

# Title here

Author 1 \*

Department of YYY, University of XXX  
and

Author 2

Department of ZZZ, University of WWW

May 31, 2020

## Abstract

1. check consistency of using PP for projection pursuit and PPI for projection pursuit index

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

---

\*The authors gratefully acknowledge ...

# 1 Introduction

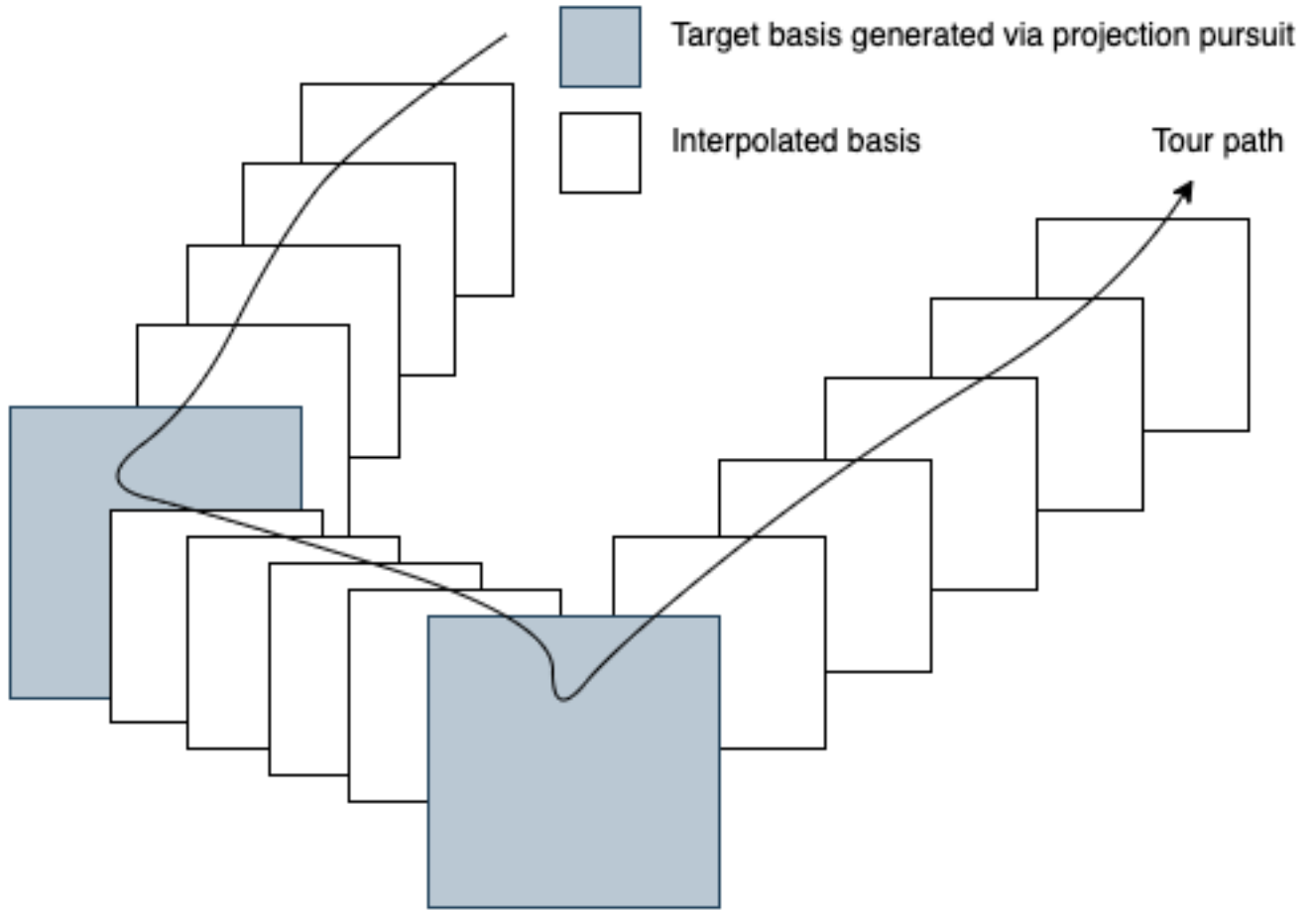
## 1.1 Tour

Tour provides a way to explore multivariate data interactively via an established tour path. A tour path is formed by interpolating between randomly generated plane. Different types of tours are available depends on the purpose of the exploration i.e. a grand tour is suitable for randomly exploring the data from different angles; a guided tour detects a particular structure in the data. a manual tour allows user to manually control the projection (Cook & Buja 1997).

## 1.2 Guided tour

Guided tour is usually used in conjunction projection pursuit, a method coined by Friedman & Tukey (1974) to detect “interesting” low-dimension projection of multivariate data. A projection pursuit requires the definition of a projection pursuit index function and an optimisation routine. The projection pursuit index measures the “interestingness” of data defined as its departure from normality. Numerous indices have been proposed in the literature, including lengendre 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). An optimisation routine is required to find the projection basis (thus projection) that maximises the projection pursuit index. The discussion of existing optimisation procedure will be discussed in the next section.

Guided tour creates visualisation for the projections found by projection pursuit by constructing a tour path. An illustration modified from (Buja et al. 2005) entails the geodesic interpolation between the plane generated by projection pursuit.



### 1.3 Optimisation methods in projection pursuit literature

As Friedman & Tukey (1974) said “... , the technique use for maximizing the (one- and two-dimensional) projection index strongly influences both the statistical and the computational aspects of the procedure.” The quality of the optimisation procedure largely affect the tour view and thus, the interesting projection one could possibly observe. An ideal optimisation procedure needs to have following characteristics:

- *Being able to handle non-differentiable index function:* the index function could be noisy and non-differentiable
- *Being able to optimise with constraints:* the projection matrix is restricted to an orthonormal matrix.

