Frame to frame interpolation for high-dimensional data visualisation using the woylier package

by Zoljargal Batsaikhan, Dianne Cook, and Ursula Laa

Abstract The woylier package implements tour interpolation paths between frames using Givens rotations. This provides an alternative to the geodesic interpolation between planes currently available in the tourr package. Tours are used to visualise high-dimensional data and models, to detect clustering, anomalies and non-linear relationships. Frame-to-frame interpolation can be useful for projection pursuit guided tours when the index is not rotationally invariant. It also provides a way to specifically reach a given target frame. We demonstrate the method for exploring non-linear relationships between currency cross-rates.

Introduction

When data has up to three variables, visualization is relatively intuitive, while with more than three variables, we face the challenge of visualizing high dimensions on 2D displays. This issue was tackled by the *grand tour* (Asimov, 1985) which can be used to view data in more than three dimensions using linear projections. It is based on the idea of rotations of a lower-dimensional projection in high-dimensional space. The grand tour allows users to see dynamic low-dimensional (typically 2D) projections of higher dimensional space. Originally, Asimov's grand tour presents the viewer with an automatic movie of projections with no user control. Since then new work has added interactivity to the tour, giving more control to users (Buja et al., 2005). New variations include the manual (Cook and Buja, 1997) or radial tour (Laa et al., 2023), little tour, guided tour (Cook et al., 1995), local tour, and planned tour. These are different ways of selecting the sequence of projection bases for the tour, for an overview see Lee et al. (2022).

The guided tour combines projection pursuit with the grand tour and it is implemented in the **tourr** package (Wickham et al., 2011). Projection pursuit is a procedure used to locate the projection of high-to-low dimensional space that should expose the most interesting feature of data, originally proposed in Kruskal (1969). It involves defining a criterion of interest, a numerical objective function that indicates the interestingness of each projection, and an optimization for selecting planes with increasing values of the function. In the literature, a number of such criteria have been developed based on clustering, spread, and outliers.

A tour path is a sequence of projections and we use an interpolation to produce small steps simulating a smooth movement. The current implementation of tour in **tourr** package uses geodesic interpolation between planes. The geodesic interpolation path is the locally shortest path between planes with no within-plane spin (see Buja et al. (2004) for more details). As a result, the rendered target plane could be a within-plane rotation of the target plane originally specified. This is not a problem when the structure we are looking for can be identified from any rotation. However, even simple associations in 2D, such as the calculated correlation between variables, can be very different when the basis is rotated.

Most projection pursuit indexes, particularly those provided by the **tourr** are rotationally invariant. However, there are some projection pursuit index where the orientation of frames does matter. One example is the splines index proposed by Grimm (2016). The splines index computes a spline model for the two variables in a projection, in order to measure non-linear association. It compares the variance of the response variable to the variance of residuals, and the functional dependence is stronger when the index value is larger. It can be useful to detect non-linear relationships in high-dimensional data. However, its value will change substantially if the projection is rotated within the plane (Laa and Cook, 2020). The procedure in Grimm (2016) was less affected by the orientation because it considered only pairs of variables, and it selects the maximum value found when exchanging which variable is considered as predictor and response variable.

Figure 1 illustrates the rotational invariance problem for a modified spline2D index, where we always consider the horizontal direction as the predictor variable, and the vertical direction as the response. Thus, our modified index computes the splines on one orientation, exaggerating the rotational variability. The example data was simulated to follow a sine curve and the modified splines index is calculated on different within-plane rotations of the data. Although they have the same structure, the index values vary greatly.

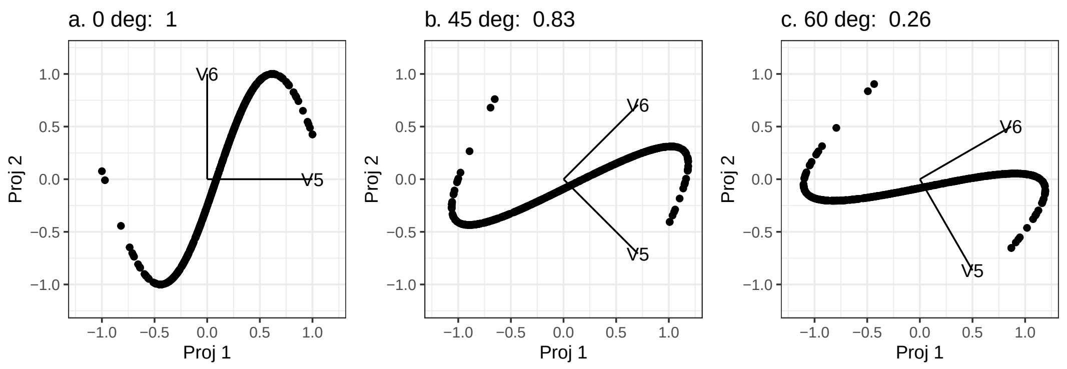


Figure 1: The impact of rotation on a spline index that is NOT rotation invariant. The index value for different within-plane rotations take very different values: (a) original projection has maximum index value of 1.00, (b) axes rotated 45^o drops index value to 0.83, (c) axes rotated 60^o drops index to a very low 0.26. Geodesic interpolation between planes will have difficulty finding the maximum of an index like this because it is focused only on the projection plane, not the frame defining the plane.

