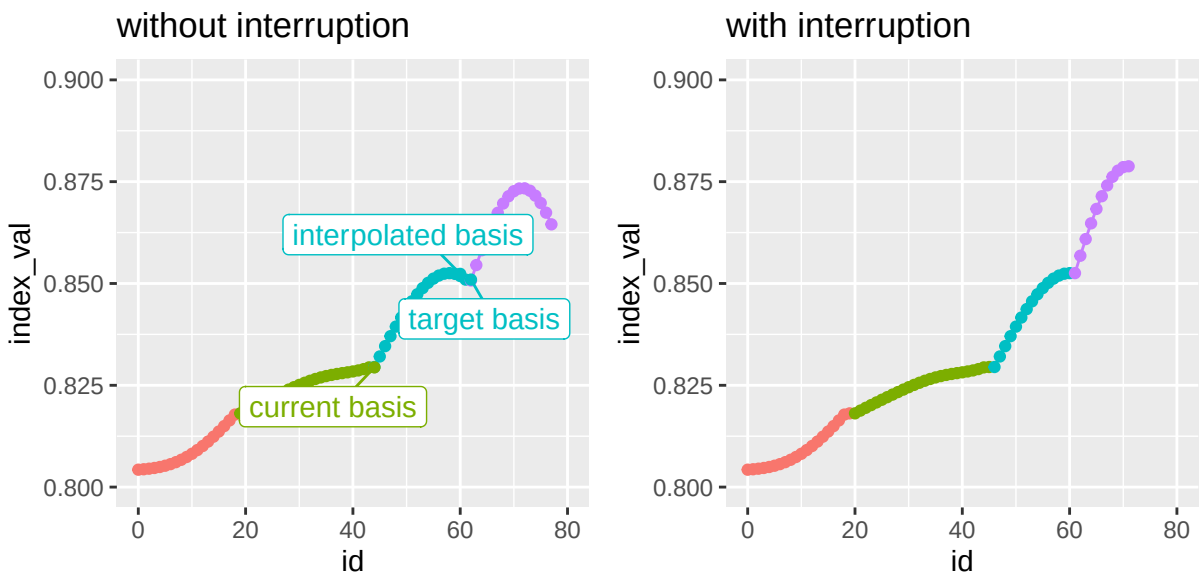

Figure 5: Trace plots of the interpolated basis with and without the interruption. The interruption stops the interpolation when the index value starts to decrease at id = 60. The implementation of the interuption finds an ending basis with higher index value using fewer steps.