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

Below present three existing methods in tour.

Posse (1995) presented a random search algorithm that samples new basis in the neighbourhood of the current basis. The neighbourhood is defined via 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. [Pursuit package uses this method and it works great! But I don't think we implement this - although don't think it is too hard to do it].

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  $\alpha$  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  $A = \min\left(\exp\left(-\frac{I(B_j) - I(B_i)}{T(i)}\right), 1\right)$  where  $I(\cdot)$  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 Iterative algorithm and its diagnostics

### 2.1 Tour components

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). A tour includes two major components: a *generator* that generating the projection basis according to projection pursuit and an *interpolator* that performing geodesic interpolation between the projection basis.

The pseudo-code below illustrates the implementation of guided tour in the *tourr* package. Given an projection pursuit index function and a randomly generated projection basis (current basis), the optimisation procedure produces a target basis inside `generator()`. Both the current basis and the target basis will be supplied to `tour_path()` to prepare information needed for constructing a geodesic path. This information is then used to compute a series of interpolating bases inside the `tour()` function. All the basis will be sent to create animation for visualising the tour in the `animate()` function.

```
animation <- function(){

  # compute projection basis
  tour <- function(){

    # construct bases on the tour path
    new_geodesic_path <- function(){
      tour_path <- function(){

        # GENERATOR: generate projection basis via projection pursuit
```

```

guided_tour <- function(){
  generator <- function(){

    # define projection pursuit index
    # generate the target basis from the current basis via optimisation
  }
}

# prepare geodesic information needed for interpolating along the tour path
}

# INTERPOLATOR: interpolate between the current and target basis
function(){
  # generate interpolating bases on the geodesic path
}
}

# animate according to different display methods
}

```

## 2.2 Diagnostics

Visualisation has been widely used for exploring and understanding data. Visualisation presents information in a graphical manner and often allows people to see information they would otherwise not see in the reporting of numerical summarisation. Visual diagnostics can be real-time or post-run. Real time diagnostic directly uses the data produced in the algorithm to produce visual representation and thus doesn't need to store the data. This section focuses on the definition and production of post-run diagnostics and the next section discusses real-time diagnostics.

Post-run diagnostics requires the data produced during the algorithm to be stored/

saved in order to produce plot diagnostics.

The graphical system in R is established based on the conception grammar of graphic (Wickham 2010), where a graphic is defined using stacked layers in a coordination system. A layer includes 1) the dataset that powers the plot; 2) a geometric object that represents the visual shape of the plot and 3) relevant statistical transformation if needed. An important concept in the grammar of graphic is *aesthetic mapping*. Aesthetic mapping links the variable in a dataset to information needed to produce a geometric object. For example, we map one variable on the x-axis and another on the y-axis to create a scatterplot. To create a boxplot, we first map one variable on the x-axis and then map the five point summary of another variable on the y-axis. This computation of five point summary from the origin variable is statistical transformation. This definition of graphic through layers provides advantages to produce complex plots since all the plots, however complex, can be decomposed into basic geometric objects shaped by the variable supplied.

This idea of decomposing a graph to basic elements inspires me to characterise the diagnostics of iterative algorithm in a similar fashion. The optimisation in many machine learning algorithms these days is iterative in nature, while remains as a black box. Being able to diagnose it visually allows researchers to have the tool to unfold the myth and thus provides more understand to the algorithm.

The concept of grammar of graphic requires a dataset to be supplied in making a plot, thus a global object needs to be created in the iterative algorithm. The variables will be mapped into element of a graphic to explore and thus should contain all the parameters of interest. The next section shows how a global object is created in projection pursuit guided tour.

