Studying the Performance of the Jellyfish Search Optimiser for the Application of Projection Pursuit

H. Sherry Zhang[a,*], Dianne Cook[b], Nicolas Langrené[c], Jessica Wai Yin Leung[b]

[a] *University of Texas at Austin, Department of Statistics and Data Sciences, Austin, United States, 78751*

[b] *Monash University, Department of Econometrics and Business Statistics, Melbourne, Australia, 3800*

[c] *Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College, Department of Mathematical Sciences, Zhuhai, China, 519087*

Abstract

The projection pursuit (PP) guided tour interactively optimises a criteria function known as the PP index, to explore high-dimensional data by revealing interesting projections. The optimisation in PP can be non-trivial, involving non-smooth functions and optima with small a "squint angle", detectable only from close proximity. To address these challenges, this study investigates the performance of a recently introduced swarm-based algorithm, Jellyfish Search Optimiser (JSO), for optimising PP indexes. The performance of JSO for visualising data is evaluated across various hyper-parameter settings and compared with existing optimisers. Additionally, this work proposes novel methods to quantify two properties of the PP index – smoothness and squintability – that capture the complexities inherent in PP optimisation problems. These two metrics are evaluated with JSO hyper-parameters to determine their effect on JSO success rate. The JSO algorithm has been implemented in the `tourr` package, while functions to calculate smoothness and squintability of indexes are available in the `ferrn` package.

Keywords: projection pursuit, jellyfish search optimiser (JSO), optimisation, grand tour, high-dimensional data, exploratory data analysis

README:

- British English ("American or British usage is accepted, but not a mixture of these")
- No "we" - always third person
- Check if the affiliation information is correct
- word use:
  - Use jellyfish search optimiser, or JSO
  - "hyper-parameter" rather than "hyperparameter"
  - PP index rather than PPI, plural PP indexes
  - datasets rather than data sets

1. Introduction [Nicolas and Jessica]

Projection Pursuit (PP) (Kruskal [1], Friedman and Tukey [2]) is a dimension reduction technique aimed at identifying informative linear projections of data. The method involves optimising a objective function

*Corresponding author

Email addresses: `huize.zhang@austin.utexas.edu` (H. Sherry Zhang), `dicook@monash.edu` (Dianne Cook), `nicolaslangrene@uic.edu.cn` (Nicolas Langrené), `Jessica.Leung@monash.edu` (Jessica Wai Yin Leung)

known as the PP index (e.g. Hall [3], Cook et al. [4], Lee and Cook [5], Loperfido [6], Loperfido [7]), which defines the criteria for what constitutes interesting or informative projections. Given high-dimensional data $X \in \mathbf{R}^{n \times p}$ and an index function $f(\cdot)$, PP finds the orthonormal basis $A \in \mathbf{R}^{p \times d}$ that maximises the index value of the projection, $Y = XA$:

$$\max_{A} \quad f(XA) \quad \text{subject to} \quad A'A = I \tag{1}$$

The index functions can be non-linear and non-convex, hence an effective and efficient optimisation procedure is essential to explore the data landscape and achieving a globally optimal viewpoint of the data.

Significant efforts have been devoted to enhancing the functionality of PP, improving optimisation (e.g. Posse [8], Marie-Sainte et al. [9], Grochowski and Duch [10]) and extending it's use to visually explore high-dimensional data. Cook et al. [11] introduced the PP guided tour, which monitors the optimisation visually to see the data projections leading in and out of the optima. An implementation is available in the R [12] package `tourr` [13]. Zhang et al. [14] illustrated how to diagnose optimisation processes, particularly focusing on the guided tour, and revealed a need to for improved optimisation. While improving the quality of the optimisation solutions in the tour is essential, it is also important to be able to view the data projections as the optimisation progresses. Integrating the guided tour with a global optimisation algorithm that is efficient in finding the global optimal and enables viewing of the projected data during the exploration process is a goal.

Here, the potential for a Jellyfish Search Optimiser (JSO) to be integrated with the projection pursuit guided tour is explored. JSO [15] -[16], inspired by the search behaviour of jellyfish in the ocean, is a swarm-based metaheuristic designed to solve global optimisation problems. Compared to traditional methods, JSO demonstrated stronger search ability and faster convergence, and requires fewer tuning parameters. These practical advantages make JSO a promising candidate for enhancing PP optimisation.

The primary goal of the study reported here is to investigate the performance of JSO in PP optimisation for the guided tour. It is of interest to assess how quickly and closely the optimiser reaches a gobal optima, for various PP indexes that may have differing complexities. To observe the performance of JSO with different types of PP indexes, metrics are introduced to capture specific properties of the index including squintability and smoothness as introduced in Laa and Cook [17]. Metrics for squintability and smoothness are precisely defined here, something that is new for the field. A series of simulation experiments using specially structured datasets and various PP indexes are conducted to assess the behaviour of JSO under different settings and its sensitivity to hyper-parameter choices. The relationship between the JSO performance, hyper-parameter tuning and properties of PP indexes is analysed to provide guidance for practitioners using projection pursuit, and developers of new indexes.

The paper is structured as follows. Section 2 introduces the background of PP guided tour and reviews existing optimisers and index functions in the literature. Section 3 details JSO and introduces metrics that measure different properties of PP indexes. Section 4 outlines the data structures, numerical summaries, and visualisations used in the simulation studies conducted in Section 5 to assess JSO's performance. Section 5 presents two sets of simulation experiments, comparing JSO permformance improvements relative to an existing optimiser, and studies the impact of different PP index properties on the optimisation performance. Section 6 summarises the work and provides suggestions for future directions.