The lack of rotation invariance of the splines index raises complications in the optimisation process in the projection-pursuit guided tour as available in the **tourr**. Fixing this is the motivation of this work. The goal with the frame-to-frame interpolation is that optimisation would find the best within-plane rotation, and thus appropriately optimize the index.

A few alternatives to geodesic interpolation were proposed by Buja et al. (2005) including the decomposition of orthogonal matrices, Givens decomposition, and Householder decomposition. The purpose of the **woylier** package is to implement the Givens paths method in R. This algorithm adapts the Given's matrix decomposition technique which allows the interpolation to be between frames rather than planes.

This article is structured as follows. The next section provides the theoretical framework of the Givens interpolation method followed by a section about the implementation in R. The method is applied to search for nonlinear associations between currency cross-rates.

Background

The tour method of visualization shows a movie that is an animated high-to-low dimensional data rotation. It is a one-parameter (time) family of static projections. Algorithms for such dynamic projections are based on the idea of smoothly interpolating a discrete sequence of projections (Buja et al., 2005).

The topic of this article is the construction of the paths of projections. The interpolation of these paths can be compared to connecting line segments that interpolate points in Euclidean space. Interpolation acts as a bridge between a continuous animation and the discrete choice of sequences of projections.

The interpolating paths of plane versus frames

The **tourr** package implements the locally shortest (geodesic) interpolation of planes. The pitfall of this interpolation method is that it does not account for rotation variability within the plane. Therefore, the interpolation of frames is required when the the orientation of projections matters. If the rendering on a frame and on the rotated version of the frame yields the same visual scenes, it means the orientation does not matter. Figure 2 shows an example where this is not the case.

The orientation of the frames could be important when a non-linear projection pursuit index function is used in the guided tour. This is illustrated by the different index values shown in Figure 2, as well as the spline index for the sine curve in Figure 1.

Before continuing with the interpolation algorithms, we need to define the notations.

- Let the p be the dimension of original data and d be the dimension onto which the data is being projected.
- A frame F is defined as a $p \times d$ matrix with pairwise orthogonal columns of unit length that satisfies
$$F^T F = I_d,$$
where I_d is the identity matrix in d dimensions.

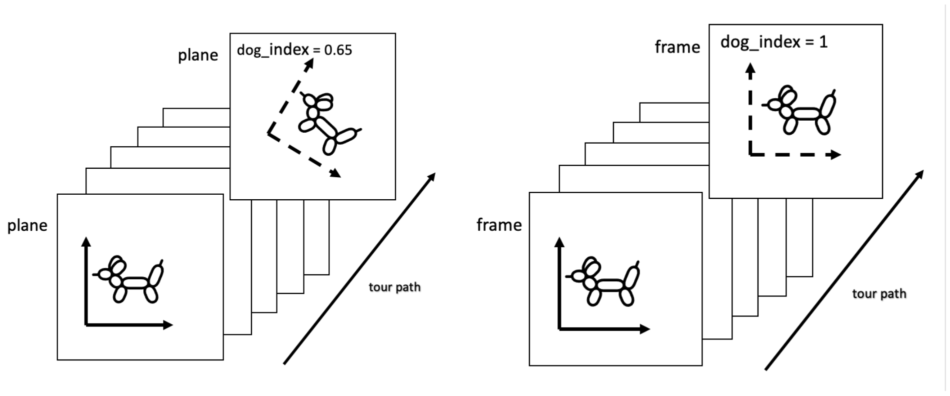


Figure 2: Plane to plane interpolation (left) and Frame to frame interpolation (right). We used dog index for illustration purposes. For some non-linear index orientation of data could affect the index.

- Paths of projections are given by continuous one-parameter families $F(t)$ where $t \in [a, z]$ represents time. We denote the starting frame (at time a) by $F_a = F(a)$ and target frame (at time z) by $F_z = F(z)$. Usually, F_z is the target frame that has been chosen according to the selected tour method. While a grand tour chooses target frames randomly, the guided tour chooses the target frame by optimizing the projection pursuit index. Interpolation methods are used to find the path that moves from F_a to F_z.

Preprojection algorithm

In order to make the interpolation algorithm simple, we carry out a "preprojection" step. The purpose of preprojection is to find the subspace that the interpolation path, $F(t)$, is traversing. In other words, the preprojection step makes sure the interpolation path between two frames F_a and F_z is limited to the part of the space related to F_a and F_z. Simply, a prepojection algorithm is defining the joint subspace of F_a and F_z.