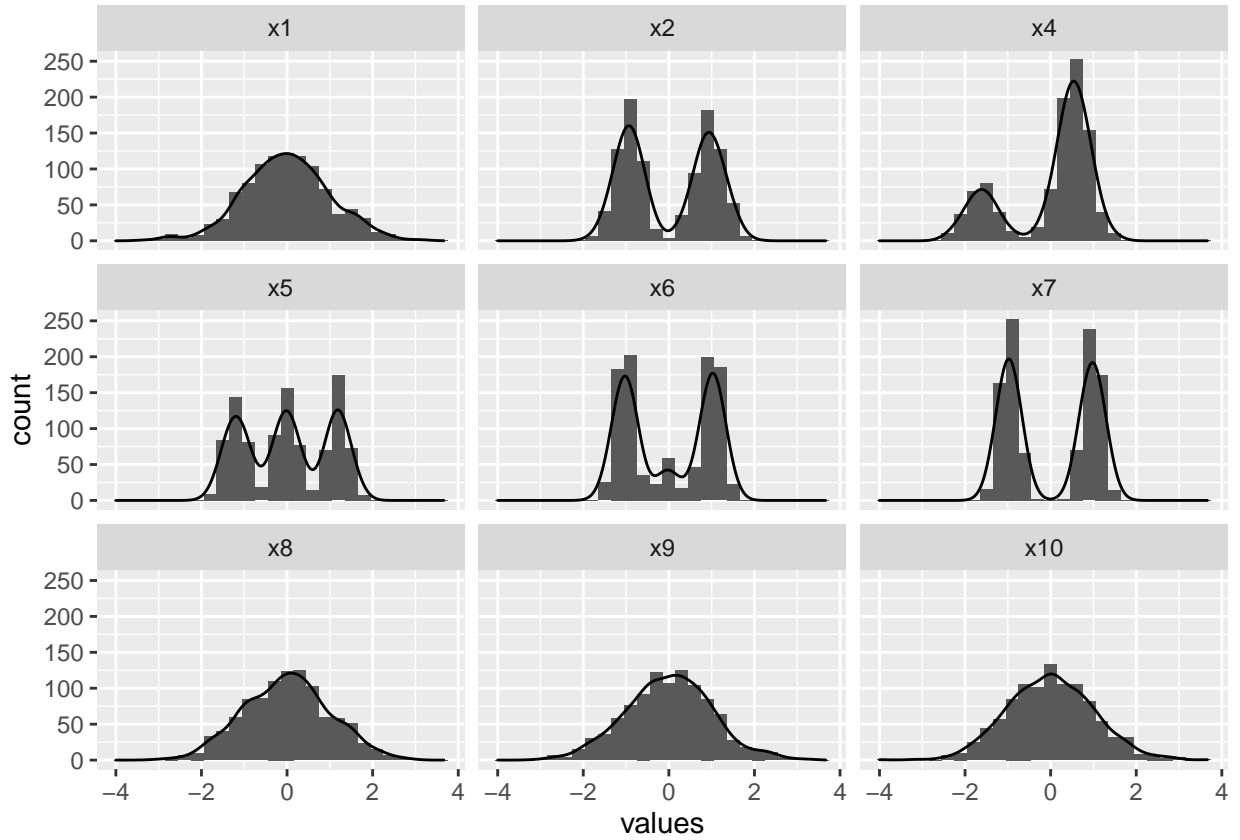
## 2.3 Data Structure

In the current implementation of the `tourr` package, while the target basis generated by the projection pursuit can be accessed later via `save_history()`, interpolating bases and those randomly nearby bases generated in the optimisation are not stored. This creates difficulties for fully understand the behaviour of the optimisation and interpolation of tour in complex scenario **[need a rephrase this part]**.



Two set of simulated data are used in the demonstration of the visualisation and diagnostics of the tour optimisation. A small dataset consists of 1000 randomly simulated observations of five variables ( $x_1$ ,  $x_2$ ,  $x_8$ ,  $x_9$ ,  $x_{10}$ ).  $x_2$  is the informative variable simulated from two bi-modal normal distribution centred at -3 and 3 with variance being 1 and the other four are simulated from  $N(0, 1)$ . The data has been scaled to ensure  $x_2$  has variance of 1.

A larger dataset contains more informative variables ( $x_3$  to  $x_7$ ) of different types.  $x_3$  takes 500 positive one and 500 negative one. The distribution of all the variables except  $x_3$  is plotted below. *[should I introduce the dist for each var?]*



Once the dataset is sent to the `tourr` package, all the information generated will be stored in a global structure. The global structure consists of six columns: `basis`, `index_val`, `tries`, `info`, `loop`, `id` and captured all the basis generated during whole tour process. The example below presents the global object of a 1D projection of the small dataset with geodesic searching method.

```
holes_1d_geo %>% head(5)
```

```
## # 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
```

`tries` has an increment of one once the generator is called (equivalently a new target basis is generated); `info` records the stage the basis is in. This would include the interpolation stage and the detailed stage in the optimisation i.e. `direction_search`, `best_direction_search`, `line_search` and `best_line_search` for geodesic searching (`search_geodesic`); `random_search` and `new_basis` for simulating annealing (`search_better`). `loop` is the counter used for the optimisation procedure and thus will be `NA` for interpolation steps. `id` creates a sequential order of the basis. This information will be stored and printed when the optimisation ends and can be turned off via `print = FALSE`. Additional messages during the optimisation can be displayed via `verbose = TRUE`. Another examples is a 2D projection of the larger dataset with two informative variable (`x2` and `x7`) using `search_better` method. Notice in this example, the dimension of the bases becomes 6 by 2.

```
holes_2d_better %>% head(5)
```

```
## # A tibble: 5 x 8
##   basis          index_val tries info          loop method      alpha    id
##   <list>          <dbl> <dbl> <chr>          <dbl> <chr>      <dbl> <int>
## 1 <dbl[,2] [6 x 2]>    0.804     1 start          NA <NA>        0.5     1
## 2 <dbl[,2] [6 x 2]>    0.793     1 random_search    1 search_bett~ 0.5     2
## 3 <dbl[,2] [6 x 2]>    0.784     1 random_search    2 search_bett~ 0.5     3
## 4 <dbl[,2] [6 x 2]>    0.773     1 random_search    3 search_bett~ 0.5     4
## 5 <dbl[,2] [6 x 2]>    0.795     1 random_search    4 search_bett~ 0.5     5
```

