Title here

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

May 26, 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

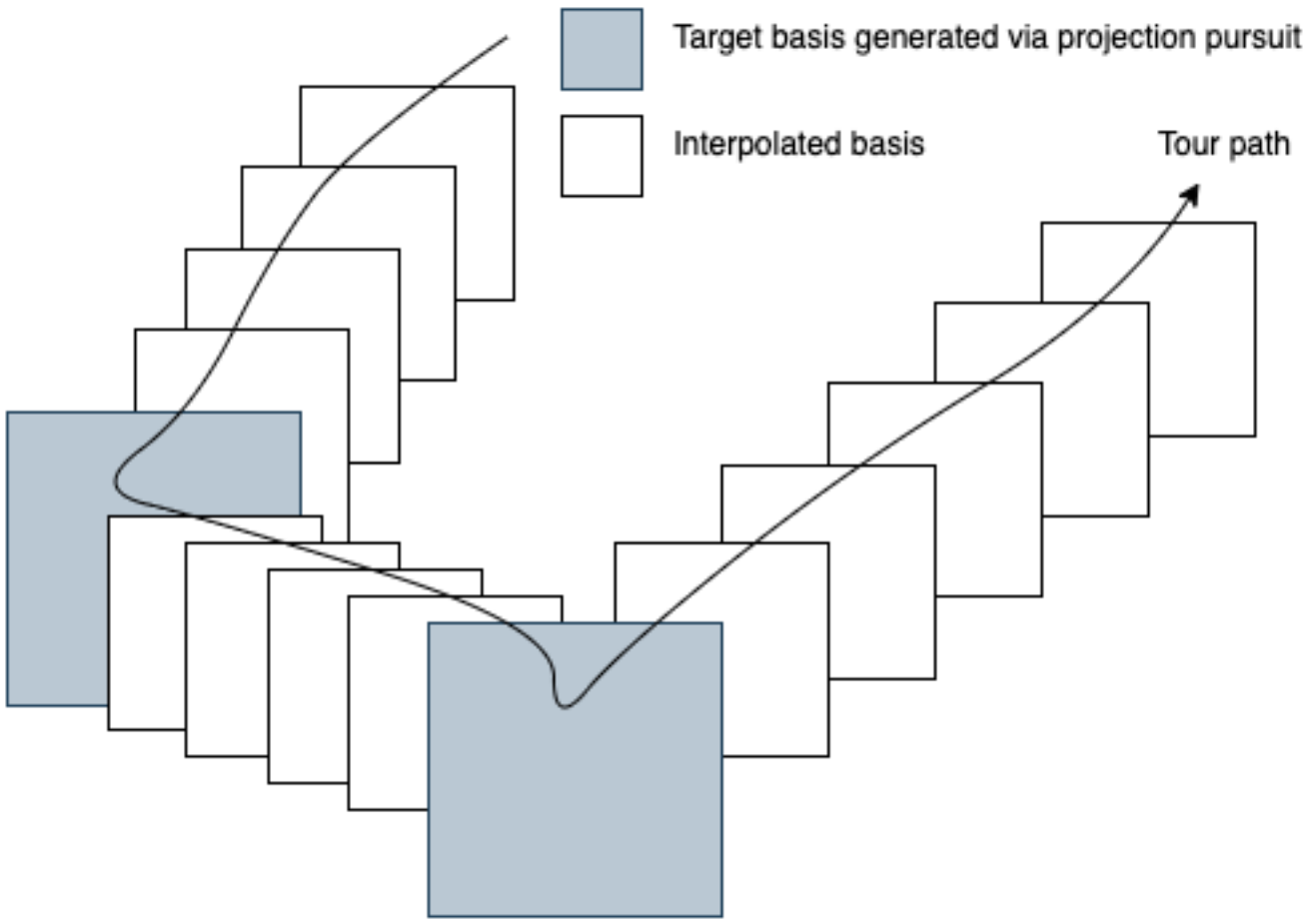
1.1 Tour

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-diemsion 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 bewteen 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-differentible
- *Being able to optimise with constraints*: the projection matrix is restricted to an orthornormal 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-dimenionsal 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 humber of tries n, the radius c is halved. The algorithm stops when the maximum numbere of iteration is attained or the radius c is less than a pre-determined number. [Pursuit package uses this method and it works greate! 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 psudo-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 pacakge help people to understand optimisation

- diagnostic on stochastic optim
- vis the progression of multi-parameter decision variable

- understanding learning rate - neightbourhood 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 psudo-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

Visualisatioin has been widely used for exploring and understanding data. Visualisation presents information in a graphicial manner and often allows people to see information they would otherwise not seen 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 diagnoistics 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 gramma 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 gramma 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 decompositing 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.