2. Projection pursuit, tours, index functions and optimisation [Di and Sherry]

A tour on high-dimensional data is constructed by geodesically interpolating between pairs of planes. Any plane is described by an orthonormal basis, A_t, where t represents time in the sequence. The term "geodesic" refers to maintaining the orthonormality constraint so that each view shown is correctly a projection of the data. The PP guided tour operates by geodesically interpolating to target planes (projections) which have high PP index values, as provided by the optimiser. The geodesic interpolation means that the viewer sees a

continuous sequence of projections of the data, so they can watch patterns of interest forming as the function is optimised. There are five (unsophisticated) optimisation methods implemented in the `tourr` package:

- `search_geodesic()`: provides a pseudo-derivative optimisation. It searches locally for the best direction, based on differencing the index values for very close projections. Then it follows the direction along the geodesic path between planes, stopping when the next index value fails to increase.
- `search_better()`: is a brute-force optimisation searching randomly for projections with higher index values.
- `search_better_random()`: is essentially simulated annealing [18] where the search space is reduced as the optimisation progresses.
- `search_posse()`: implements the algorithm described in Posse [19].
- `search_polish()`: is a very localised search, to take tiny steps to get closer to the local maximum.

There are several PP index functions available: `holes()` and `cmass()` [4]; `lda_pp()` [20]; `pda_pp()` [5]; `dcor2d()` and `splines2d()` [21]; `norm_bin()` and `norm_kol()` [22]; `slice_index()` [23]. Most are relatively simply defined, for any projection dimension, and implemented because they are relatively easy to optimise. A goal is to be able to incorporate more complex PP indexes, for example based on scagnostics (Wilkinson et al. [24], Wilkinson and Wills [25]).

An initial investigation of PP indexes, and the potential for scagnostics is described in Laa and Cook [17]. To be useful here an optimiser needs to be able to handle index functions which are possibly not very smooth. In addition, because data structures might be relatively fine, the optimiser needs to be able to find maxima that occur with a small squint angle, that can only be seen from very close by. One last aspect that is useful is for an optimiser to return local maxima in addition to global because data can contain many different and interesting features.

3. The jellyfish optimiser and properties of PP indexes [Nicolas and Jessica]

JSO mimics the natural movements of jellyfish, which include passive and active motions driven by ocean currents and their swimming patterns, respectively. In the context of optimization, these movements are abstracted to explore the search space in a way that balances exploration (searching new areas) and exploitation (focusing on promising areas). The algorithm aims to find the optimal solution by adapting the behaviour of jellyfish to navigate towards the best solution over iterations [15].

To solve the optimisation problem embedded in the PP guided tour, a starting projection, an index function and the maximum number of iterations are provided as input. Then, the current projection is evaluated by the index function. The projection is then moved in a direction determined by a random factor, influenced by how far along we are in the optimization process. Occasionally, completely new directions may be taken like a jellyfish might with ocean currents. A new projection is accepted if it is an improvement compared to the current one, rejected otherwise. This process continues and iteratively improves the projection, until the pre-specified maximum number of trials is reached.

Algorithm: Jellyfish Optimizer Pseudo Code

Input: `current_projections`, `index_function`, `trial_id`, `max_trial`
Output: `optimized_projection`
Initialize `current_best` as the projection with the best index value from `current_projections`, and `current_idx` as the array of index function values for each projection in `current_projections`
for each trial_id in 1 to max_tries **do**
    Calculate c_t based on the current_idx and max_trial
    if c_t is greater than or equal to 0.5 **then**
        Define trend based on the current_best and current_projections
        Update each projection towards the trend using a random factor and orthonor-

malisation
else
if a random number is greater than $1 - c_t$ **then**
Slightly adjust each projection with a small random factor (passive)
else
For each projection, compare with a random jelly and adjust towards or away from it (active)
Update the orientation of each projection to maintain consistency
Evaluate the new projections using the index function
if any new projection is worse than the current, revert to the `current_projections` for that case
Determine the projection with the best index value as the new current_best
if trial_id $\geq$ max_trial, print the last best projection **exit**
return the set of projections with the updated current_best as the optimized_projection

The JSO implementation involves several key parameters that control its search process in optimization problems. These parameters are designed to guide the exploration and exploitation phases of the algorithm. While the specific implementation details can vary depending on the version of the algorithm or its application, we focus on two main parameters that are most relevant to our application: the number of jellyfish and the maximum number of trials.

Laa and Cook [17] has proposed five criteria for assessing projection pursuit indexes (smoothness, squintability, flexibility, rotation invariance, and speed). Since not all the properties affects the execution of the optimisation, we choose to focus here on the first two properties, namely *smoothness* (Section 3.1) and *squintability* (Section 3.2), and propose two metrics to evaluate them.

3.1. Smoothness

This subsection proposes a metric for the smoothness of a projection pursuit index.

If a PP index function is evaluated at some random points, as is done at the initialization stage of the jellyfish search optimization, then these random index values can be interpreted as a random field, indexed by a space parameter, namely the random projection angle. This analogy suggests to use this random training sample to fit a spatial model, a simple one being a (spatial) Gaussian process.

The distribution of a Gaussian process is fully determined by its mean function and covariance function. The smoothness property comes into play in the definition of the covariance function: if a PP index is very smooth, then two close projection angles should produce close index values (strong correlation); by contrast, if a PP index is not very smooth, then two close projection angles might give very different index values (fast decay of correlations with respect to distance between angles).