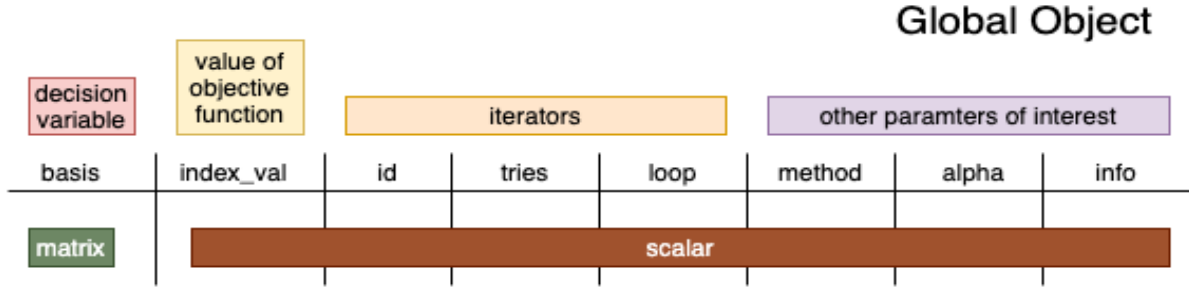


Figure 1: this is xxxx

### 3 Post-run static diagnostics plots

Given all the information of interest in a global object, the next thing is to design its visual representation that can diagnose our algorithm. In an iterative algorithm, one would be interested to know how one parameter changes as the algorithm iterates. The parameter could be the decision variable, the value of the optimisation function or other parameter that facilitate the algorithm. In projection pursuit guided tour, apart from the decision variable projection basis, the value of index function (`index_val`), we are also interested to explore the effect the different searching methods, the neighbourhood parameter `alpha`, and the stage a particular observation is in (`info`). These are the variables that will be mapped into the diagnostic plot. The global object also has its time series nature since the observations are recorded as the algorithm progresses and thus can't be switch in rows. This time series structure is always hierarchical and nested since each row can be labelled with the smallest unit `id` and up by the iterative structure of the algorithm. Take our example, in the global object, each row has an `id` label which is the smallest unit the observation is ordered. A larger ordering unit is `tries` since it is updated every time a new target basis is found. A nested ordering unit is `loop`, which increases by one as searching method iterates and starts over at a new `tries`. These features explain the definition of the global object for projection pursuit guided tour in Figure 1.

### 3.1 Explore scalar parameters

The most interesting parameter to explore is the value of objective function. The points recorded in the global object can be divided into two broad categories: searching points and interpolating points

- *Searching points* include the observations that are recorded in the searching algorithm in order 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.

#### 3.1.1 Explore searching points

As mentioned previously, 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 towards the end occupying the vast majority of the space in the plot.

This motivates the use of summarised statistics. 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.

Boxplot is a suitable candidate that provides five points summary of the data, while it has one drawback: it doesn't report the number of point in each box. We may risk losing information on how long it takes for the search 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. This is achieved via the `cutoff` argument. A line geometry is also added to link the points with the largest index value for each `tries`. This helps to visualise the improvement made by each `tries`.

*Example: exploring searching points* The data is sourced from a 2D projection of the larger dataset and `search_better` is used with `max.tries = 500`. In Figure ?? and ??, a

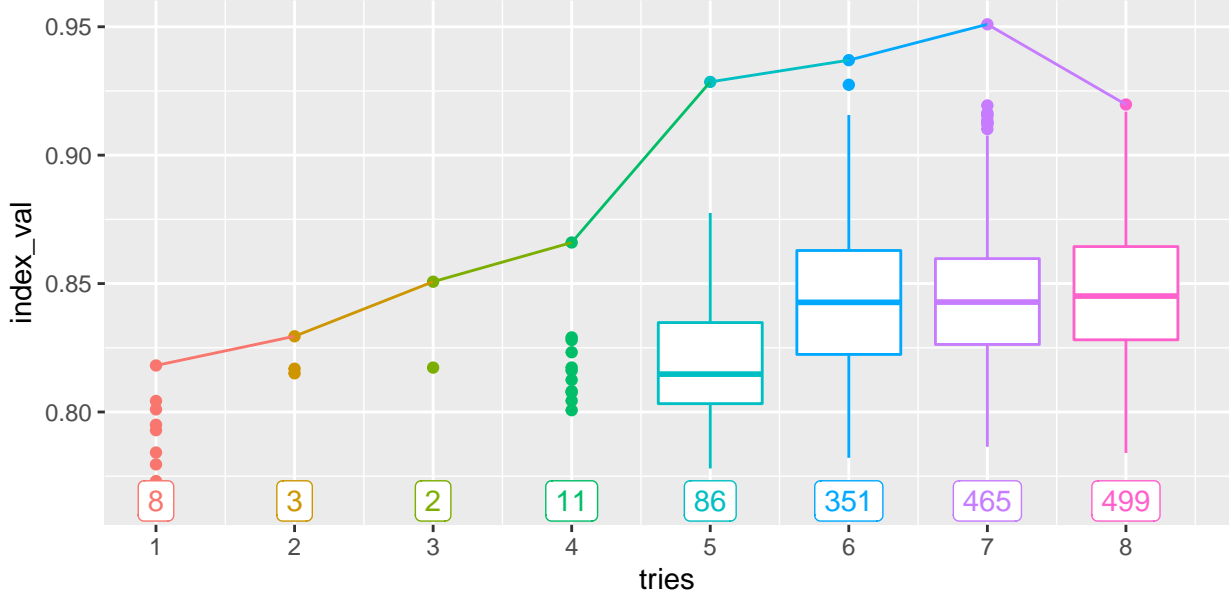


Figure 2: A comparison of plotting the same search points with different plot designs. The left plot doesn't efficiently use the plot space to convey information from the plot while the right plot provides good summarisation of data and number of points in each tries.

comparison has been made on visualising the searching points with `id` and `tries` on the x-axis, colored by `tries`. In Figure ??, the searching points of the first few `tries` are squeezed in a small width leaving the large uninteresting searching points in the last three `tries` taking a vast majority of the plotting space. While in Figure ?? the data is spaced by `tries` evenly in the plot. 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 suggest the largest improvement happens at `tries = 5`.

### 3.1.2 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 the tour algorithm and different index functions.

*Example: Interruption* This examples uses `search.better` for a 2D projection on the larger dataset using the `holes` index. As mentioned previously, the interpolation starts