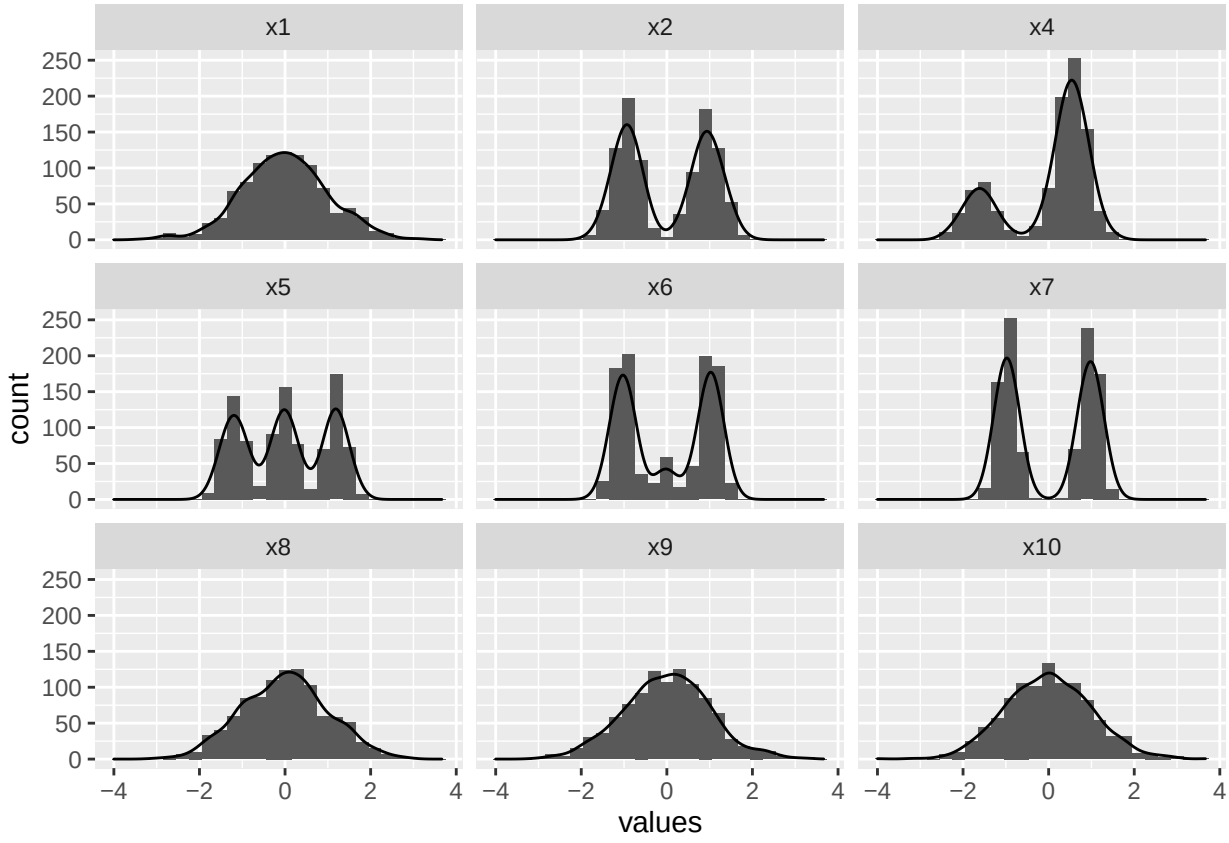
The concept of gramma of graphic requires a dataset to be supplied in making a plot, thus a global object needs to be created in the iterative algorithm. The variables will be mapped into element of a graphic to explore and thus should contain all the parameters of interest. The next section shows how 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 randonly 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 **[needa rephrase this part].**

Two set of simulated data are used in the demonstration of the visualiation and diagnostics of the tour optimisation. A small dataset consists of 1000 randomly simulated observations of five variables (`x1`, `x2`, `x8`, `x9`, `x10`). `x2` is the informative variable simulated from two bi-modal normal distribution centered 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 `x2` has variance of 1.