Popular covariance functions are parametric positive semi-definite functions, some of which have a parameter to capture the smoothness of the Gaussian field. In particular, consider the Matérn class of covariance functions, defined by

$$K(u) := \frac{(\sqrt{2\nu}u)^\nu}{\Gamma(\nu)2^{\nu-1}}\mathcal{K}_\nu(\sqrt{2\nu}u) \ ,$$

where $\nu > 0$ is the smoothness parameter and where $\mathcal{K}_\nu$ is the modified Bessel function. The Matérn covariance function can be expressed analytically when ν is a half-integer, the most popular values in the literature being 1/2, 3/2 and 5/2 . The parameter ν, called *smoothness parameter*, controls the decay of the covariance function. As such, it is an appropriate measure of smoothness of a random field.

In our context, we suggest to use this parameter ν as a measure of the smoothness of the index function by fitting a Gaussian process prior with Matérn covariance on a dataset generated by random evaluations of the index function, as done at the initialization stage of the jellyfish search optimization. There exist several R packages, such as GpGp [26] or ExaGeoStatR [27], to fit the hyperparameters of a GP covariance function on data. In this project, we make use of the GpGp package.

After fitting the smoothness parameter $\nu > 0$, its value can be interpreted as follows: the higher ν, the smoother the index function.

3.2. Squintability

In this subsection, a metric to capture the notion of squintability is proposed, and two approaches to compute this metric numerically are detailed.

In the existing literature, it is commonly understood that a large squint angle implies that the objective function value is close to optimal even when we are not very close to the perfect view to see the structure. A small squint angle means that the PP index value improves substantially only when we are very close to the perfect view. As such, low squintability implies rapid improvement in the index value when near the perfect view. For PP, a small squint angle is considered to be undesirable because it means that the optimiser needs to be very close to be able to "see" the optima. Thus, it could be difficult for the optimiser to find the optima. We now propose a mathematical formulation of this intuition.

Let $A \in \mathbf{R}^{p\times 2}$ be the 2D projection basis and A^* the optimal basis achieving the maximum index value for a given projection pursuit problem defined by the shape to find, data dimension, and the index function $f(.)$. The projection distance $d(A, A^*)$, is defined as the Frobenius norm $\|.\|_F$, between the squared difference of these two bases:

$$d(A, A^*) = \|M\|_F = \sum_{ij} M_{ij}^2, \text{ where } M = AA' - A^*A^{*\prime} \ . \tag{2}$$

The index value $f(XA)$ is defined as described in the introduction. We now focus our attention to how the index value $f(XA)$ changes with respect to the projection distance $d(A, A^*)$, over the course of the JSO.

It is expected that for a PP index with high squint angle, the optimization (1) should make substantial progress early on. Conversely, for a PP index with low squint angle, it might take a long while for the optimization to make substantial progress, as the candidate projections would need to be very close to the optimal one for the structure of the index function to be visible enough to be amenable to efficient optimization. This observation suggests that the extreme values of f' (the ones for which $f'' = 0$, assuming that f is twice differentiable), and the projection distances for which these values are attained, should play an important role in the mathematical definition of squintability. Noticing that f is expected to be a decreasing function, we propose the following squintability metric:

$$\varsigma := -4 \min_x f'(x) \times \arg\min_x f'(x) \ . \tag{3}$$

On the one hand, $-\min_x f'(x)$ represents the largest value of the gradient of $-f$. On the other hand, $\arg\min_x f'(x)$ represents the projection distance at which this largest gradient is attained. Based on the above discussion, it is expected that these two values should be both high in the case of high squintability (fast increase in f early on), and both low in the case of low squintability (any substantial increase in f happens very late, close to the optimal angle). This suggest that their product (3) should provide a sensible measure of squintability. The multiplicative constant 4, which can be deemed arbitrary, does not change the interpretation of the squintability metric ς; it is here to adjust the range of values of ς and simplify the explicit formula for ς obtained later on.

To compute the squintability metric (3) in practice, several approaches are possible. The first one is to propose a parametric model for f, and use it to obtain an explicit formula for ς. Numerical experiments suggest a scaled sigmoid shape as described below. Define

$$\ell(x) := \frac{1}{1 + \exp(\theta_3(x - \theta_2))} \ , \tag{4}$$

which is a decreasing logistic function depending on two parameters θ_2 and θ_3 , such that $\ell(\theta_2) = \frac{1}{2}$. Then, define

$$f(x) = (\theta_1 - \theta_4)\frac{\ell(x) - \ell(x_{\max})}{\ell(0) - \ell(x_{\max})} + \theta_4 \ , \tag{5}$$

which depends on three additional parameters, θ_1, θ_2 , and $x_{\max}$, such that $f(0) = \theta_1$ and $f(x_{\max}) = \theta_4$. Under the parametric model (5), the squintability metric (3) can be shown to be equal to

$$\varsigma = \frac{(\theta_1 - \theta_4)\theta_2\theta_3}{\ell(0) - \ell(x_{\max})} \ . \tag{6}$$

In practice, the parameters of this model (5) can be estimated numerically, for example by non-linear least squares, and then used to evaluate ς as in equation (6).

Alternatively, one can estimate (3) in a nonparametric way, for example by fitting f using kernel regression, then numerically estimate the angle at which $-f'$ attains its highest value.

To the best of our knowledge, this is the first attempt to measure mathematically the notion of squintability.

4. Visualisation of jellyfish optimiser

The JSO provides a new swarm-based search strategy to optimise projection pursuit problems. This section outlines two simulation setups to demonstrate the performance of JSO. In the first example, the performance of JSO is compared with an exist optimiser, Creeping Random Search (CRS), and explores how JSO behaves with different combinations of hyper-parameters. The second example studies the effect of optimisation properties defined in Section 3, along with JSO hyper-parameters, on the outcome (success rate) of the optimisation.