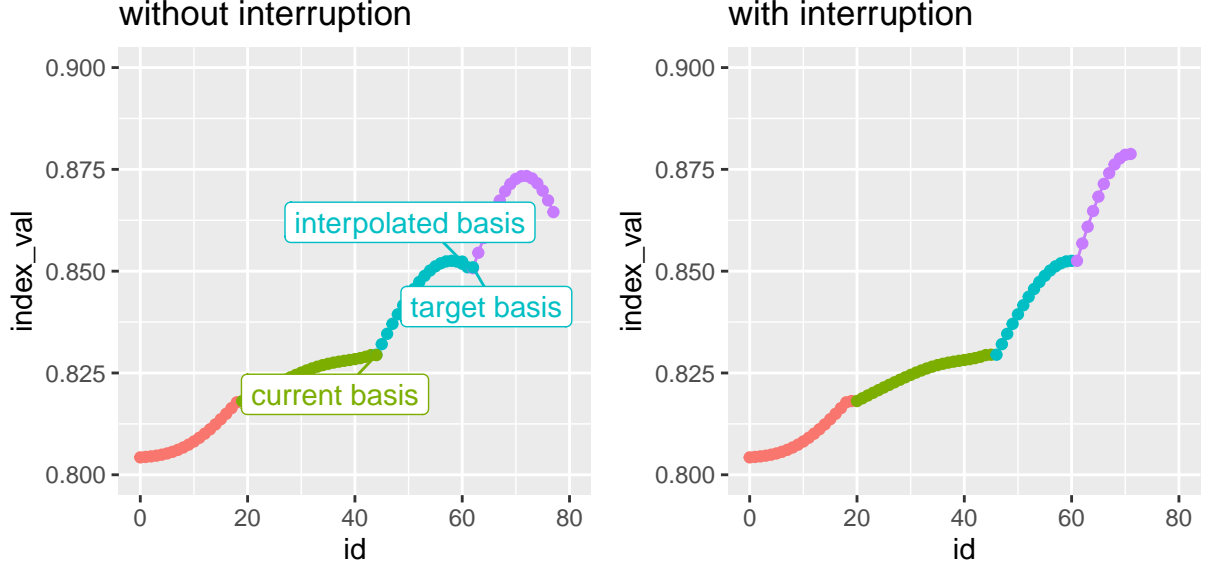


Figure 3: Trace plots of the interpolated basis with and without the interruption. The interruption stops the interpolation when the index value starts to decrease at  $\text{id} = 60$ . The implementation of the interruption finds an ending basis with higher index value using fewer steps.

from the current basis to the target basis, which has been found by the projection pursuit algorithm to have a higher index value. After the interpolation, the target basis will become the current basis and send back to the projection pursuit algorithm to find the next target basis. From figure ??, it is possible that there are bases with index value higher than the target basis on the interpolation path and these bases could be used to search for new basis in the next iteration.

Thus an interruption is constructed to accept the interpolating bases up to the one with the larger index value on the interpolation path, and that basis is taken as the current basis for the next iteration. After implementing this interruption, the tracing plot with the same configuration is shown on the lower panel. We can observe that rather than interpolating to the target basis at  $\text{id} = 62$ , the interpolation stops before the index value starts to decrease at  $\text{id} = 60$ . 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

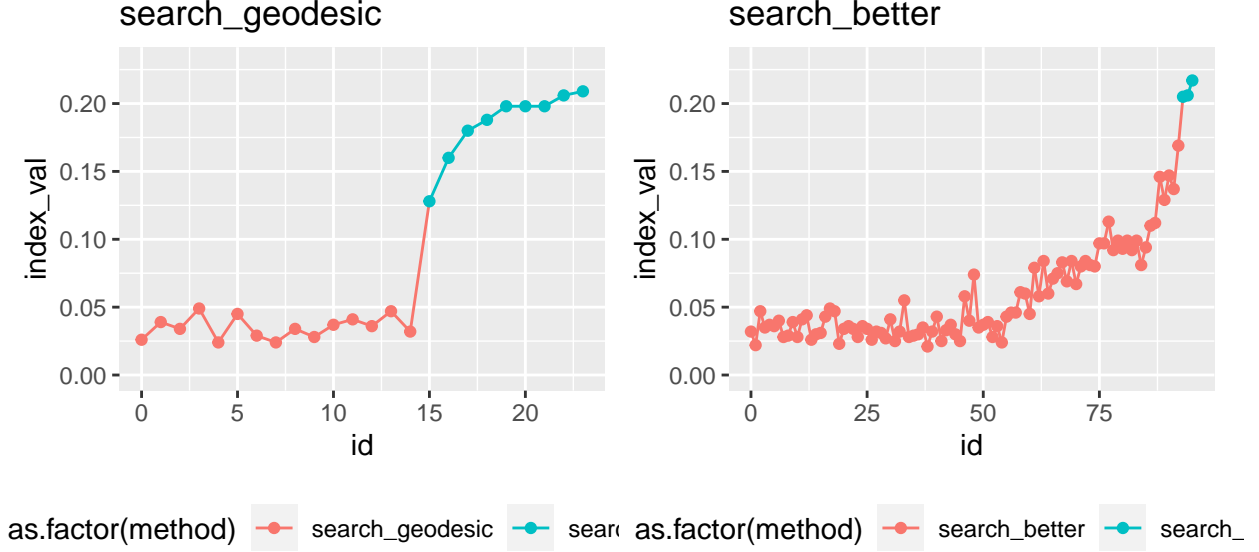


Figure 4: Comparison of two different searching methods: `search_geodesic` and `search_better` on 1D projection problem for a noisy index: `kol_cdf`. The geodesic search rely heavily on the polishing step to find the final index value while search better works well.

??, is smooth, while this may not be the case for more complicated index function. `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 visualisations below shows some characteristic of this index function.

Figure ?? compares the tracing plot of the interpolating points for `search_geodesic` and `search_better`. Rather than a smooth interpolation, the interpolation path for `kol_cdf` 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.