The procedure starts with forming an orthonormal basis by applying Gram-Schmidt to F_z with regards to F_a, i.e. we find the $p \times d$ matrix that contains the component of F_z that is orthogonal to F_a. We denote this orthonormal basis by $F_\star$. Then we build the preprojection basis B by combining F_a and $F_\star$ as follows:

$$B = (F_a, F_\star)$$

The dimension of the resulting orthonormal basis, B, is $p \times 2d$.

Then, we can express the original frames in terms of this basis:

$$F_a = BW_a, F_z = BW_z$$

The interpolation problem is then reduced to the construction of paths of frames $W(t)$ that interpolate between the preprojected frames W_a and W_z. By construction W_a is a $2d \times d$ matrix of 1s and 0s. This is an important characteristic for our interpolation algorithm of choice, the Givens interpolation.

Givens interpolation path algorithm

A rotation matrix is a transformation matrix used to perform a rotation in Euclidean space. The matrix that rotates a 2D plane by an angle θ looks like this:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

If the rotation is in the plane of two selected variables, it is called a Givens rotation. Let's denote those 2 variables as i and j. The Givens rotation is used for introducing zeros, for example when computing the QR decomposition of a matrix in linear algebra problems.

The interpolation method in the **woylier** package is based on the fact that in any vector of a matrix, one can zero out the i-th coordinate with a Givens rotation (Golub and Loan, 1989) in the (i, j)-plane for any $j \neq i$. This rotation affects only coordinates i and j and leaves all other coordinates unchanged. Sequences of Givens rotations can map any orthonormal d-frame F in p-space to the standard d-frame

$$E_d = ((1, 0, 0, ...)^T, (0, 1, 0, ...)^T, ...).$$

The interpolation path construction algorithm from starting frame F_a to target frame F_z is illustrated below. The example is for $p = 6$ and $d = 2$.

1. Construct preprojection basis B by orthonormalizing F_z with regards tp F_a with Gram-Schmidt.

In our example, F_a and F_z are $p \times d$ or 6×2 matrices that are orthonormal. The preprojection basis B is $p \times 2d$ matrix that is 6×4.

2. Get the preprojected frames using the preprojection basis B.

$$W_a = B^T F_a = E_d$$

and

$$W_z = B^T F_z$$

In our example, W_a looks like:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

W_z is an orthonormal $2d \times d$ matrix that looks like:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}$$

3. Then, we can construct a sequence of Givens rotations that maps W_z to W_a with such angles that makes one element zero at a time:

$$W_a = R_m(\theta_m)...R_2(\theta_2)R_1(\theta_1)W_z$$

At each rotation, the angle θ_i that zero out the next coordinate of a plane is calculated. Here $m = \sum_{k=1}^{d}(2d - k)$, so when $d = 2$ we need $m = 5$ rotations with 5 different angles, each making one element 0. For example, the first rotation angle θ_1 is an angle in radiant between $(1, 0)$ and (a_{11}, a_{21}). This rotation matrix would make element a_{21} zero:

$$R_1(\theta_1) = G(1, 2, \theta_1) = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In the same way we zero out the elements a_{31} and a_{41}. Because of the orthonormality this means that now $a_{11} = 1$ and that $a_{12} = 0$. We thus need only two more rotations to zero out a_{32} and a_{42}.

4. The inverse mapping is obtained by reversing the sequence of rotations with the negative of the angles, we starts from the starting basis and end at the target basis.

$$R(\theta) = R_1(-\theta_1)...R_m(-\theta_m),\ W_z = R(\theta)W_a$$

Performing these rotations would go from the starting frame to the target frame in one step. But we want to do it sequentially in a number of steps so interpolation between frames looks dynamic.

5. Next we include the time parameter, t, so that the interpolation process can be rendered in the movie-like sequence. We break each θ_i into the number of steps, n_{step}, that we want to take from the starting frame to the target frame, which means it moves by equal angle in each step. Here n_{step} should vary based on the angular distance between F_a and F_z, such that when watching a sequence of interpolations we have a fixed angular speed.
6. Finally, we reconstruct our original frames using B. This reconstruction is done at each step of interpolation so that we have the interpolated path of frames as the result.

$$F_t = BW_t$$

At each time t we can project the data using the frame F_t.