The data structure proposed in Zhang et al. [14] is used here for assessing JSO. A sample for the sine-wave structure in 6D data using a spline index function (`splines2d`) is printed below:

```
Rows: 10
Columns: 12
$ index     <chr> "splines2d", "splines2d", "splines2d", "splines2d", "splines~
$ d         <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6
$ n_jellies <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
$ max_tries <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
$ sim       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
$ seed      <int> 3462, 3462, 3462, 3462, 3462, 3462, 3462, 3462, 3462, 3462
$ basis     <list> <<matrix[6 x 2]>>, <<matrix[6 x 2]>>, <<matrix[6 x 2]>>, <<m~
$ index_val <dbl> 0.036664532, 0.005121912, 0.058683104, 0.056527991, 0.015734~
$ method    <chr> "search_jellyfish", "search_jellyfish", "search_jellyfish", ~
$ tries     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
$ loop      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
$ time      <drtn> 4.362133 mins, 4.362133 mins, 4.362133 mins, 4.362133 mins, ~
```

The JSO hyper-parameters used are 50 jellyfishes (`n_jellies`) and a maximum of 50 tries (`max_tries`). The full data contains 50 simulations, indexed by the variable `sim`, with the associated `seeds`. Recorded in columns `basis`, `index_val`, `tries`, `loop` and `time`, are the sampled projection basis, index value, iteration ID, jellyfish ID, and computational time respectively. The basis column records every visited basis before comparing with the current (last) basis. If a visited basis has a smaller index value than the current (last) one, JSO will keep the current basis for the next iteration in the search but still records the visited inferior basis.

4.1. Performance of JSO in pipe-finding problems

The performance of JSO is investigated in two folds: 1) compare it with a commonly used optimiser: the search-better optimiser (SBO) [14, 17], and 2) examine its success rate under different hyper-parameters (number of jellyfishes and maximum number of tries). The projection pursuit problem used is finding the pipe shape using the holes index, investigated by Laa and Cook [17]. Fifty simulations are conducted with both JSO and the SBO, in four dimensions ($d = 6, 8, 10, 12$). JSO uses 100 jellyfishes and 100 maximum number of tries, while the SBO allows a maximum of 1000 samples at each iteration before the algorithm terminates.

In the second experiment, fifty simulations are conducted at each hyper-parameter combination to analyse its effects on the JSO success rate. The hyper-parameters tested include: 1) 20, 50, and 100 jellyfishes, and 2) 50 and 100 maximum number of tries. The success rate is defined as the proportion of simulations that achieves a final index value within 0.05 of the best index value found among all 50 simulations. In the dataset above, the optimal index value is 0.999. Out of the 50 simulations, 44 achieved an index value within [0.949, 0.999], resulting in a success rate of 0.88.

4.2. Factors affecting JSO success rate: optimisation properties and jellyfish hyper-parameters

In addition to the pipe-finding problem, a new shape, sine wave, is investigated in 6D and 8D spaces with six indexes considered: `dcor2d_2`, `loess2d`, `MIC`, `TIC`, `spline`, and `stringy`. Combining with two JSO hyper-parameters, a total of 52 cases is produced, comprising of 24 pipe-finding cases and 28 sine-wave finding cases. For each case, JSO is run fifty times and the success rate is calculated.

Smoothness and squintability are computed for each case, following the procedures outlined in Section 3.1 and Section 3.2 and illustrated in Figure 1 and Figure 2. To calculate smoothness, three hundred random bases are simulated. Index values and projection distance (to the optimal basis) are calculated for each random basis before fitting into a Gaussian process model to obtain the smoothness measure. For squintability, fifty random bases are sampled and interpolated to the optimal basis with a step size of 0.005. Index values and projection distances are calculated for these interpolated bases and the index values are averaged with a bin width of 0.005. The squintability measure is then calculated from fitting a 4-parameter scaled logistic function of index value against projection distance, estimated by non-linear least squares.

Another illustration for smoothness

Figure 1: Illustration of steps to calculate smoothness.

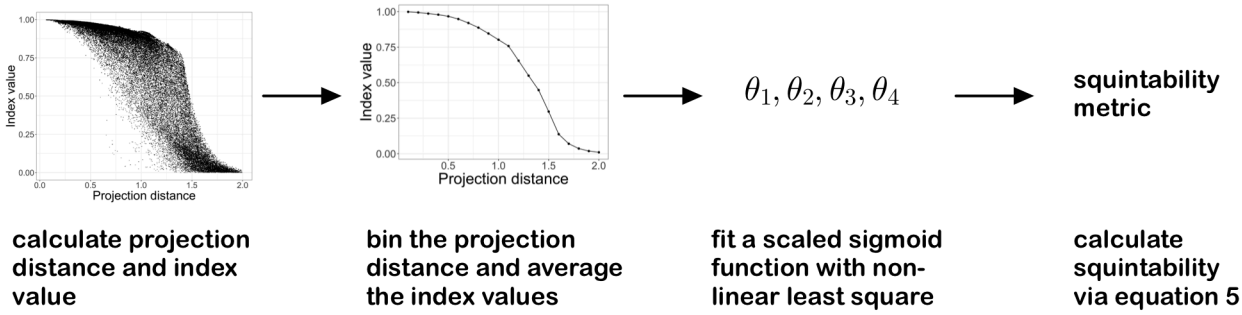

Figure 2: Illustration of steps to calculate squintability. For all the bases, calculate the projection distance to the optimal basis and the corresponding index value. Bin the index value by the projection distance, then claculate the average index value for each bin. Fit the scaled sigmoid function from Equation 3 to the binned index values against the projection distance using non-linear least square. Use Equation 5 to calculate the squintability measure from the estimated θ_1 to θ_4.