A larger dataset contains more informative variables (`x3` to `x7`) of different types. `x3` takes 500 positive one and 500 negative one. The distribution of all the variables except `x3` 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 columnes: `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 seraching 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
```

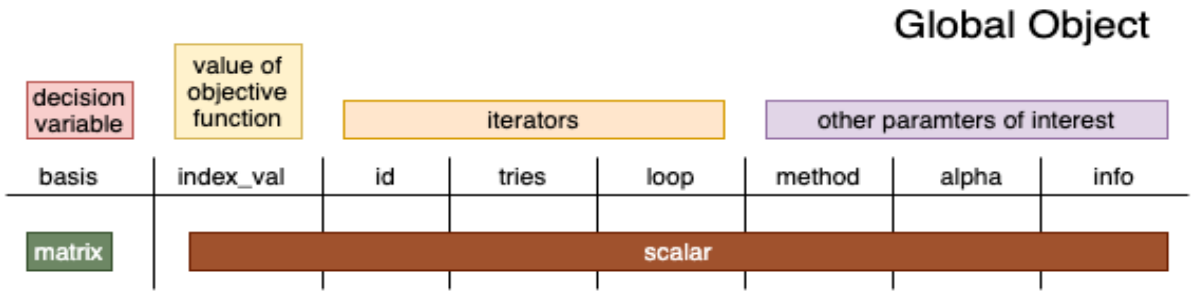


Figure 1: this is xxxx

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 could be the decision variable, the value of the optimisation function or other parameter that facilitate the algorithm. In projectio 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 hierachical 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 everytime 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. Given the structure of projection pursuit guided tour algorithm, whatever searching method we use, we could roughly divide the observation into three category: searching, updating and interpolating points.

- *Searching points* include the observations that ones recorded during the trials and errors in the searching algorithm to find the target basis. In one `tries`, there could be only a few searching points while towards the end of the algorithm, they can easily go to hundreds.

- *Updating points* are the observations for the target points and there are only one points per `tries`. We use "all searching points" to refer to the collection of both searching and updating points.

- *interpolation points* exist specifically in the guided tour in order to provide continuous animated view from one target basis to another, thus they are not included in the searching procedure. There are multiple points in one tries but it doesn't have `loop` value.

3.1.1 Explore all searching points

