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 alternative method for interpolation path between tour frames using Givens rotation.

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 screens. This issue was tackled by 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. Grand tour allows users to see dynamic 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 lots of work has been done about the interactivity of the tour giving control to users (Buja et al. 2005). The alternatives to the grand tour include manual tour, little tour, guided tour, local tour, and planned tour. These are different ways of selecting a basis for projection.

The guided tour combines projection pursuit with the tour (Cook et al. 1995) and it is implemented in the **tourr** package. The idea of projection pursuit is a procedure used to locate the projection of high-to-low dimensional space that exposes the most interesting feature of data originally proposed by Kruskal (1969). It involves defining a criterion of interest, a numerical objective function, that indicates the interestingness of each projection and 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 projection and we use interpolation method to produce the path. The current implementation of guided tour in **tourr** package uses geodesic interpolation between planes. Geodesic interpolation path is the locally shortest path between planes with no within-plane spin. As a result of this method, the rendered target plane could be the rotated version of the target plane we wanted. This is not a problem when the structure we are looking can be identified without turning the axis around. The detailed discussion about the role of orientation in data projection can be found in Buja et al. (2004).

However, in some cases of non-linear projection pursuit index, the orientation of frames does matter. One example is the splines2D index. The value of the splines2D index (Grimm 2016) changes depending on the rotation (Laa and Cook 2020). The lack of rotation invariance of the splines2d index raises complications in the optimization process. This rotational dependence issue of non-linear projection pursuit functions is the motivation of this work. Figure 1 illustrates the rotational invariance problem for a modified spline2D index. The original implementation for the splines2d index makes a couple of additional calculations to reduce (but not remove) the rotational invariance. 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. The goal with the frame to frame interpolation is that optimization 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 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 provided the theoretical framework of the Givens interpolation method followed by a section about the implementation Givens path in R. Furthermore, we will apply this interpolation method to the projection pursuit of splines index to search for nonlinear associations between variables in the example data set. Finally, this article includes a discussion about the further steps.

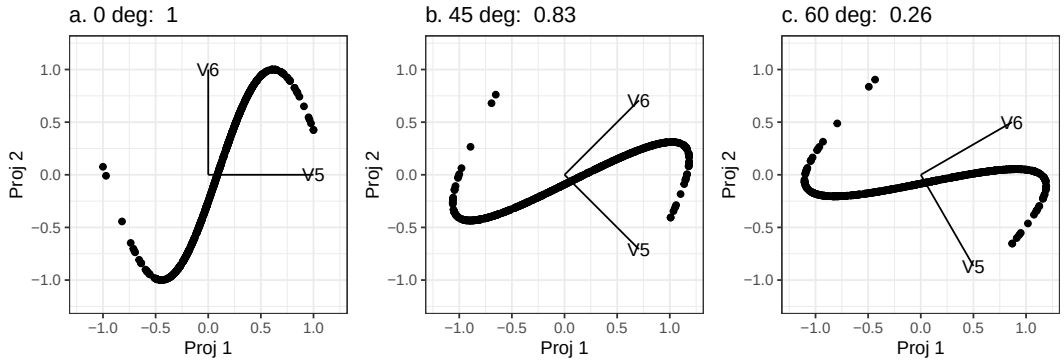


Figure 1: Modified spline index computed on within-plane rotations of the same projection has very different values: (a) original pair has maximum index value of 1.00, (b) axes rotated 45^o drops index value to 0.83, (c) axes rotated 60^o drops index to a very low 0.26. This shows an index that is rotationally variable.