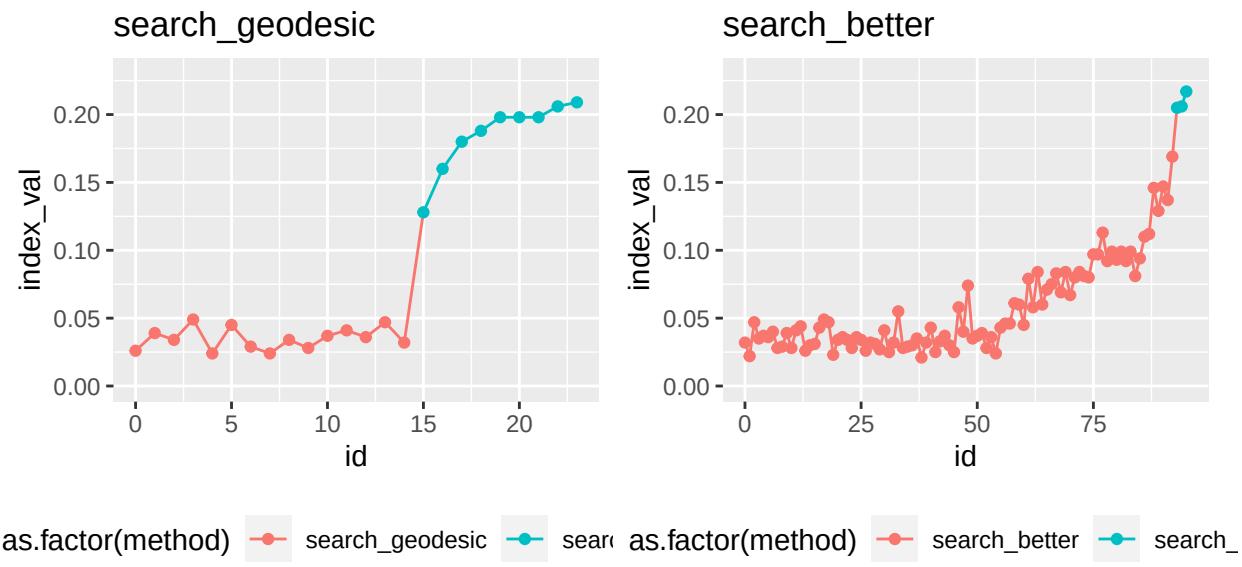

Figure 6: Comparison of two different searching methods: search_geodesic and search_better on 1D projection problem for a noisier index: kol_cdf. The geodesic search rely heavily on the polishing step to find the final index value while search better works well.

there are two informative variables and the projection is 1D, there are two local maximum when the projection matrix at $[0, 1, 0, 0, 0, 0]$ and $[0, 0, 1, 0, 0, 0]$. As in Figure 7, different local maximum can be found using `search_better` with different seeds while `search_better_random` can always find the global maximum, as in Figure 8, although at a high cost of number of points evaluated.

3.2.3 Adding more variables to the mapping

Example: Polish In principle, all the optimisation routines should result in the same output on the same problem while this may not be true in real application. This motivates the creation of a polishing search that polishes the ending basis and achieves unity on different methods.

`search_polish` takes the ending basis of a given search as the current basis and uses a brutal-force approach to sample a large number of basis (`n_sample`) in the neighbourhood, whose radius is controlled by `polish_alpha`. Among the `n_sample` basis, the one with the largest index value becomes the candidate basis. If its index value of the candidate basis is

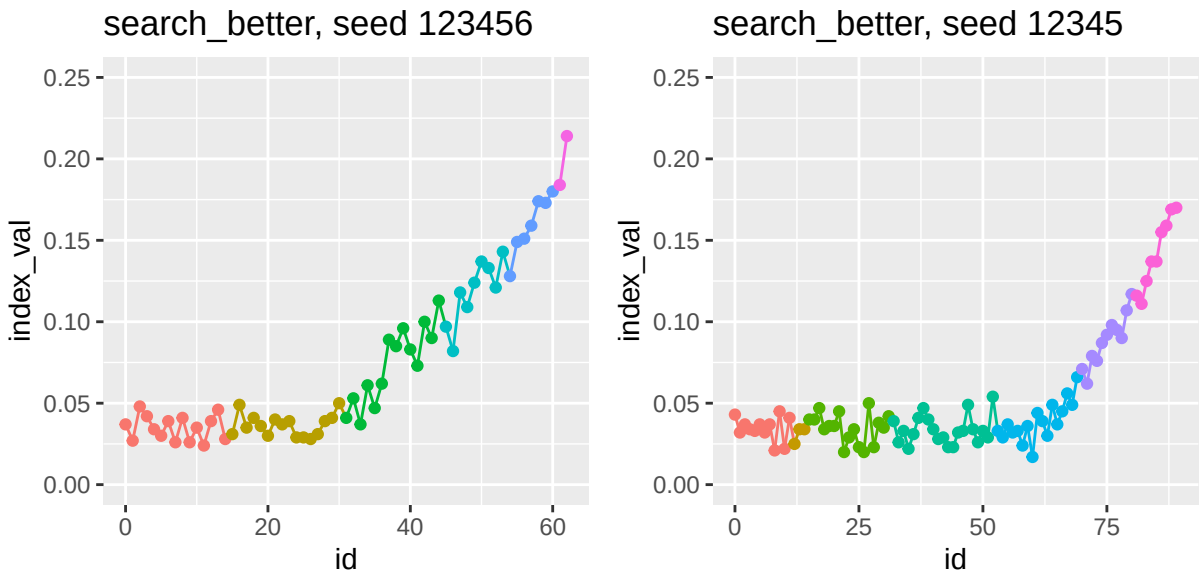

Figure 7: The trace plot search better in a 1D projection problem with two informative variables using different seeds (without polishing). Since there are two informative variables, setting different value for seed will lead search better to find either of the local maximum.