As mentioned previously, the largest feature of exploring searching points is its uncertainty in the number per `tries`. This makes it difficult to map `id` on the x-axis since the first few tries that can be informative to the evolving of the algorithm occupies a small number of points on the graph while the large less informative search points towards the end occupies the vast majority of the plot space.

This motivates the use of summarised statistics. At each iteration, rather than knowing the 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 is the one that prescribe the interpolation and future search. Therefore, boxplot is a suitable candidate.

However, boxplot doesn't tells us the number of point in each box. By showing boxplot for all `tries`, we lose information on how long it takes for the search to find the target

basis. We may still want to show the exact points of every point if there are only a few points. Therefore a cutoff point is placed as an argument to decise when to switch from point geometry to boxplot geometry.

This design is easy to implement with the concept of gramma of graphic. With the idea of layed gramma, we only need to subset the data and map it onto the either a point geometry or boxplot geometry. We can also add a layer with line geometry that connect the points with the largest `index_val` in each tries. This would show us which `tries` has the largest improvement.

A comparison of using scatter plot (all point geometry) and my new design is shown in Figure **??**. As we can see the points for the first few tries on hte scatter plot are all squeezed in a tiny region, being nearly unseeable. While the plot on the right balances out the points in each `tries`. By linking the updating points for each `tries`, we observe that the largest improvement occurs in `tries = 5` where the number of searching points dramastically increased from 11 points in `tries = 4` to 86 points in `tries = 5`.