Moving from a 1D projection on a dataset with only one informative variable, we now introduce another informative variable in the data and remain 1D projection. In this problem, there are two local maximum being 0.17 when the projection matrix is  $[0, 1, 0, 0, 0, 0]$  and 0.228 when the projection matrix is  $[0, 0, 1, 0, 0, 0]$ . As in Figure ??, using `search_better` with different seeds, both local maximum can be found. A simulated annealing algorithm,

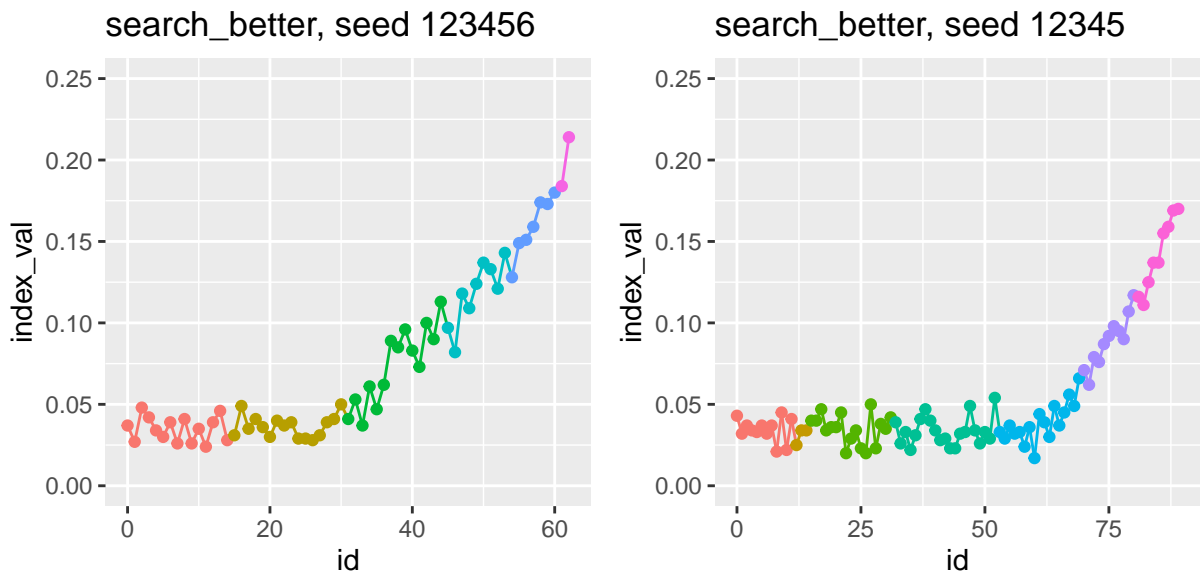


Figure 5: The trace plot search better in a 1D projection problem with two informative variables using different seeds (without polishing). Since there are two informative variables, setting different value for seed will lead search better to find either of the local maximum.

`search_better_random` in Figure ??, can always find the global maximum at a cost of a large number of `tries`.

### 3.1.3 Adding more variables to the mapping

At previous two sections, only the iterator and the index value are mapped onto the x and y aesthetics of the plot; while more aesthetics i.e. color, could be added to compare other parameters in the global object. Two examples are shown below to explore and compare different searching methods and neighbourhood parameter alpha.

*Example: Polish* In principle, all the optimisation routines should result in the same output on the same problem while this may not be the case 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



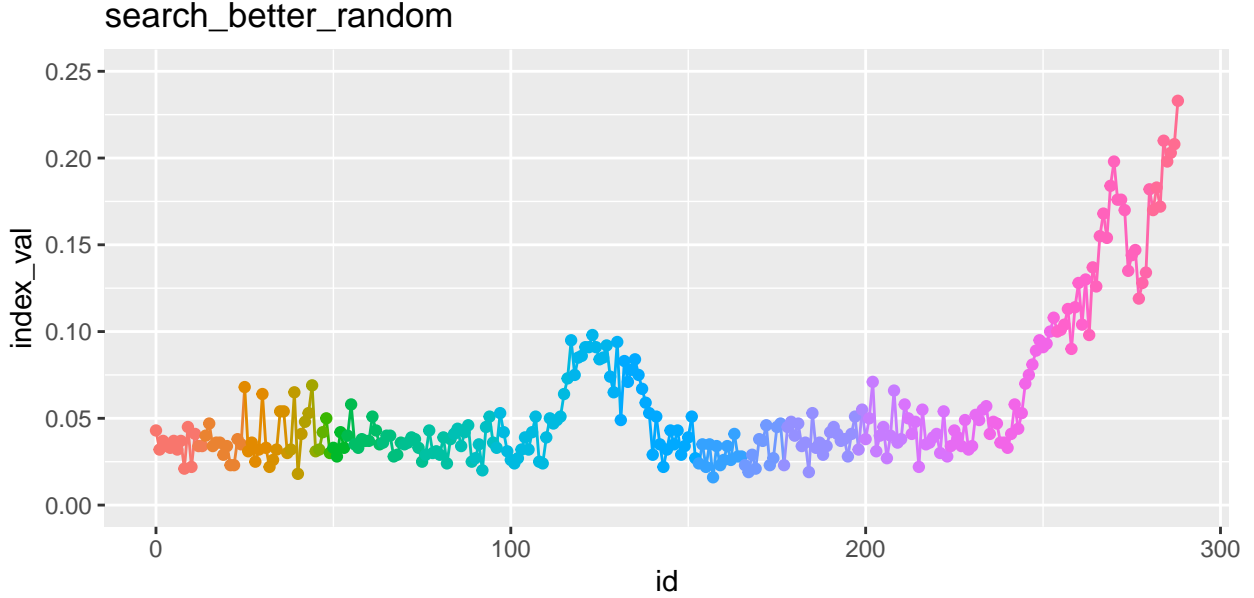


Figure 6: Using search better random for the problem above will result in finding the global maximum but much larger number of iteration is needed.

largest index value becomes the candidate. If its index value of the candidate basis 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 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`

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)

```

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 ??, after polishing, both trials attain the same index value. However, a small `max.tries` of 25 is not sufficient for the algorithm to find the true maximum. This is because 25 tries is not sufficient for the 2D searching space.

*Example: The neighbourhood parameter  $\alpha$*  Add an example on comparing the neighbourhood parameter in `search_better` & `search_posse`.