5. Results [Di and Sherry]

5.1. Performance of JSO in pipe-finding problems

Figure 3 presents the final projections found by the two optimisers, broken down by 10th quantile, faceted by the data dimension. In the 6-dimensional data scenario, JSO consistently identifies a clear pipe shape. The SBO also finds the pipe shape but with a wide rim, suggesting a further polish search may be required. With increasing dimensions, JSO may not always identify the pipe shape due to random sampling, but it still presents a pipe shape in over 50% of cases. When compared to the SBO JSO reaches higher index values and clearer pipe shapes across all quantiles.

Figure 4 presents the proportion of success for each hyper-parameter combinations. As the number of jellyfishes and maximum tries increase, the success rate also increases. For simpler problems (6 dimensions), small parameter values (20 jellyfishes and a maximum number of tries of 50) can already result in high success rate, while for higher-dimensional problems (8, 10, and 12 dimensions), a combination of 100 jellyfishes and 100 maximum number of tries is necessary for at least half of the jellyfishes to find the projection close to the optimal.

5.2. Factors affecting JSO success rate: optimisation properties and jellyfish hyper-parameters

Table 1 presents the parameters estimated from the Gaussian process (variance, range, smooth, and nugget) and the scaled logistic function (θ_1 to θ_4) for each case. The column "smooth" is used as the smoothness measure and the column "squint" is calculated as equation 6 as squintability. The low squint value of MIC and TIC index is due to its convex shape of the index value against the projection distance, as shown in Figure 5. Comparing to other concave shapes, the maximum first derivative happens at a smaller projection distance (θ_2), requiring the optimiser to get closer to the optimal to see a significant change in the index value, hence a small squintability measure.

Table 2: The first 7 rows of the datasets processed for modelling.

Index	D	Smoothness	Squintability	Num. of jellyfish	Max. Num. of tries	Prop. of success	Time
MIC	6	2.457	1.257	20	50	0.12	2.479 secs
MIC	6	2.457	1.257	20	100	0.24	8.950 secs
MIC	6	2.457	1.257	50	50	0.52	5.651 secs
MIC	6	2.457	1.257	50	100	0.64	13.223 secs
MIC	6	2.457	1.257	100	50	0.76	19.453 secs
MIC	8	2.636	0.772	20	50	0.08	2.566 secs
MIC	8	2.636	0.772	20	100	0.08	4.960 secs