Implementation

We implemented each steps in the Givens interpolation path algorithm in separate functions and combined them in the `givens_full_path()` function for deriving the full set of F_t. The same functions are used to integrate the Givens interpolation with the `animate()` functions of the **tourr** package. Here is the input and output of each functions and it's descriptions, functions to use with `animate()` are described separately below.

name	description	input	output
`givens_full_path(Fa, Fz, nsteps)`	Construct full interpolated frames.	Starting and target frame (Fa, Fz) and number of steps	An array with nsteps matrix. Each matrix is interpolated frame in between starting and target frames.
`preprojection(Fa, Fz)`	Build a d-dimensional pre-projection space by orthonormalizing Fz with regard to Fa.	Starting and target frame (Fa, Fz)	B pre-projection p x 2D matrix
`construct_preframe(Fa, B)`	Construct preprojected frames.	A frame and the pre-projection p x 2D matrix	Pre-projected frame in pre-projection space
`row_rot(a, i, k, theta)`	Performs Givens rotation .	A frame and the pre-projection p x 2D matrix	theta angle rotated matrix a
`calculate_angles(Wa, Wz)`	Calculate angles of required rotations to map Wz to Wa.	Preprojected frames (Wa, Wz)	Names list of angles
`construct_moving_frame(Wt, B)`	Reconstruct interpolated frames using pre-projection.	Pre-projection matrix B, Each frame of givens path	A frame of on a step of interpolation

When using **tourr** we typically want to run a tour live, such that target selection and interpolation are interleaved, and the display will show the data for each frame F_t in the interpolation path. The implementation in **tourr** was described in Wickham et al. (2011), and with the **woylier** we provide functions to use the Givens interpolation with the grand tour, guided tour and planned tour. To do this we rely primarily on the function `givens_info()` which calls the functions listed in the Table above and collects all necessary information for interpolating between a given starting and target frame. The function `givens_path()` then defines the interpolation and can be used instead of `tourr::geodesic_path()`. Wrapper functions for the different tour types are available to use this interpolation, since in the `tourr::grand_tour()` and other path functions this is fixed to use the geodesic interpolation. Calling a grand tour with Givens interpolation for direct animation will then use:

```
tourr::animate_xy(<data>, tour_path = woylier::grand_tour_givens())
```

Comparison of geodesic interpolation and Givens interpolation

The `givens_full_path()` function returns the intermediate interpolation step projections in given number of steps. The code chunk below demonstrates the interpolation between 2 random basis in 5 steps.

```
set.seed(2022)
p <- 6
base1 <- tourr::basis_random(p, d=2)
base2 <- tourr::basis_random(p, d=2)

base1
```

```
#>             [,1]        [,2]
#> [1,]  0.24406482 -0.57724655
#> [2,] -0.31814139  0.06085804
#> [3,] -0.24334450  0.38323969
#> [4,] -0.39166263  0.01182949
#> [5,] -0.08975114  0.59899558
#> [6,] -0.78647758 -0.39657839
```

```
base2
```

```
#>             [,1]        [,2]
#> [1,] -0.64550501 -0.17034478
#> [2,]  0.06108262  0.87051018
#> [3,] -0.03470326  0.26771612
#> [4,] -0.05281183  0.25452167
#> [5,] -0.43004248  0.27472455
#> [6,] -0.62502981  0.03560765
```

```
givens_full_path(base1, base2, nsteps = 5)
```

```
#> , , 1
#>
#>             [,1]        [,2]
#> [1,]  0.24406482 -0.57724655
#> [2,] -0.31814139  0.06085804
#> [3,] -0.24334450  0.38323969
#> [4,] -0.39166263  0.01182949
#> [5,] -0.08975114  0.59899558
#> [6,] -0.78647758 -0.39657839
#>
#> , , 2
#>
#>             [,1]        [,2]
#> [1,]  0.02498501 -0.57102411
#> [2,] -0.26080833  0.26278410
#> [3,] -0.19820064  0.40434178
#> [4,] -0.35542927  0.08341593
#> [5,] -0.14433023  0.57626698
#> [6,] -0.86308174 -0.31951242
#>
#> , , 3
#>
#>            [,1]       [,2]
#> [1,] -0.1909937 -0.5290164
#> [2,] -0.1874044  0.4550600
#> [3,] -0.1459678  0.4046873
#> [4,] -0.2970111  0.1522888
#> [5,] -0.2003186  0.5261305
#> [6,] -0.8824688 -0.2197674
#>
#> , , 4
#>
#>             [,1]       [,2]
#> [1,] -0.38527579 -0.4457635
#> [2,] -0.10411664  0.6258684
#> [3,] -0.09577045  0.3811614
#> [4,] -0.22183655  0.2089977
#> [5,] -0.26412984  0.4533801
#> [6,] -0.84414115 -0.1137724
#>
#> , , 5
#>
#>             [,1]        [,2]
#> [1,] -0.54115467 -0.32350096
#> [2,] -0.01855341  0.76518462
#> [3,] -0.05630484  0.33422743
#> [4,] -0.13748432  0.24504604
#> [5,] -0.34020920  0.36617868
#> [6,] -0.75431619 -0.02150119
#>
#> , , 6
#>
#>             [,1]        [,2]
#> [1,] -0.64550501 -0.17305000
#> [2,]  0.06108262  0.86649508
#> [3,] -0.03470326  0.26851774
#> [4,] -0.05281183  0.25487107
#> [5,] -0.43004248  0.27511042
#> [6,] -0.62502981  0.03766958
```

In this section, we illustrate the use of *givens_full_path()* function by plotting the interpolated path between 2 frames. This also a way of checking if interpolated path is moving in equal size at each step.