```

# nrow(holes_2d_better_max_tries)
# nrow(holes_2d_pos)

```

## 3.2 Explore matrix parameter

Matrix parameter could also be interested to explore, for example, in the tour, we are interested to explore the relative position of the projection basis in the vector space. A

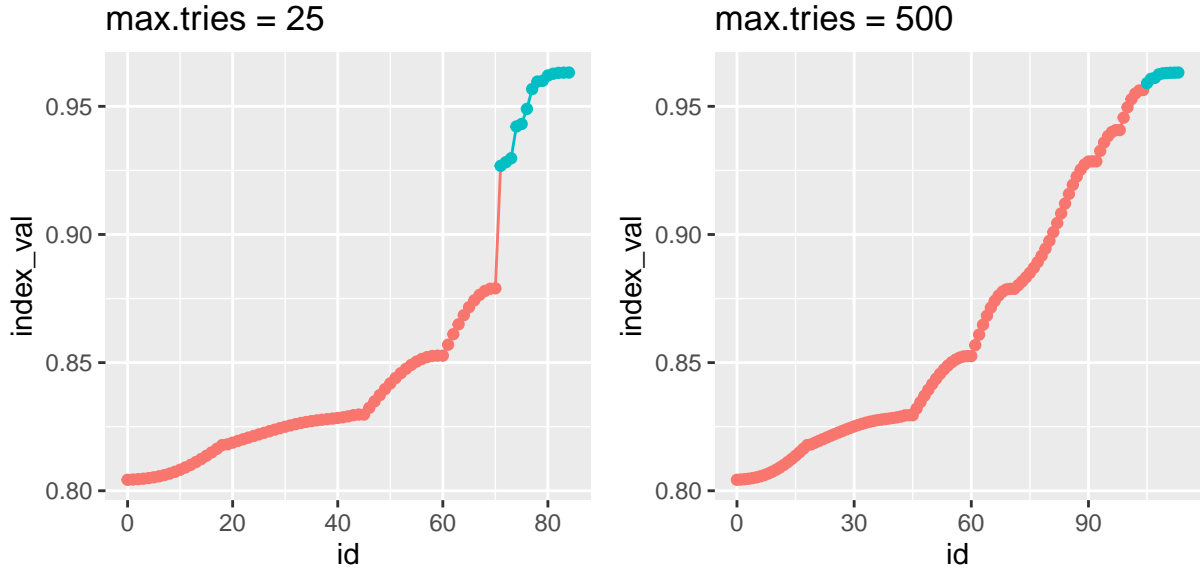


Figure 7: Breakdown of index value when using different max.tries in search better in conjunction with search polish. Both attain the same final index value after the polishing while using a max.tries 25 is not sufficient to find the true maximum.

projection basis is a vector/matrix whose row number is the number of variable in the given dataset. This imposes difficulties in visualisation since we are bounded to perceive at most three dimensions. Thus, principal component analysis is used to reduce the dimension of the projection basis and the first two principal component are mapped to the x and y axis of the plot. Another variable of interest could be mapped to the color aesthetics to see how it changes as the projection basis changes in the projected 2D space. [have a second read here]

*Example: understand search\_geodesic via mapping **info** to color 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 works. 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 basis are then interpolated. In the next iteration, the target basis becomes the current basis and then procedures continues. [should probably reword this part with info levels xxx]

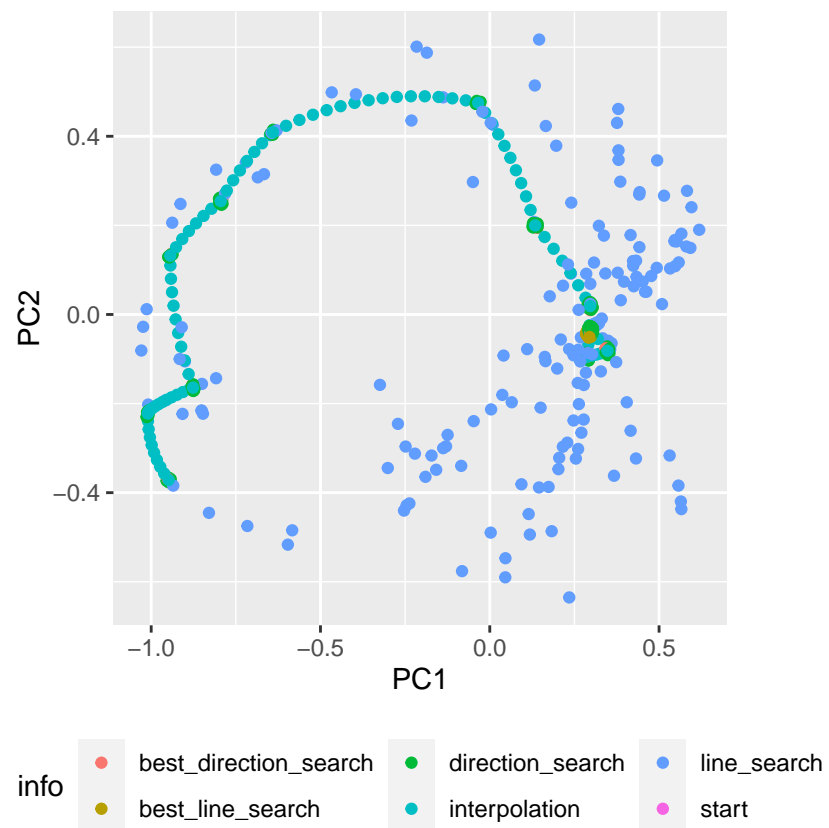


Figure 8: PCA plot of search geodesic Coloring by info allows for better understanding of each stage in the geodesic search