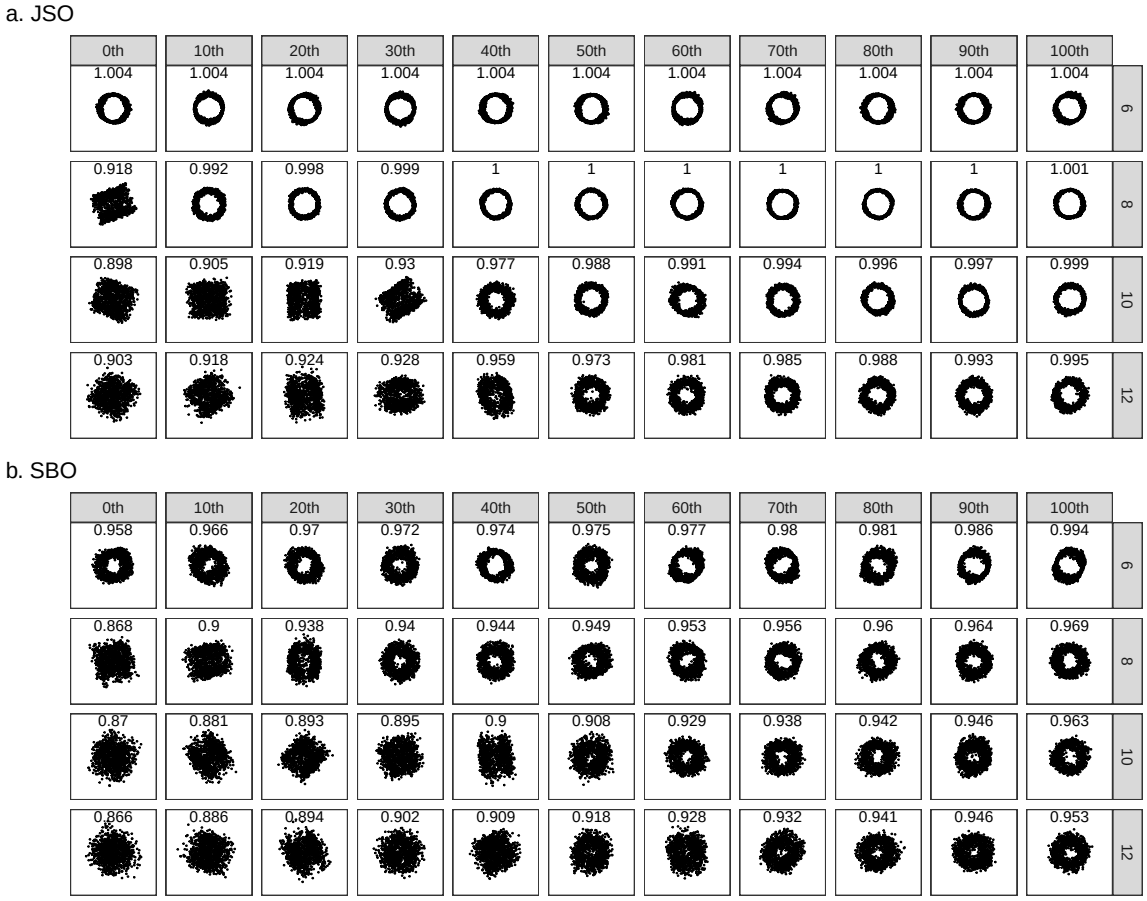

Figure 3: Projections found by the JSO and SBO at each 10th quantile across 50 simulations. The projection pursuit problem is to find the pipe shape using the holes index in the 6, 8, 10, and 12-dimensional spaces. The JSO uses 100 jellyfishes and a maximum number of tries of 100. The SBO uses a maximum of 1000 tries in each step of random sampling step before the algorithm terminates. In the 6-D data space, JSO always finds a clear pipe shape while the SBO also finds the pipe shape but with a wide rim. At higher data dimensions, JSO finds a higher index value and a clearer pipe shape across all the quantiles than the SBO.

Table 1: Parameters estimated from the Gaussian process (including variance, range, smoothness, and nugget) and scaled logistic function (θ_1 to θ_4) for the pipe-finding and sine-wave finding problems. The column "smooth" and "squint" represent the smoothness and squintability measures.

	shape	index	d	variance	range	smooth	nugget	θ_1	θ_2	θ_3	θ_4	squint
1	pipe	holes	6	0.002	0.371	2.355	0.221	1.015	0.860	3.368	0.018	3.053
2	pipe	holes	8	0.000	0.178	2.189	0.820	1.014	0.869	3.264	0.029	2.960
3	pipe	holes	10	0.000	0.107	2.192	3.027	1.016	0.885	3.151	0.022	2.948
4	pipe	holes	12	0.000	0.148	2.295	1.581	1.011	0.878	3.345	0.004	3.117
5	sine	MIC	6	0.016	0.092	2.457	0.083	0.894	0.571	1.623	-0.024	1.257
6	sine	MIC	8	0.016	0.081	2.636	0.084	0.932	0.328	1.314	-0.030	0.772
7	sine	TIC	6	0.124	0.110	2.444	0.086	0.951	0.536	1.719	-0.027	1.322
8	sine	TIC	8	0.125	0.099	2.475	0.085	0.949	0.564	1.723	-0.028	1.369
9	sine	dcor2d	6	0.034	0.167	2.663	0.114	0.954	1.039	2.742	-0.019	2.947
10	sine	loess2d	6	0.079	0.336	1.988	0.309	1.016	1.039	2.648	0.080	2.756
11	sine	splines2d	6	0.042	0.242	2.537	0.102	1.014	1.051	2.730	-0.009	3.118
12	sine	stringy	6	0.000	0.007	1.536	38.166	1.045	0.011	254.734	0.053	2.957

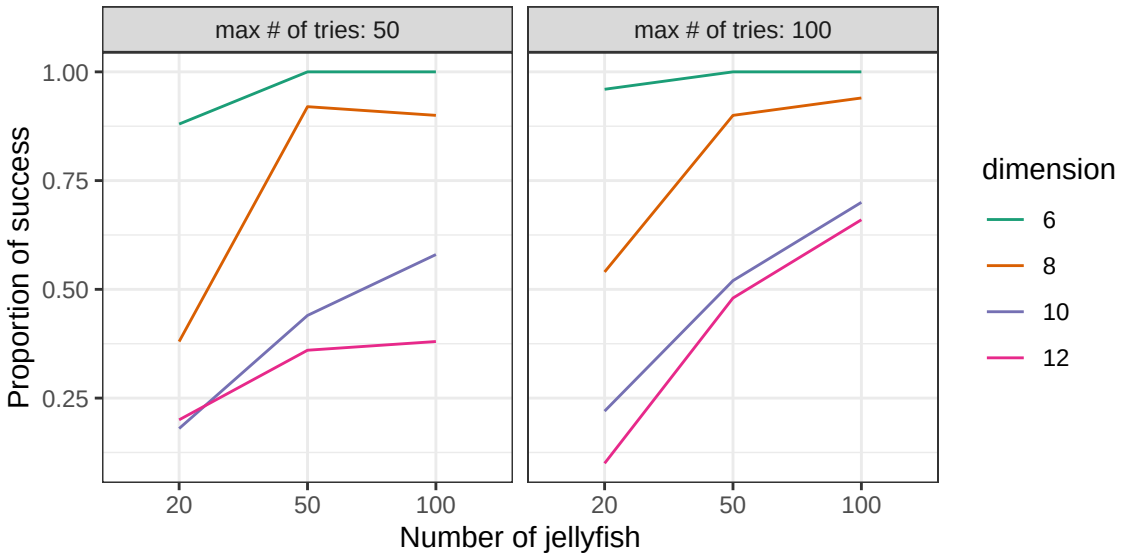

Figure 4: Proportion of simulations reaches near-optimal index values in the pipe-finding problem using the holes index. The proportion is calculated based on the number of simulations, out of 50, that achieve an index value within 0.05 of the best-performing simulation. As the dimensionality increases, the proportion of simulations reaching the optimal index value increases.