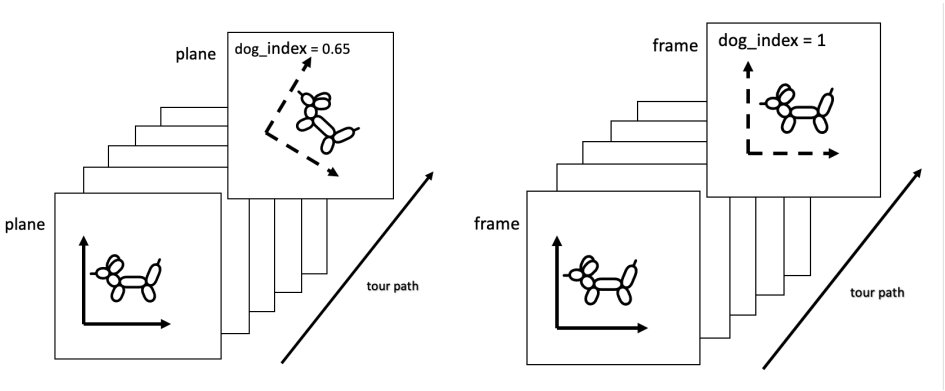


Figure 2: Plane to plane interpolation (left) and Frame to frame interpolation (right). We used dog index for illustration purposes. For some non-linear index orientation of data could affect the index.

Background

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

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

The interpolating paths of plane versus frames

Current implementation of **tourr** package is locally shortest (geodesic) interpolation of planes. The pitfall of this interpolation method is that it does not account for rotation variability. Therefore, the interpolation of frames is required when the orientation of projection 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.

The orientation of frames could be important when non-linear projection pursuit function is used in guided tour. An illustration of such cases are shown in Figure 2.

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 $p \times d$ matrix with pairwise orthogonal columns of unit length that satisfies, where I_d is the identity matrix in d dimensions.

$$F^T F = I_d$$

- Paths of projections are given by continuous one-parameter families $F(t)$ where $t \in [a, z]$ interval representing time. We denote the starting frame by $F_a = F(a)$ and target frame by $F_z = F(z)$. Usually, F_z is selected target basis that has chosen via various methods. While grand tour chooses target frames randomly, guided tour chooses the target plane by optimizing the projection pursuit index. Interpolation methods are used to move from F_a to F_z.
- B is preprojection basis of F_a and F_z.

Preprojection algorithm

In order to make the interpolation algorithm simple, we need to carry out "preprojection" step. The purpose of preprojection is to limit data subspace that the interpolation path, $F(t)$, is traversing. In other words, preprojection step make sure the interpolation path between two frames F_a and F_z is not going to the data space that is not related to F_a and F_z. Simply, 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 regard to F_a. We denote this orthonormal basis by $F_\star$. Then build 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:

$$F_a = B^T W_a, F_z = B^T W_z$$

The interpolation problem is then reduced to the construction of paths of frames $W(t)$ that interpolate the preprojected frames W_a and W_z. Because B is orthonormalized basis of F_z with regard to F_a, W_a is $2d \times d$ matrix of 1, 0s. This is an important character for our interpolation algorithm of choice, Givens interpolation.

Givens interpolation path algorithm

A rotation matrix is a transformation matrix used to perform a rotation in Euclidean space in a plane. A rotation matrix that transforms 2-D 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 selected 2 variables, it is called Givens rotation. Let's denote those 2 variables i and j. The Givens rotation is useful for introducing zeros on a grand scale and used for computing the QR decomposition of matrix in linear algebra problems. One advantage over other transformation methods which is particularly useful in our case is the ability to zero elements more selectively.

The interpolation methods 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 in the (i, j)-plane for any $j \neq i$. This rotation affects only coordinate i and j and leave all other coordinates unchanged.

Sequences of Givens rotations can map any orthonormal d-frame F in p-space to 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 2D path construction process of original 6D data frame.

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 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 second coordinate of a plane is calculated.

When $d = 2$, there are 5 rotations involved with 5 different angles that makes each elements 0. For example, the first rotation angle θ_1 is an angle in radian between $(1, 0)$ and (a_{11}, a_{21}). This rotation matrix would make element a_{21} zero:

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

6th rotation is not necessary due to orthonormality of columns. If we make one element of a column 1 that means all other elements must be 0.