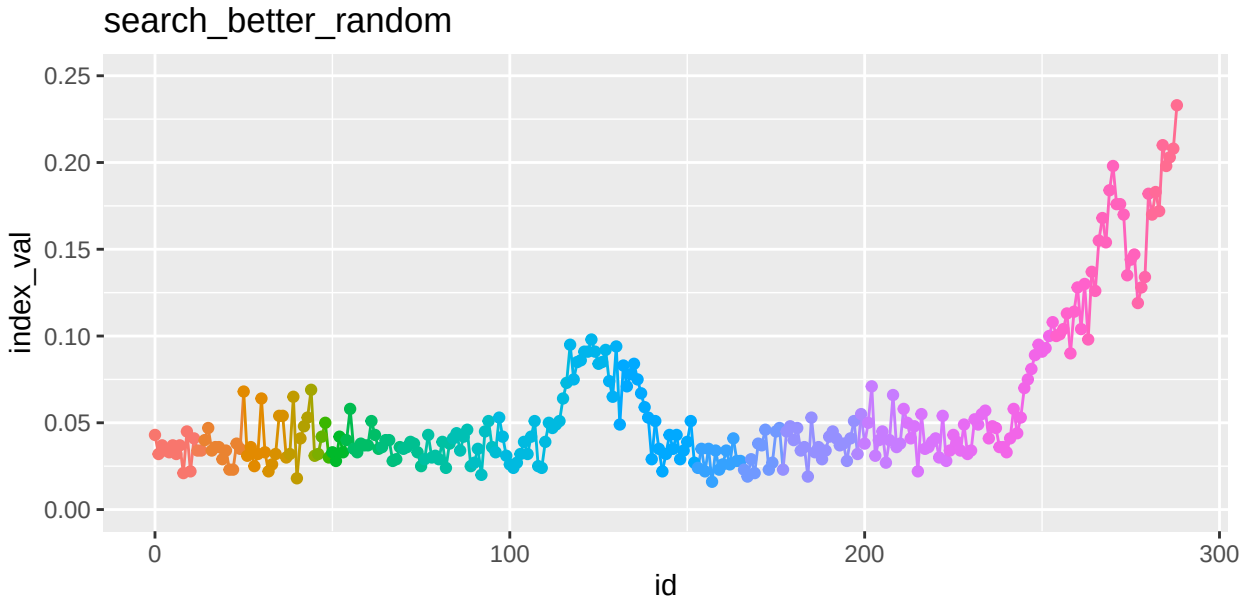

Figure 8: Using search better random for the problem above will result in finding the global maximum but much larger number of iteration is needed.

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 than the current basis, the searching neighbourhood will be shrunk and the search continues. The polishing search ends when one of the four stopping criteria is satisfied:

1) the two basis can't be too close
2) the percentage improvement of the index function 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`

[should the stopping criteria be more detailed? xxx]