3.1.2 Explore only interpolation points

While mapping `tries` to the x axis allows for viewing the magnitude of the interpolation and also its density, we can't really see the curvature of the interpolated path. Think this would be important when the index is no longer smooth and the interpolation will jump up and down. Thus the geom_point + geom_line specification is still preferred.

Example 1: Interruption The left panel of Figure **??** presents the trace plot of the interpolated basis using `search_better` to optimise the holes index (2D projection using the larger dataset). Tour interpolates from the current basis to the target basis, which has been found to have a higher index value by projection pursuit. The target basis will then be sent to the projection pursuit as the current basis to find the next target basis. We can observe from the plot that there are basis with even higher index value on the interpolation path. These bases could be used to search for new basis in the next round. Therefore, an interruption is constructed to only take interpolated basis up to the point where the index value stops increasing and the last basis is taken as the current basis for the next round of search. After implementing the interruption, the tracing plot with the

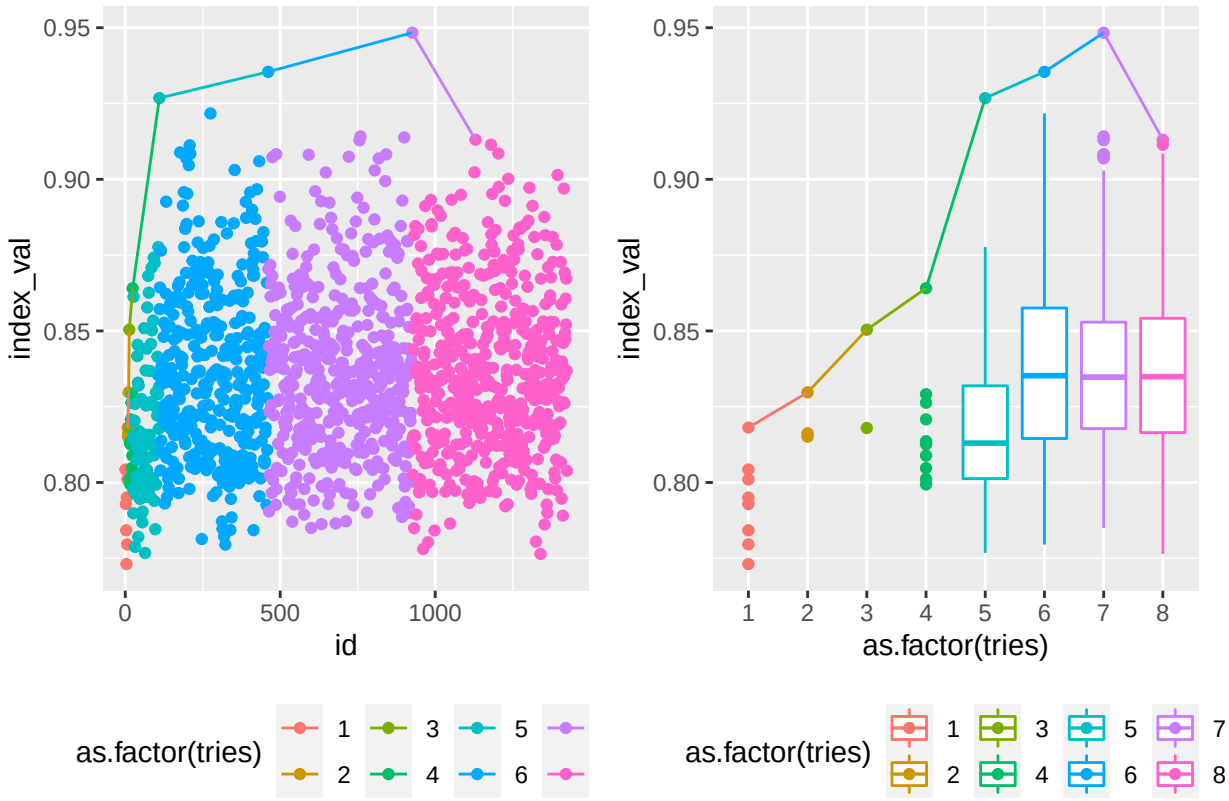


Figure 2: This is xxx

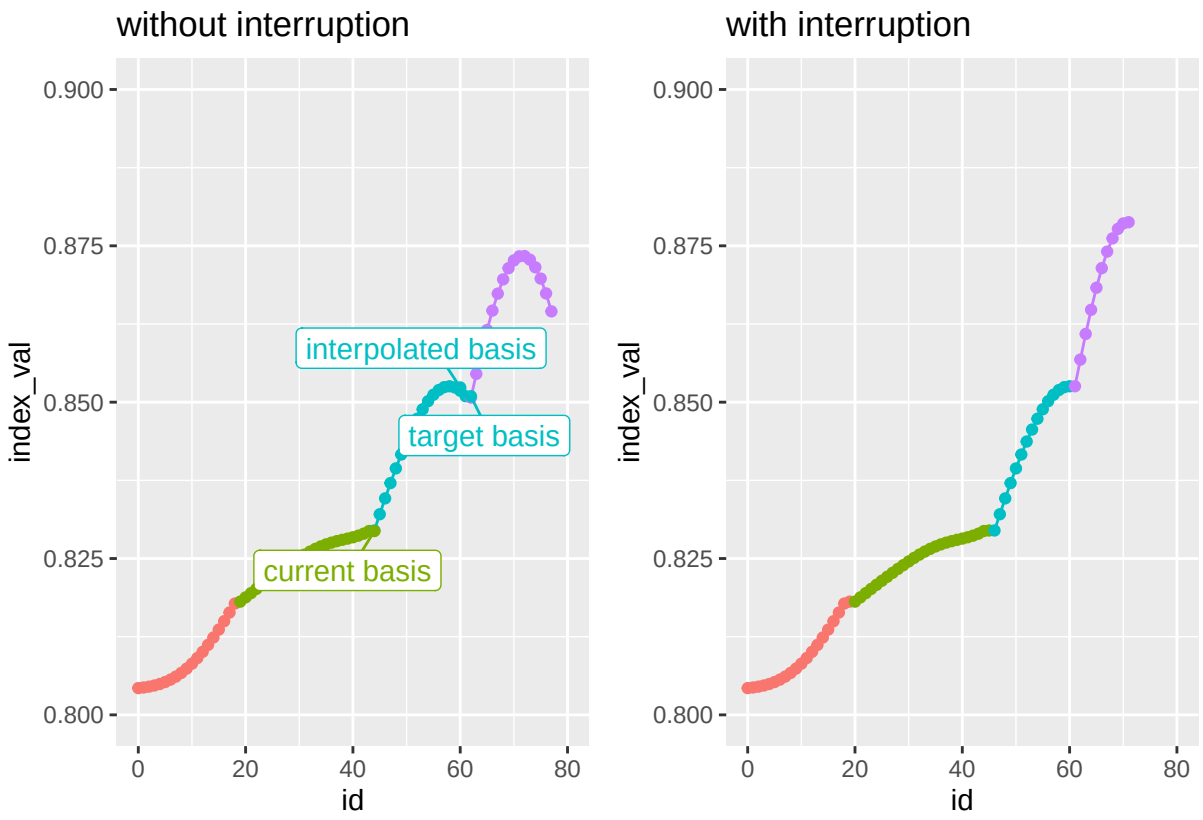
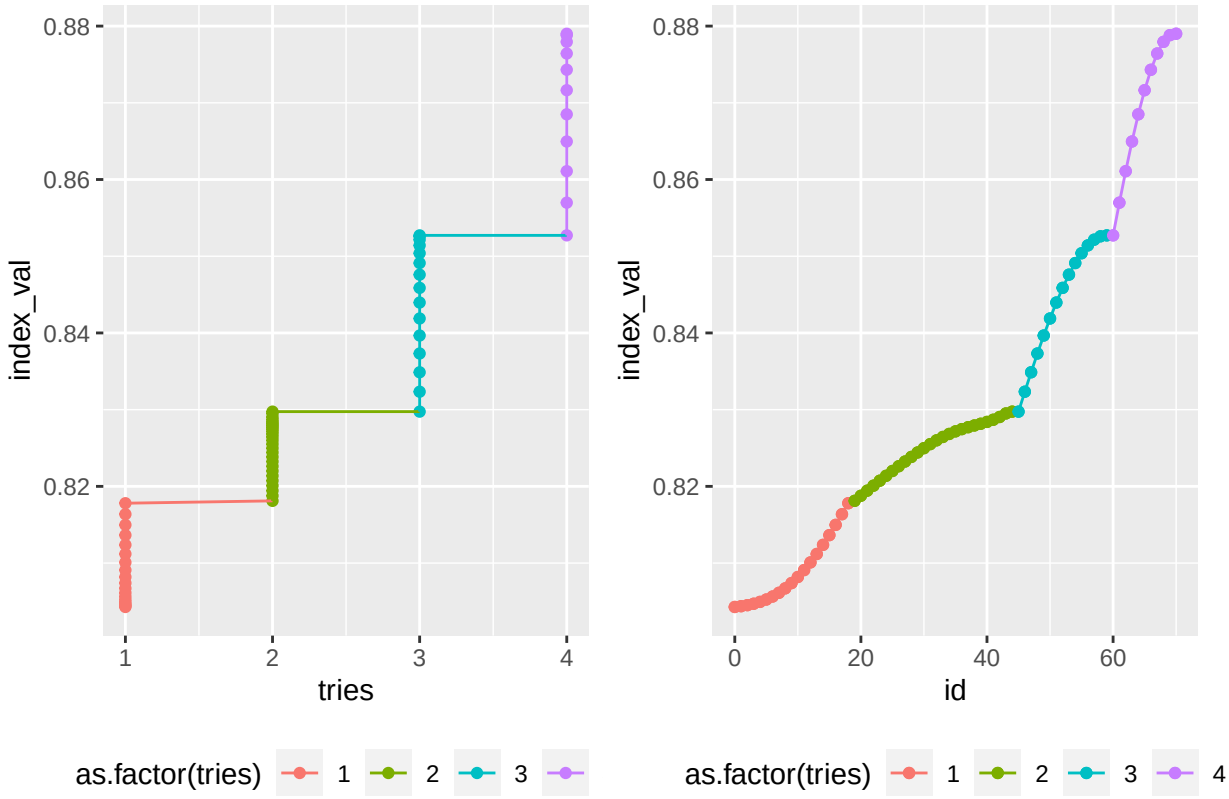


Figure 3: Trace plots of the interpolated basis with and without the interruption. The interruption stops the interpolation when the index value starts to decrease, see the difference at id being around 60. The implementation of the interuption finds an ending basis with higher index value using fewer steps.