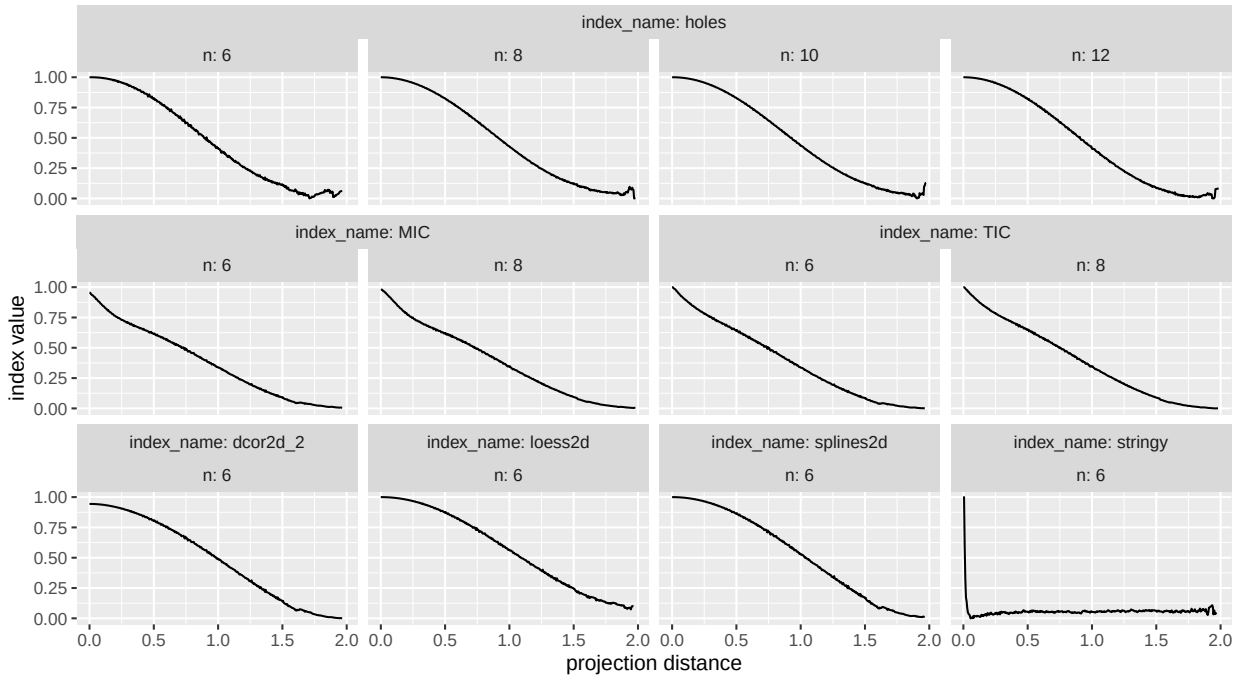

Figure 5: Index values versus projection distance for the 12 pipe/sine-wave finding problem, after the binning procedure for calculating the squintability measure. The index values, averaged at bin width of 0.005, are scaled from 0 to 1 for comparison (holes, TIC, and stringy). The `MIC` and `TIC` index curves are convex while others are concave. The stringy curve shows an instantaneous jump to the optimum when aproaching the best basis.

Table 3: Model estimates of proportion of jellyfish success on optimisation properties and jellyfish hyper-parameters from the generalised linear model with a binomial family and a logit link function. The variable smoothness, squintability, number of jellyfish and maximum number of tries are positively associated with JSO success rate while data dimension and being flagged as long runtime are negatively associated with the success rate. The variable squintability and dimension are significant, suggesting their importance relative to jellyfish hyper-parameters in the optimisation success.

term	estimate	std.error	statistic	p.value
Intercept	-4.523	5.332	-0.848	0.396
Smoothness	1.195	1.915	0.624	0.533
Squintability	2.060	0.684	3.011	0.003
dimension (d)	-0.628	0.255	-2.462	0.014
long time	-0.874	1.286	-0.680	0.497
number of jellyfish	0.216	0.127	1.701	0.089
maximum number of tries	0.113	0.150	0.752	0.452

Table 2 combines all the variables calculated above (JSO success rate, hyper-parameters, and optimisation properties) into one table. Three pre-processing steps are applied to the data: 1) scaling the JSO hyper-parameters by a factor of 10 for interpretation, 2) creating a new binary variable `long_time` to indicate cases with an average run time over 30 seconds, and 3) encoding the success rate for the stringy case as 0, because upon visual inspection of the final projection, none of the 50 simulations identified the sine-wave shape. A generalised linear model with a binomial family and a logit link function is used to fit the data and Table 3 presents the model outputs. The model suggests that JSO success rate is positively associated with the two hyper-parameters, as well as with the optimisation properties – smoothness and squintability. Specifically, using 10 more jellyfishes and 10 more tries will increase the odd ratio of success by 24.11% and 11.93%, respectively. However, being flagged with long runtime and an increase of data dimension will reduce the success rate by 41.72% and 53.36%, respectively. The variable `squintability` and `dimension` are significant, suggesting their importance relative to JSO hyper-parameters in the optimisation success. Users should consider increasing the number of jellyfish and the number of tries to improve the success rate, until JSO exceeds a 30-second runtime.

6. Conclusion [Di and Sherry]

- Summarise the result of the simulation: JSO immense improvement on current methods, …
- JSO for a guided tour, what changes are needed? Do we follow one tentacle, do we perform off-line and then pull tentacles out to watch, how does it integrate with ferrn

References

[1] J. B. Kruskal, Toward a practical method which helps uncover the structure of a set of observations by finding the line transformation which optimizes a new 'index of condensation', in: R. C. Milton, J. A. Nelder (Eds.), Statistical Computation, Academic Press, New York, 1969, pp. 427–440.