The usage of search_polish is as follows. After the first tour, the final basis from the interpolation is extracted and supplied into a new tour with the `start` argument and `search_polish` as the searching function in the guided_tour. 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)
```

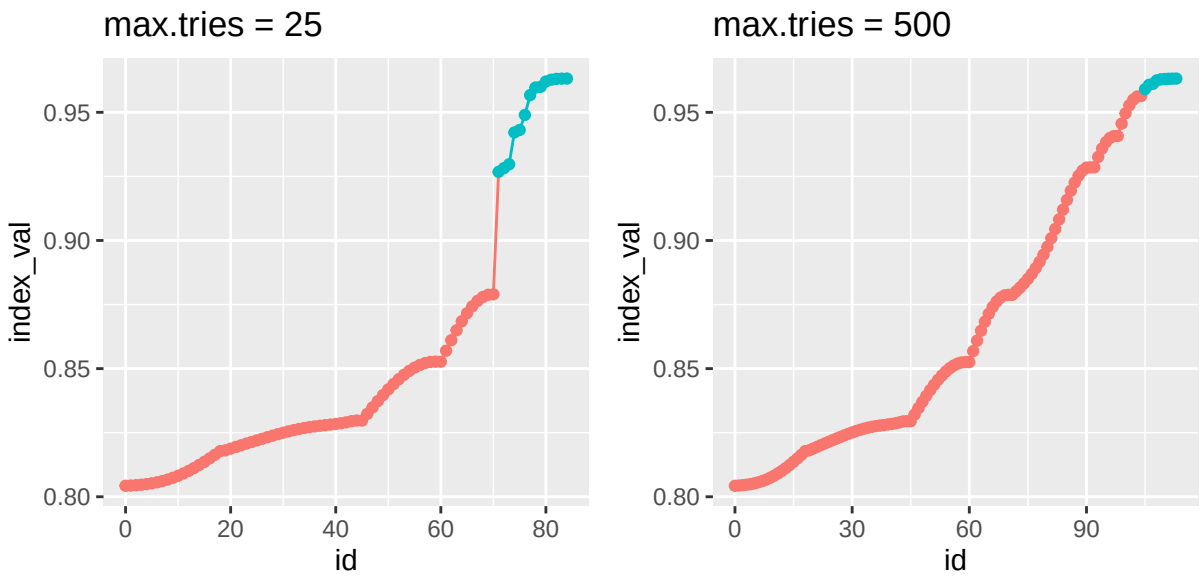

Figure 9: Breakdown of index value when using different max.tries in search better in conjunction with search polish. Both attain the same final index value after the polishing while using a max.tries 25 is not sufficient to find the ture maximum.

The following example conducted a 2D projection on the larger dataset using search better with different configurations. `max.tries` is a hyperparameter that controls the maximum number of try without improvement and its default value is 25. As shown in Figure 9, both trials attain the same index value after polishing while a small `max.tries` of 25 is not sufficient for `search_better` to find the global maximum. The `max.tries` argument needs to be adjusted to ensure it is sufficient to explore the parameter space.

3.3 Explore matrix parameter

The parameter to explore can be a vector or matrix instead of scalar, for example, the projection basis in tour. This imposes difficulties in visualisation since humans are bounded to perceive at most three dimensions in a plot. Thus, principal component analysis is used to reduce the dimension of projection bases and the first two principal components are mapped to the x and y axis of the plot. Additional variable of interest could be mapped to the color aesthetics for exploration.

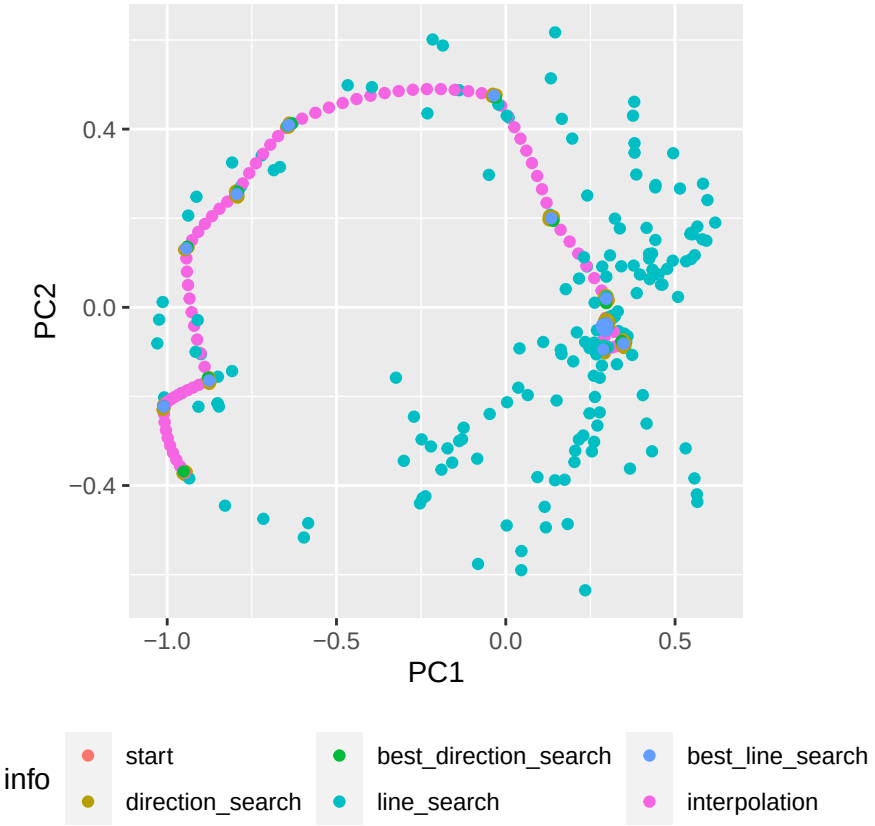

Figure 10: PCA plot of search geodesic Coloring by info allows for better understanding of each stage in the geodesic search

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.

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 11 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 11 shows a more desirable concentrated searching space near the ending basis. Both specifications of initial value allow the searches to reach the same ending index values.

4 Animated diagnostic plots