For plotting the interpolated path of projections, we used **geozoo** package (Schloerke, 2016). 1D projection is plotted on unit sphere, while 2D projection is visualized on torus. The points on the surface of sphere and torus shape are randomly generated by functions from the **geozoo** package.

Interpolated paths of 1D projection

1D projection of data in high dimension linear combination of data that is normalized. Therefore, we can plot the point on the surface of a hypersphere. Figure 3 shows the Givens interpolation steps between 2 points, 1D projection of 6D data that is.

Interpolated paths of 2D projection

In case of 2D projections, we can plot the interpolated path between 2 frames on the surface of torus. Torus can be seen as crossing of 2 circles that are orthonormal. Figure **??** shows the Givens interpolation steps in 2D projection of 6D data.

For a 2D projection the same target plane is found when rotating the basis within the plane, or when reflecting across one of the two directions (the reflected basis can then also be rotating). This means the space of target bases is constrained to two circles in the high-dimensional space, and these are disconnected because the reflection corresponds to a jump along the torus surface. In the high-dimensional space we can imagine the reflection as flipping over the target plane, resulting in a reflection of the normal vector on the plan. While the Givens interpolation will reach the exact basis, a geodesic interpolation towards the same target plane can land anywhere along those two circles, depending on the starting basis in the interpolation.

In this section, we used simulated data for comparing geodesic and Givens interpolation paths. The data has 6 variables and 500 observations. The variable 5 and 6 has sine structure and the remaining variables are randomly generated from normal distribution. The sine is non-linear structure and can be detected using splines2d index.

Figure **??** shows the Geodesic and Givens interpolation to target frame where the two variables forms sine curve.

Data application

This section describes the application of Givens interpolation path with guided tour to explore non-linear association in multivariate data.

We have cross-rates for currencies relative to the US dollar. A cross-rate is an exchange rate between two currencies computed by reference to a third currency, usually the US dollar.

The data was extracted from openexchangerates and contains cross-rate for ARS, AUD, EUR, JPY, KRW, MYR between 2019-11-1 to 2020-03-31. Figure 4 shows how the currencies changed relative to USD over the time period. We see some collective behavior in March of 2020 with EUR and JPY increasing in a similar manner, and smaller currencies decreasing in value. This could be understood as a consequence of flight-to-quality at this uncertain times.

We are interested in capturing this relations over time, and from the time series visualization we expect that we can capture the main dynamics in a two-dimensional projection from the six-dimensional space of currencies. Thus we start from $p = 6$ (the different currencies) and $n = 152$ the number of days in our sample. We expect that a projection onto $d = 2$ dimensions should capture the relation between the two groups of currencies mentioned above, and this should be identified within the noise of the random fluctuations.