same configuration is shown on the right panel. Rather than interpolate to the target basis, the interruption interrupt the process after id = 60 and proceeds the next round of search from the interpolated basis. Taking advantage of the interpolated basis results in a higher index value in the end with fewer steps.

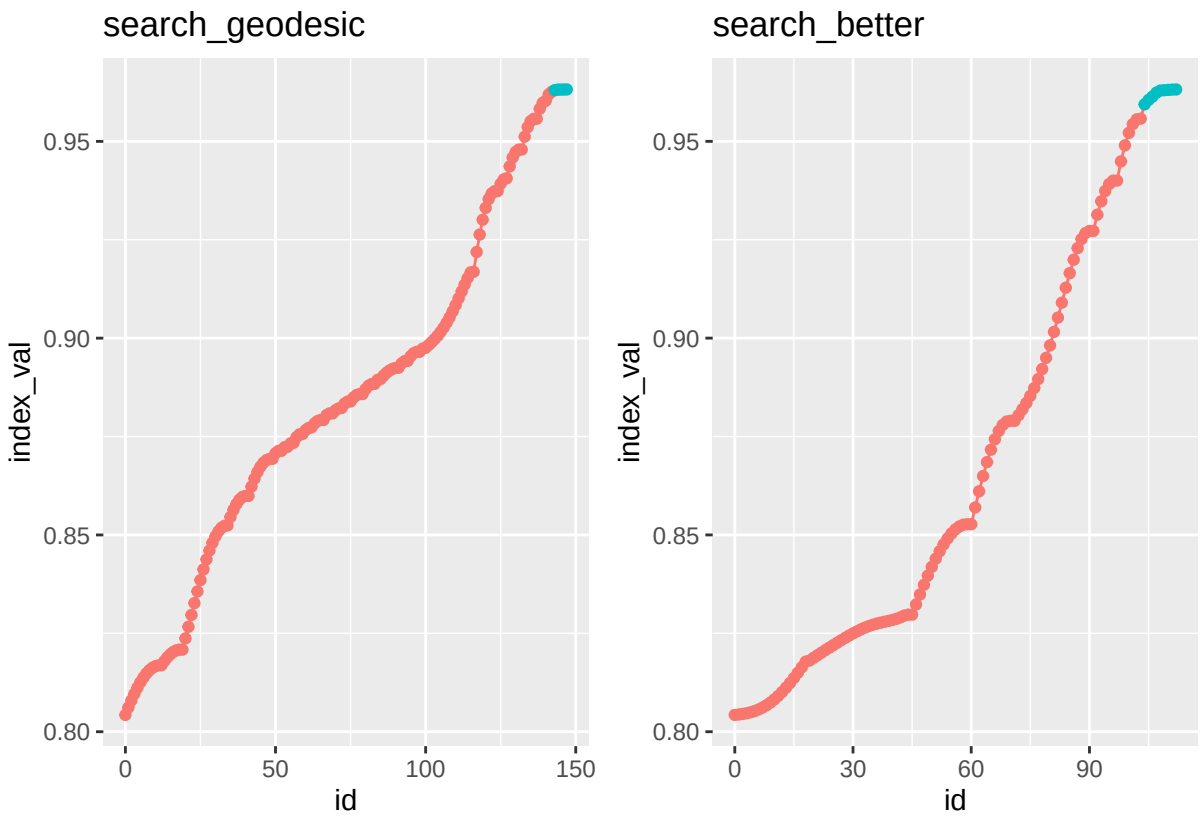
Example 2: Noisy index function Add an example on noisy function



3.1.3 Adding more variables to the mapping

For now we only include the iterator and `index_val` in the mapping of the plot, while more paramters of interest could be added to explore the effect their effect.

Example 3: Polish In principle, all the optimisation routines should present the same output on the same problem. In Figure **??**, red dots shows the trace of the interpolated basis using `search_geodesic` and `search_better` (with max.tries = 100), respectively, on the 2D projection problem. We can observe that they attain slightly different ending index value, which is not ideal. 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 starting basis and uses a brutal-force approach to sample a larger number of basis (`n_sample`) in the neightbourhood, controlled by `polish_alpha`. Among the `n_sample` basis, the one with the largest index value becomes the candidate. If its index value is larger than that of the starting basis, it becomes the center of the searching neightbourhood in the next round. If no basis is found to have larger index value than the starting basis, the searching neighbourhood will shrink 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 improvement 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 4: The neighbourhood parameter alpha Add an example on comparing the neighbourhood parameter in serach_better & search_posse.