```
holes_1d_geo %>% explore_proj_pca(animate = TRUE, col = info) +
  theme(legend.position = "bottom")
```

5 Vis package

Everything is coded up in a package.

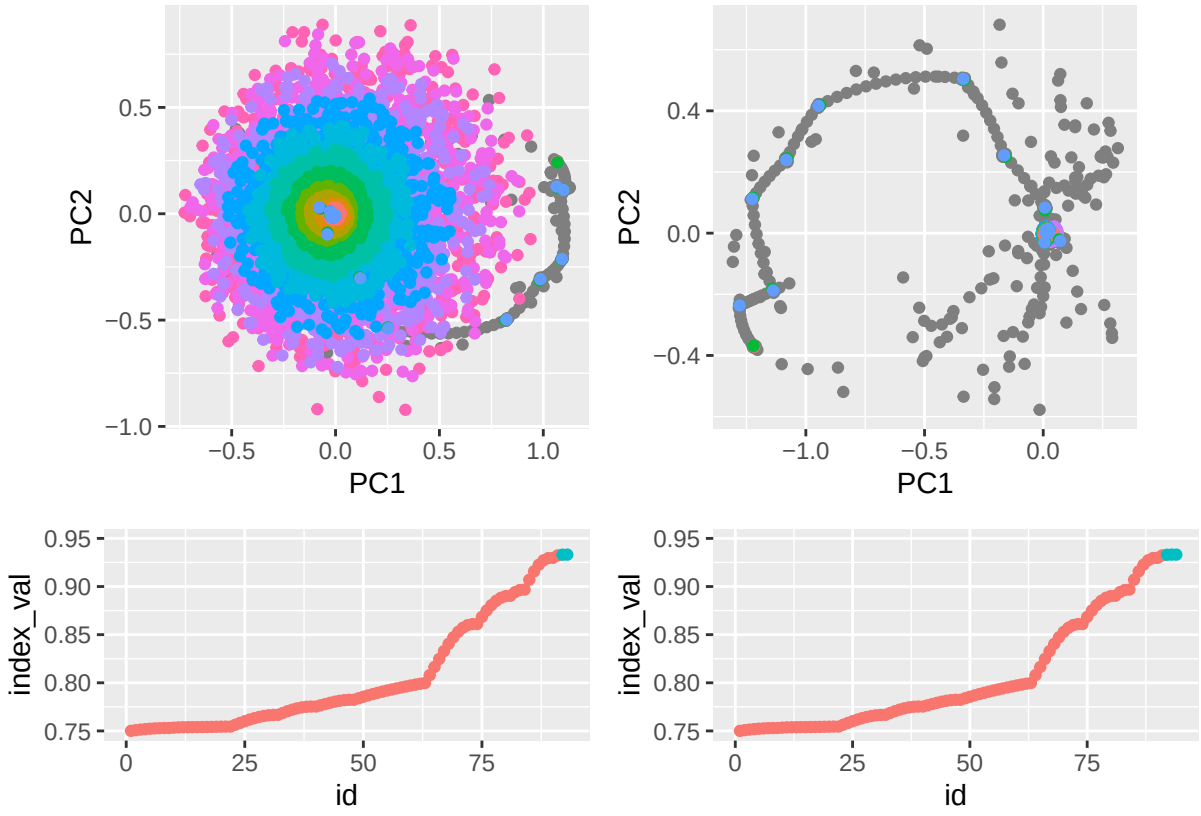

Figure 11: PCA plot of two different polish alpha initialisations. A default polish alpha = 0.5 searches a larger space that is unncessary while a small customised initial value of polish alpha will search near the ending basis. Both intialisations reach the same ending index values.

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.

Nelder, J. A. & Mead, R. (1965), 'A simplex method for function minimization', *The computer journal* **7**(4), 308–313.

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/*