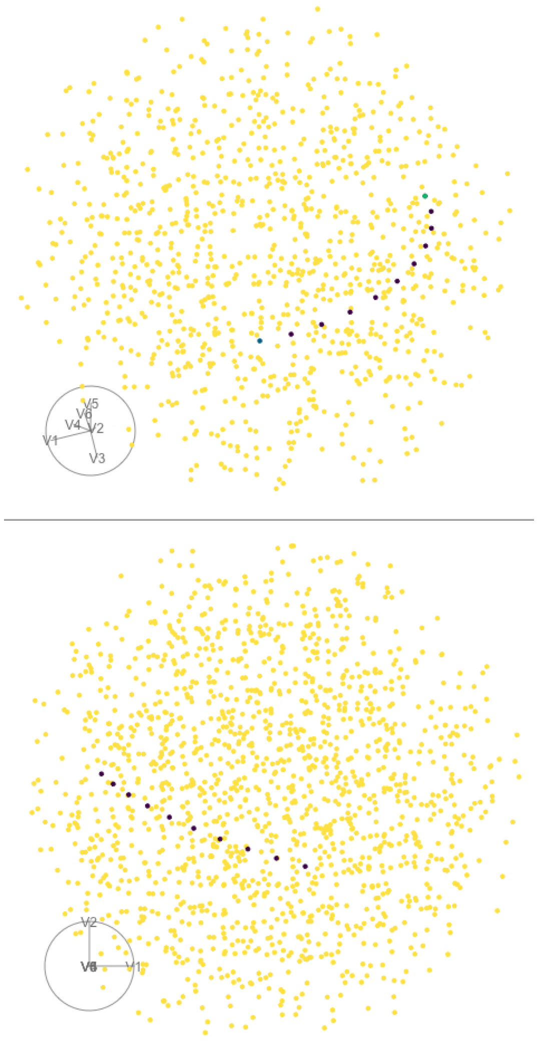


Figure 3: Interpolation steps of 1D projections of 6D data

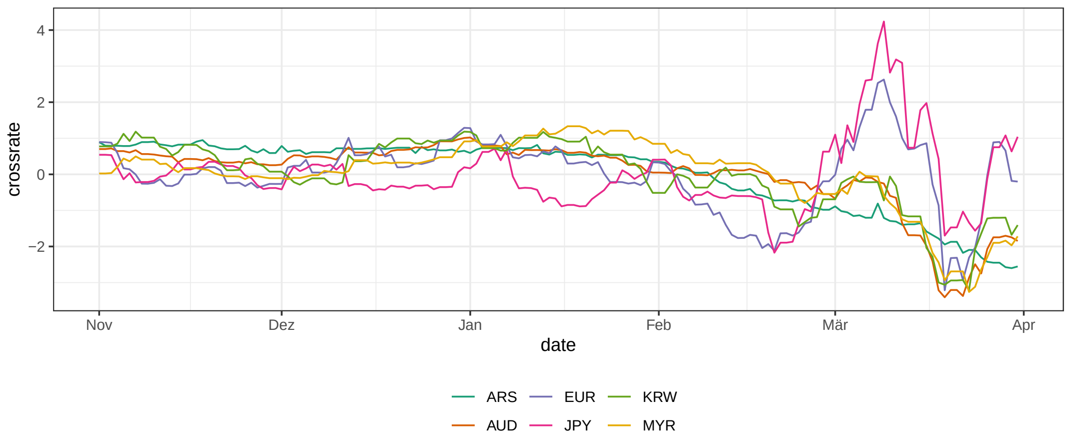


Figure 4: All the currencies are standardised and the sign is flipped. The high value means the currency strengthened against the USD, and low means that it weakened.

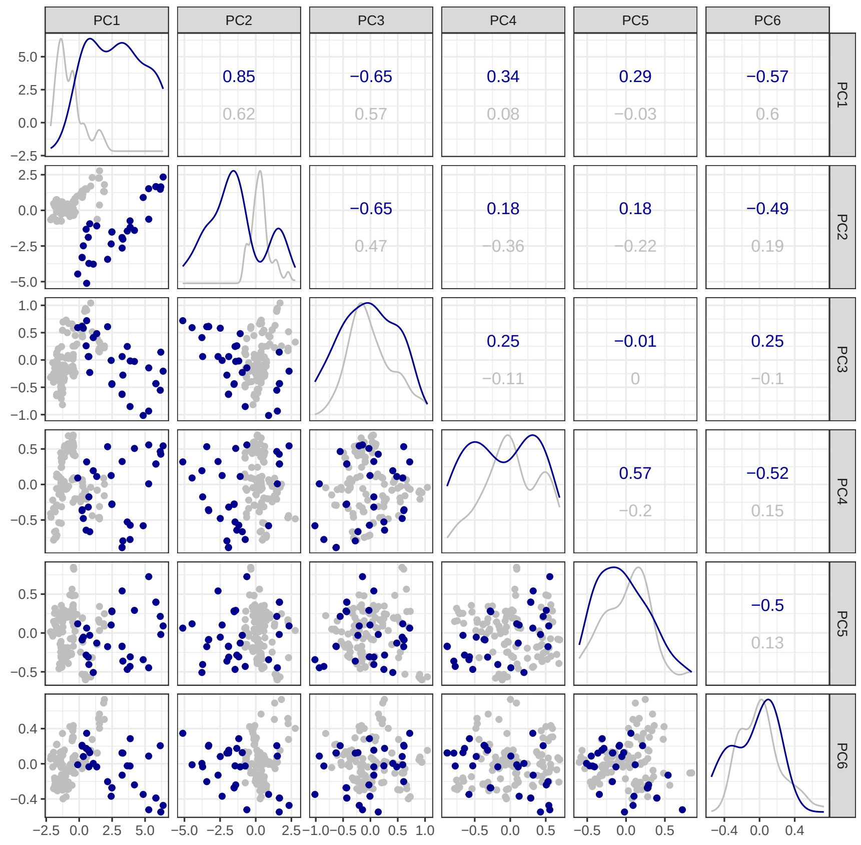


Figure 5: There is a strong non-linear dependence between PC1 and PC2. Observations in March 2020 are highlighted in dark blue, all other months are shown in grey.

PCA result

Since the collective behavior observed in March 2020 clearly stands out in the time series display, we may expect that we can capture the dependence between the two groups using principal components analysis. Figure @re(pca-result-static) shows a scatter plot matrix of the PCs of our dataset. Indeed we find strong non-linear association between the first two PCs. Investigation of the rotation shows that the first principal component is primarily a balanced combination between ARS, AUD, KRW and MYR (and a smaller contribution of EUR), and the second contribution is dominated by EUR and JPY contrasted with smaller contributions from ARS and MYR. Our next step is thus to use projection pursuit to identify the best projection matrix that captures the non-linear functional dependence.