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 step should include the time parameter, t, so that it shows the interpolation process rendered in the movie-like sequence. We break θ_i into the number of steps, $n - step$, that we want to go from starting frame to the target frame, which means it moves by equal angle in each step.
6. Finally, we reconstruct our original frames using B. This reconstruction is done at each step of interpolation so that we have interpolated path as result. We use F_t to project the orignal data into lower dimensions.

$$F_t = BW_t$$

Projection pursuit index functions

The properties of several projection pursuit index functions were investigated.(Laa and Cook 2020) The smoothness, squintability, flexibility, rotation invariance, and speed of projection pursuit index functions were examined. The one property that is interesting to us is rotation invariance. The rotational invariance is examined by computing projection pursuit index for different rotations within 2D plane. It is established that the dcor2d, splines2d and TIC index are not rotationally invariant. Splines2D index measures nonlinear association between variable by fitting spline model. It compares the variance of residuals and the functional dependence is stronger if the index value is larger.

Implementation

We implemented each steps mentioned in **Givens interpolation path algorithm** in separate functions and combined them in *givens_full_path()* function. Here is the input and output of each functions and it's descriptions.

- `givens_full_path(Fa, Fz, nsteps)`: Construct full interpolated frames.
  - `input`: Starting and target frame (Fa, Fz) and number of steps
  - `output`: 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.
  - `input`: Starting and target frame (Fa, Fz)
  - `output`: B pre-projection px2d matrix
- `construct_preframe(Fa, B)`: Construct preprojected frames.
  - `input`: A frame and the pre-projection px2d matrix
  - `output`: Preprojected frame in preprojection space
- `row_rot(a, i, k, theta)`: Performs Givens rotation (Golub and Loan 1989).
  - `input`: a-matrix, i-row, k-row that we want to zero the element, theta-angle between i, k rows
  - `output`: theta angle rotated matrix a
- `calculate_angles(Wa, Wz)`: Calculate angles of required rotations to map Wz to Wa.
  - `input`: Preprojected frames (Wa, Wz)
  - `output`: Names list of angles
- `givens_rotation(Wa, angles, stepfraction)`: It implements series of Givens rotations that maps Wa to Wz
  - `input`: Wa starting preprojected frame, list of angles of required rotations to map Wz to Wa, stepfraction.
  - `output`: Givens path
- `construct_moving_frame(Wt, B)`: Reconstruct interpolated frames using pre-projection.
  - `input`: Pre-projection matrix B, Each frame of givens path
  - `output`: A frame of on a step of interpolation

The interface of tour is that it renders one projection of data at a time. It displays one projection and asks for the next projection. Therefore, path of projections shown below is sequence of projections to be renders at tour display.