```
nrow(holes_2d_better_max_tries)
```

```
## [1] 1529
```

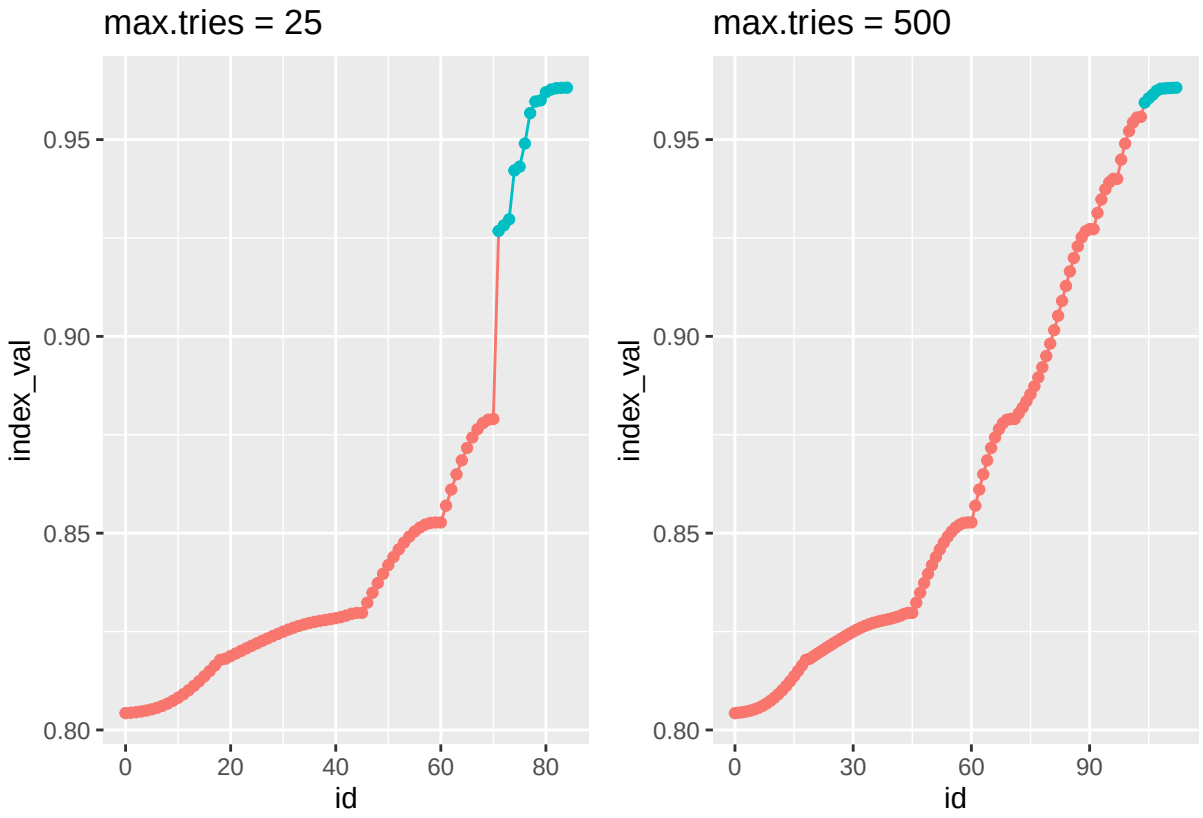
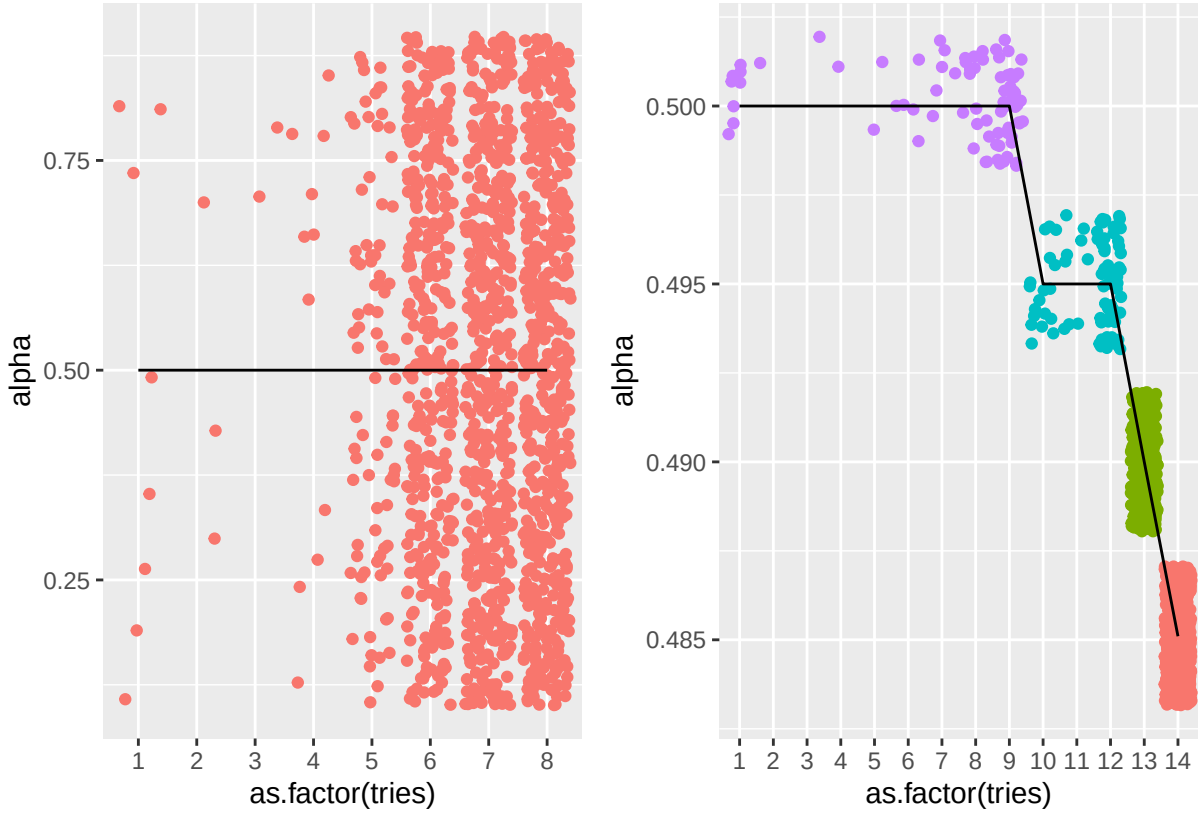


Figure 4: Breakdown of index value when using different max.tries in search better in conjunction with search polish. Both attain the same final index value after the polishing while using a max.tries 25 is not sufficient to find the ture maximum.