New splines2d index

We now use the splines index to identify a projection with functional dependence between the first and second direction of the projection. Note here that because of strong linear correlations between the currencies, we start from the first four principal components, explaining over 97% of the variance in the data. To avoid starting from the view already identified in the first two principal components we start from a projections onto the third and forth principal components. The PCA display in `tourr` is used to show each projection axes display in terms of the original variables, while only touring within the smaller space spanned by the first four principal components.

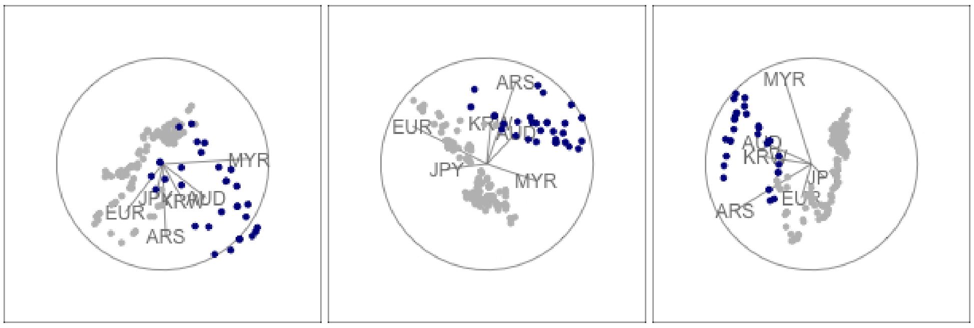
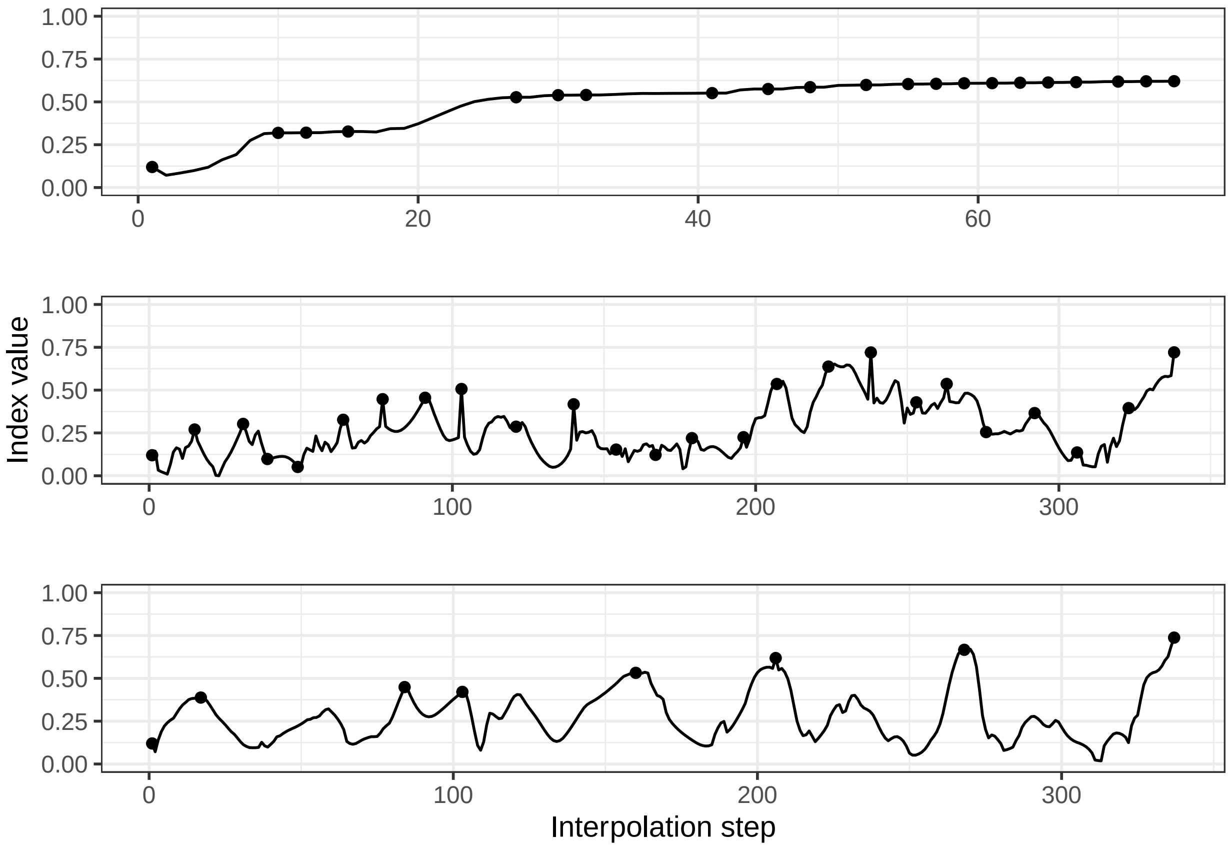


Figure 6: Final view after optimisation in the guided tour using geodesic optimisation (left), simulated annealing with geodesic interpolation (middle) and simulated annealing with givens interpolation (right).