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,]  0.02498501 -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
#>
#> , , 2
#>
#>            [,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
#>
#> , , 3
#>
#>             [,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
#>
#> , , 4
#>
#>             [,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
#>
#> , , 5
#>
#>             [,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
```

Examples of interpolated paths

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

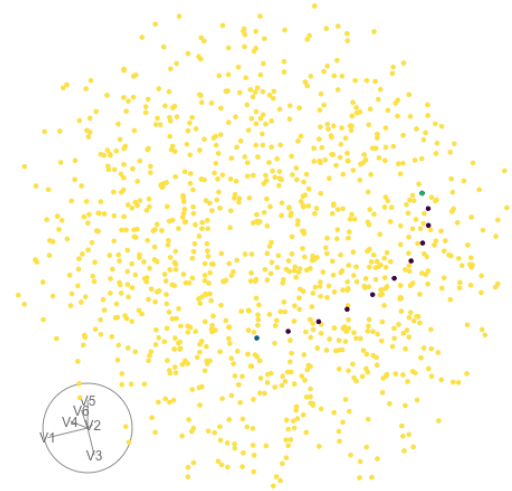


Figure 3: Interpolation steps of 1D projections of 6D data

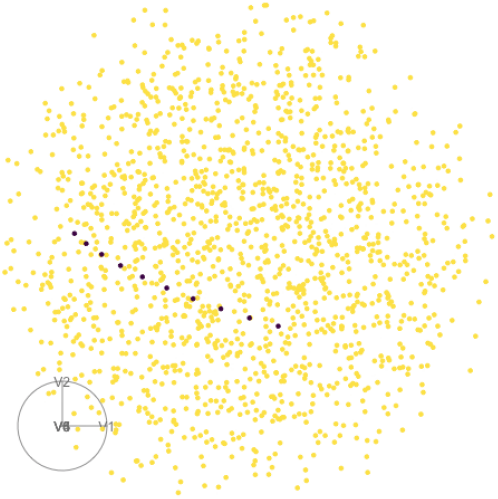


Figure 4: Interpolation steps of 2D projections of 6D data

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 4 shows the Givens interpolation steps in 2D projection of 6D data.

Comparison of geodesic interpolation and Givens 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 5 shows the Geodesic and Givens interpolation to target frame where the two variables forms sine curve.

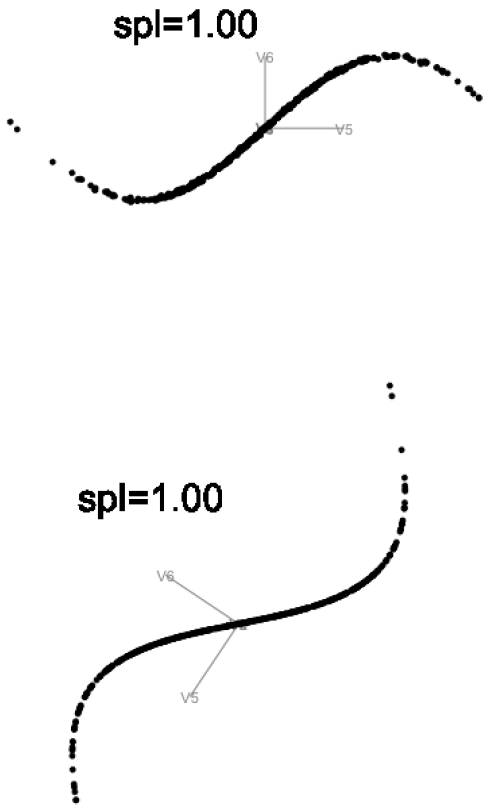