```
nrow(holes_2d_pos)
```

```
## [1] 1056
```

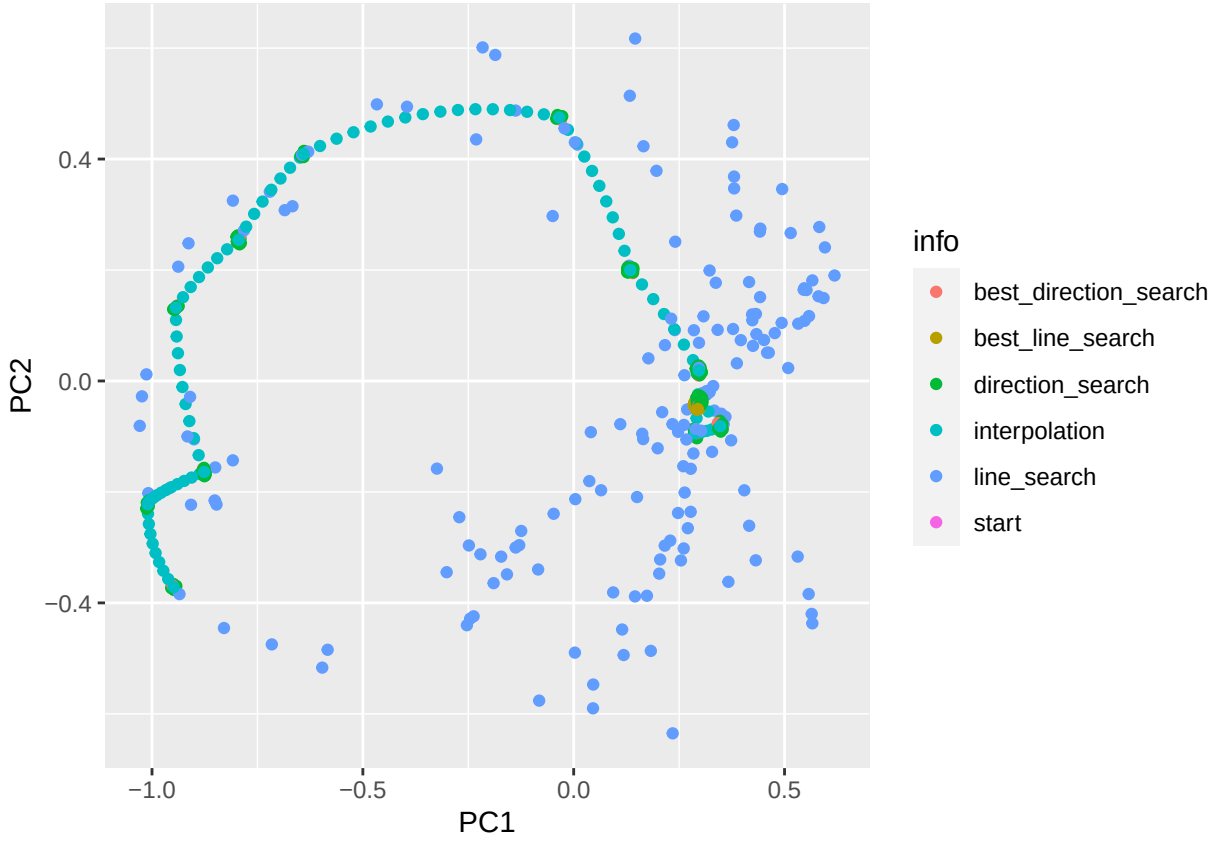


3.2 Explore matrix parameter

The variable or object to explore could sometimes be a more than a scale, for example, in a tour, we would also be interested to explore the relative position of the projection basis in the vector space. This imposes difficulties in visualisation since as human, we are bounded to preceive at most three dimensions. Dimension reduction is reuqired to summarise the projection matrix to a managable number of variable to map to a plot. Principal copmonent analysis is used to extract the direction that reflects most variation in the data.

Example 5 PCA refine the example of matrix parameter exploration

```
holes_1d_geo %>% explore_proj_pca(col = info)
```



4 Real-time animated diagnostic plots

```
holes_1d_geo %>% explore_proj_pca(animate = TRUE, col = info)
```

5 Vis package

Everything is coded up in a package.

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/*