What do we learn from looking at the traces?

- geodesic search does not have problem from rotation dependent index because it is searching along geodesic path, so everything is consistent, but it can get stuck and cannot jump to better solutions
- instead we might want to use random search (simulated annealing), but here we have huge problems from the rotation dependence of the index - even though this search should only accept target planes with higher index values we get big drops just because of the change in index value after rotation, the search path is very long and cannot find a good optimum
- using givens interpolation with the random search can fix these issues, we can step across valleys with lower index values but each target plane is increasing the index value, we do not need to restart the search - even with longer interpolations from the within-plane rotation this search is much more efficient (fewer target planes in between, similar path length due to longer interpolation), the final index value in our run is above what the random search with geodesic interpolation has found, the final view is certainly more interesting

Conclusion

The R package **woylier** provides implementation of Givens interpolation path algorithm that can be used as an alternative interpolation method for tour. The algorithm implemented in the **woylier** package comes from Buja et al. (2005). We illustrate the use of the functions provided in the package for R users.

The motivation to develop this package comes from rotational invariance problem of current geodesic interpolation algorithm implemeneted in **tourr** package. The package gives users the ability to detect non-linear association between variables more precisely.

It is important to mention that **woylier** package should be integrated with **tourr** package. The future improvements that needs to be done in the package is to generalize the interpolation for more than 2d projections of data.

Bibliography

D. Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM J Sci Stat Comput 6(1)*, page 128–143, 1985. [p1]

A. Buja, D. Cook, D. Asimov, and C. B. Hurley. Theory of dynamic projections in high-dimensional data visualization. 2004. [p1]

A. Buja, D. Cook, D. Asimov, and C. Hurley. Computational methods for high-dimensional rotations in data visualization. *Handbook of Statistics*, page 391–413, 2005. doi: 10.1016/s0169-7161(04)24014-7. [p1, 2, 11]

D. Cook and A. Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997. ISSN 1061-8600. doi: 10.2307/1390747. URL `http://www.jstor.org/stable/1390747`. [p1]

D. Cook, A. Buja, J. Cabrera, and C. Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995. ISSN 10618600. URL `http://www.jstor.org/stable/1390844`. [p1]

G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 1989. ISBN 0801837723. [p3]

K. Grimm. *Kennzahlenbasierte Grafikauswahl*. doctoral thesis, Universität Augsburg, 2016. [p1]

J. 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. *Statistical computation; New York, Academic Press*, pages 427–440, 1969. [p1]

U. Laa and D. Cook. Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics. *Comput Stat 35*, page 1171–1205, 2020. doi: https://doi.org/10.1007/s00180-020-00954-8. [p1]

U. Laa, A. Aumann, D. Cook, and G. Valencia. New and simplified manual controls for projection and slice tours, with application to exploring classification boundaries in high dimensions. *Journal of Computational and Graphical Statistics*, 0(0):1–8, 2023. doi: 10.1080/10618600.2023.2206459. URL `https://doi.org/10.1080/10618600.2023.2206459`. [p1]

S. Lee, D. Cook, N. da Silva, U. Laa, N. Spyrison, E. Wang, and H. S. Zhang. The state-of-the-art on tours for dynamic visualization of high-dimensional data. *WIREs Computational Statistics*, 14(4): e1573, 2022. doi: https://doi.org/10.1002/wics.1573. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1573`. [p1]

B. Schloerke. *geozoo: Zoo of Geometric Objects*, 2016. URL `https://CRAN.R-project.org/package=geozoo`. R package version 0.5.1. [p7]

H. Wickham, D. Cook, H. Hofmann, and A. Buja. tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18, 2011. URL `https://doi.org/10.18637/jss.v040.i02`. [p1, 5]

Zoljargal Batsaikhan
Monash University
Department of Econometrics and Business Statistics
Clayton, VIC, Australia
`https://github.com/zolabat`
ORCiD: 0009-0005-0055-1448
`zoljargal11@gmail.com`

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
Clayton, VIC, Australia
`http://www.dicook.org`
ORCiD: 0000-0002-3813-7155
`dicook@monash.edu`

Ursula Laa
BOKU University
Institute of Statistics
Vienna, Austria
`https://uschilaa.github.io`
ORCiD: 0000-0002-0249-6439
`ursula.laa@boku.ac.at`