[2] J. H. Friedman, J. W. Tukey, A Projection Pursuit Algorithm for Exploratory Data Analysis, IEEE Transactions on Computing C 23 (1974) 881–889.

[3] P. Hall, On polynomial-based projection indices for exploratory projection pursuit, The Annals of Statistics 17 (1989) 589–605. URL: https://doi.org/10.1214/aos/1176347127.

[4] D. Cook, A. Buja, J. Cabrera, Projection pursuit indexes based on orthonormal function expansions, Journal of Computational and Graphical Statistics 2 (1993) 225–250. URL: https://doi.org/10.2307/1390644.

[5] E.-K. Lee, D. Cook, A projection pursuit index for large p small n data, Statistics and Computing 20 (2010) 381–392. URL: https://doi.org/10.1007/s11222-009-9131-1.

[6] N. Loperfido, Skewness-based projection pursuit: A computational approach, Computational Statistics and Data Analysis 120 (2018) 42–57. doi:https://doi.org/10.1016/j.csda.2017.11.001.

[7] N. Loperfido, Kurtosis-based projection pursuit for outlier detection in financial time series, The European Journal of Finance 26 (2020) 142–164. doi:https://doi.org/10.1080/1351847X.2019.1647864.

[8] C. Posse, Projection pursuit exploratory data analysis, Computational Statistics & Data Analysis 20 (1995) 669–687. URL: https://www.sciencedirect.com/science/article/pii/0167947395000028. doi:https://doi.org/10.1016/0167-9473(95)00002-8.

[9] S. L. Marie-Sainte, A. Berro, A. Ruiz-Gazen, An efficient optimization method for revealing local optima of projection pursuit indices, in: M. Dorigo, M. Birattari, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. M. Gambardella, R. Groß, E. Şahin, H. Sayama, T. Stützle (Eds.), Swarm Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 60–71.
[10] M. Grochowski, W. Duch, Fast projection pursuit based on quality of projected clusters, in: A. Dobnikar, U. Lotrič, B. Šter (Eds.), Adaptive and Natural Computing Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 89–97.
[11] D. Cook, A. Buja, J. Cabrera, C. Hurley, Grand tour and projection pursuit, Journal of Computational and Graphical Statistics 4 (1995) 155–172. URL: https://doi.org/10.1080/10618600.1995.10474674.
[12] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2023. URL: https://www.R-project.org/.
[13] H. Wickham, D. Cook, H. Hofmann, A. Buja, tourr: An R package for exploring multivariate data with projections, Journal of Statistical Software 40 (2011) 1–18. URL: http://doi.org/10.18637/jss.v040.i02.
[14] H. S. Zhang, D. Cook, U. Laa, N. Langrené, P. Menéndez, Visual diagnostics for constrained optimisation with application to guided tours, The R Journal 13 (2021) 624–641. doi:10.32614/RJ-2021-105.
[15] J.-S. Chou, D.-N. Truong, A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, Applied Mathematics and Computation 389 (2021) 125535. doi:10.1016/j.amc.2020.125535.
[16] K. Rajwar, K. Deep, S. Das, An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges, Artificial Intelligence Review (2023) 1–71. doi:10.1007/s10462-023-10470-y.
[17] U. Laa, D. Cook, Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics, Computational Statistics 35 (2020) 1171–1205. doi:10.1007/s00180-020-00954-8.
[18] D. Bertsimas, J. Tsitsiklis, Simulated Annealing, Statistical Science 8 (1993) 10 – 15. URL: https://doi.org/10.1214/ss/1177011077. doi:10.1214/ss/1177011077.
[19] C. Posse, Projection pursuit exploratory data analysis, Computational Statistics and Data Analysis 20 (1995) 669–687. URL: https://www.sciencedirect.com/science/article/pii/0167947395000028. doi:https://doi.org/10.1016/0167-9473(95)00002-8.
[20] E. Lee, D. Cook, S. Klinke, T. Lumley, Projection pursuit for exploratory supervised classification, Journal of Computational and Graphical Statistics 14 (2005) 831–846. URL: https://doi.org/10.1198/106186005X77702.
[21] K. Grimm, Kennzahlenbasierte Grafikauswahl, doctoral thesis, Universität Augsburg, 2016.
[22] P. J. Huber, Projection pursuit, Ann. Statist. 13 (1985) 435–475. URL: https://doi.org/10.1214/aos/1176349519. doi:10.1214/aos/1176349519.
[23] A. B. Ursula Laa, Dianne Cook, G. Valencia, Hole or grain? a section pursuit index for finding hidden structure in multiple dimensions, Journal of Computational and Graphical Statistics 31 (2022) 739–752. URL: https://doi.org/10.1080/10618600.2022.2035230.
[24] L. Wilkinson, A. Anand, R. Grossman, Graph-theoretic scagnostics, in: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005., 2005, pp. 157–164. doi:10.1109/INFVIS.2005.1532142.
[25] L. Wilkinson, G. Wills, Scagnostics distributions, Journal of Computational and Graphical Statistics 17 (2008) 473–491. URL: https://doi.org/10.1198/106186008X320465. doi:10.1198/106186008X320465. arXiv:https://doi.org/10.1198/106186008X320465.
[26] J. Guinness, M. Katzfuss, Y. Fahmy, GpGp: fast Gaussian process computation using Vecchia's approximation, R package, 2021.
[27] S. Abdulah, Y. Li, J. Cao, H. Ltaief, D. Keyes, M. Genton, Y. Sun, Large-scale environmental data science with ExaGeoStatR, Environmetrics 34 (2023) e2770. doi:https://doi.org/10.1002/env.2770.