*Example: initial value for polishing alpha in search geodesic* `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. A default initial value for polishing step is 0.5 and we are interested in whether this is an appropriate initial value. The problem is a 1D projection of the small dataset using `search_geodesic` and followed by `search_polish`. Figure ?? display all projection bases on the first two principal components, colored by the `polish_alpha` parameter in the polishing step. Rather than concentrating on the ending basis from `search_geodesic` as what polishing step is designed, it searches a much larger vector space that are unnecessary. Thus a user-supplied smaller initial value for `polish_alpha` would be ideal.

On the right panel of Figure ??, the `polish_alpha` is initialised as the projection distance between the last two target bases and we observe a concentrated searching space near the ending basis. With the supplied initial value for `polish_alpha`, the algorithm reaches the same ending index value with fewer iterations.

## 4 Real-time 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.

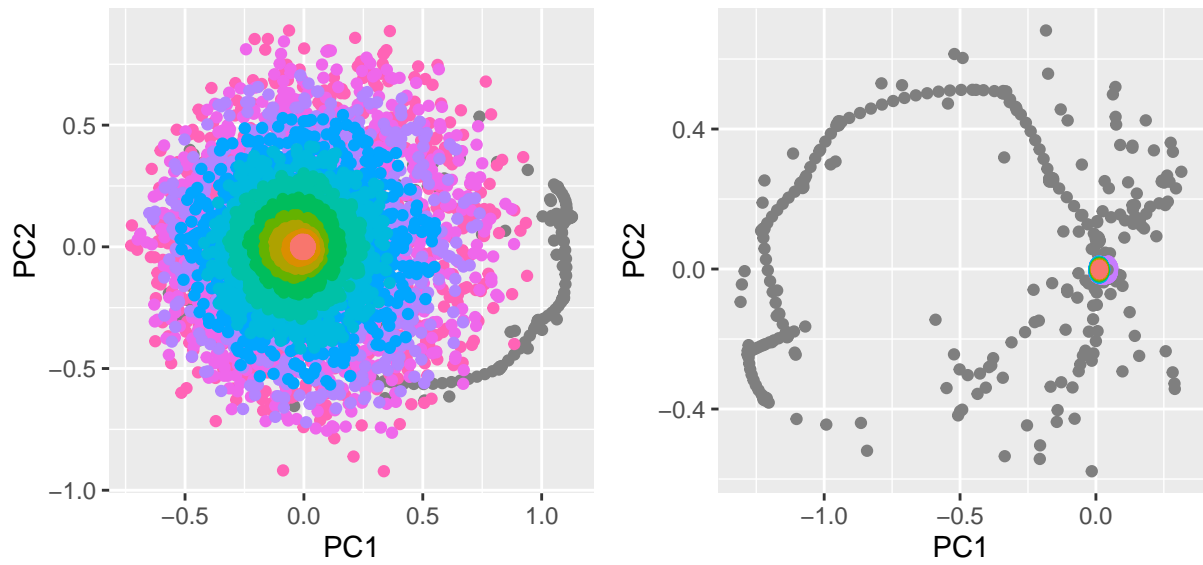


Figure 9: PCA plot of two different polish alpha initialisations. A default polish alpha = 0.5 searches a larger space that is unnecessary while a small customised initial value of polish alpha will search near the ending basis.

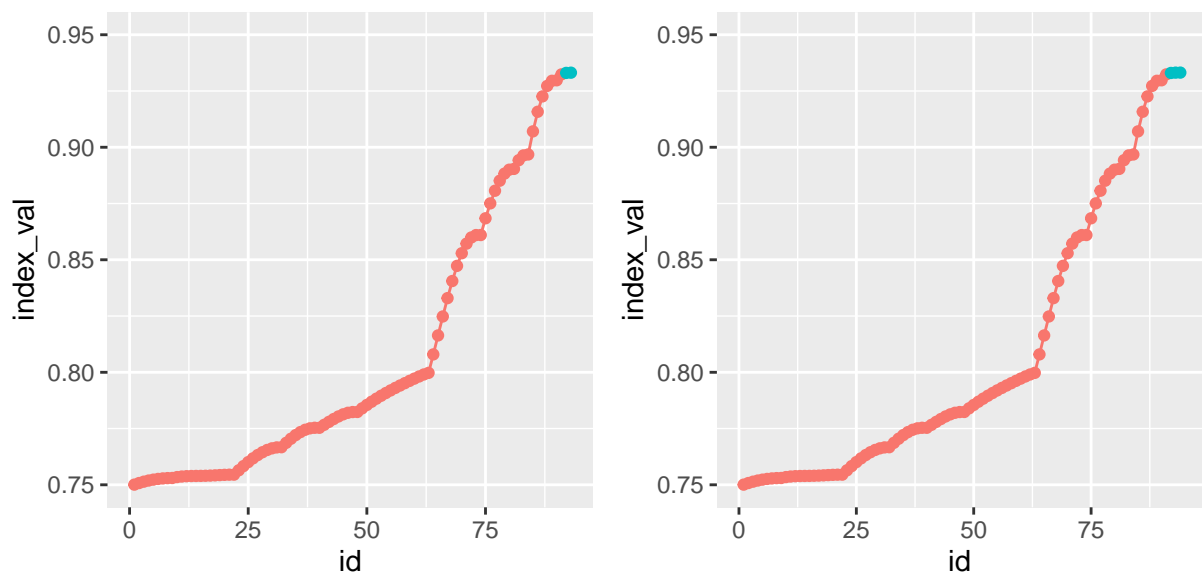


Figure 10: Comparing the trace of two different polish alpha initialisations. Both initialisations reach the same ending index values

# 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. (1997), ‘Manual controls for high-dimensional data projections’, *Journal of computational and Graphical Statistics* **6**(4), 464–480.
- 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.
- 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., 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/>