Figure 5: Givens interpolation path (left) and Geodesic interpolation path (right) to target frame. Givens interpolation goes to exact frame that has the correct orientation while Geodesic interpolation goes to rotation of the target plane.

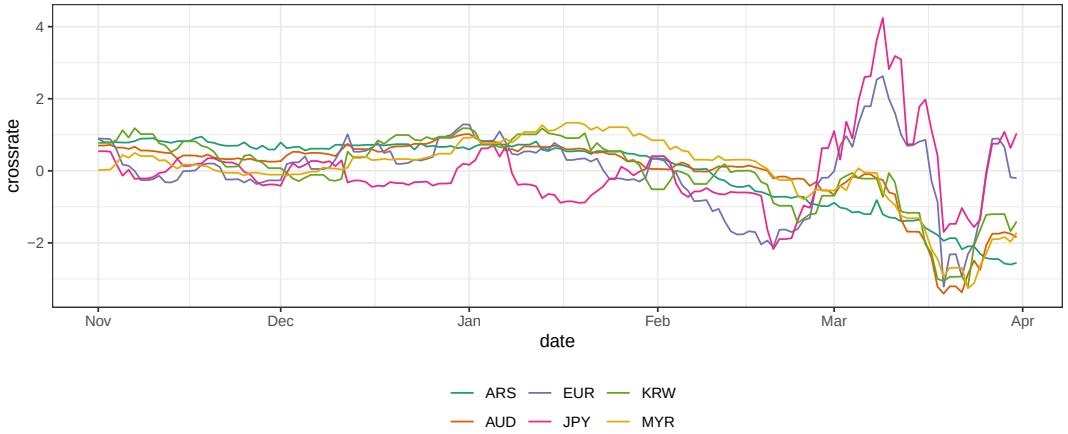


Figure 6: All the currencies are standardised and the sign is flipped. The high value means the currency strengthened against the USD, and low means that it weakened.

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 6 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.

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 99% 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` to show the projection in terms of the original variables while only touring within the smaller space spanned by the first four principal components.

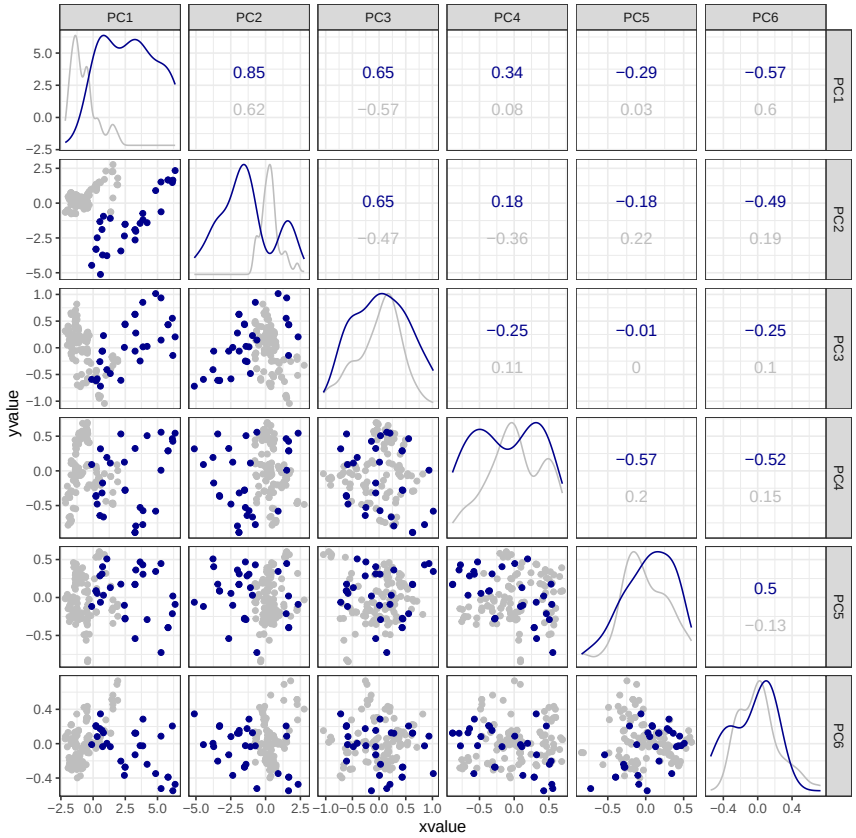
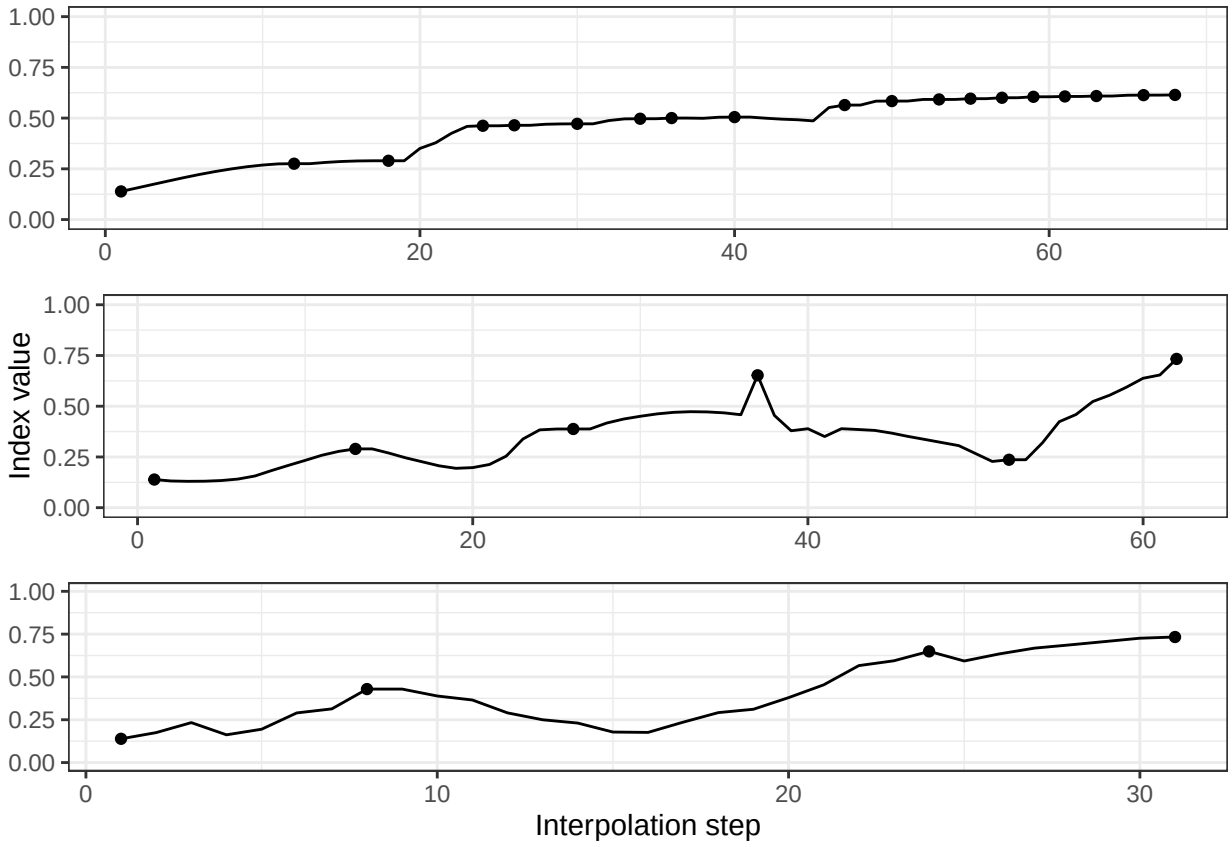


Figure 7: There is a strong non-linear dependence between PC1 and PC2. Observations in March 2020 are highlighted in dark blue, all other months are shown in grey.



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 (shorter path), 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.

References

Asimov, D. 1985. "The Grand Tour: A Tool for Viewing Multidimensional Data." *SIAM J Sci Stat Comput 6(1)*, 128–43.

Buja, Andreas, Dianne Cook, Daniel Asimov, and Catherine Hurley. 2005. "Computational Methods for High-Dimensional Rotations in Data Visualization." *Handbook of Statistics*, 391–413. https://doi.org/10.1016/s0169-7161(04)24014-7.

Buja, Andreas, Dianne Cook, Daniel Asimov, and Catherine B. Hurley. 2004. "Theory of Dynamic Projections in High-Dimensional Data Visualization." In.

Cook, Dianne, Andreas Buja, Javier Cabrera, and Catherine Hurley. 1995. "Grand Tour and Projection Pursuit." *Journal of Computational and Graphical Statistics* 4 (3): 155–72. http://www.jstor.org/stable/1390844.

Golub, Gene H., and Charles F Van Loan. 1989. *Matrix Computations*. Johns Hopkins University Press.

Grimm, Katrin. 2016. "Kennzahlenbasierte Grafikauswahl." Doctoral thesis, Universität Augsburg.

Kruskal, JB. 1969. "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*, 427–40.

Laa, Ursula, and Dianne Cook. 2020. "Using Tours to Visually Investigate Properties of New Projection Pursuit Indexes with Application to Problems in Physics." *Comput Stat 35*, 1171–1205. https://doi.org/https://doi.org/10.1007/s00180-020-00954-8.

Schloerke, Barret. 2016. *Geozoo: Zoo of Geometric Objects*. https://CRAN.R-project.org/package=geozoo.

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