SHORT TECHNICAL NOTE

Appendix: New and simplified manual controls for projection and slice tours

Ursula Laa[a], Alex Aumann[b], Dianne Cook[c], German Valencia[b]

[a]Institute of Statistics, University of Natural Resources and Life Sciences, Vienna; [b]School of Physics and Astronomy, Monash University; [c]Department of Econometrics and Business Statistics, Monash University

ARTICLE HISTORY
Compiled September 12, 2022

KEYWORDS
data visualisation; grand tour; statistical computing; statistical graphics; multivariate data; dynamic graphics

1. Refinements to enforce exact position

The problem with new simple method (Algorithm 1) is that the precise values for V_m cannot be specified because the orthonormalisation will change them.

1.1. *Adjustment method 1*

A small modification to algorithm 1 will maintain the components of V_m precisely (Figure 1. It is as follows:

1. Provide A, and m.
2. Change values in row m, giving A^*.
3. Store row m separately, and zero the values of row m in A^*, giving A^{*0}.
4. Orthonormalise A^{*0}, using Gram-Schmidt.
5. Replace row m with the original values, giving A^{**}.
6. For $d = 2$, adjust the values of $a^{**}_{.2}$ using

$$a^{**}_{j2} + \frac{a_{m1}a_{m2}}{p-1}, j = 1, ..., p, j \neq m$$

.

which ensures that

CONTACT Ursula Laa. Email: ursula.laa@boku.ac.at, Alex Aumann. Email: aaum0002@student.monash.edu, Dianne Cook. Email: dicook@monash.edu, German Valencia. Email: german.valencia@monash.edu

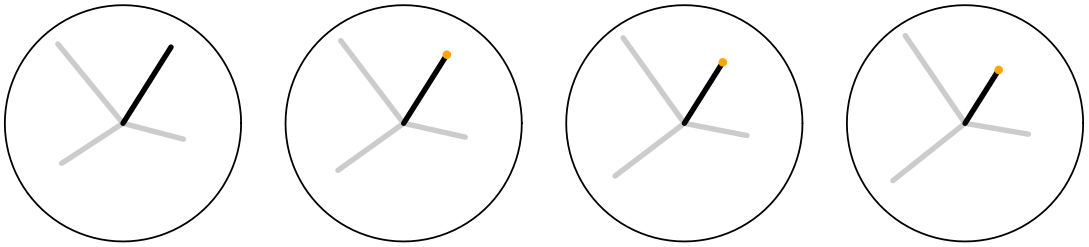

Figure 1. Manual controls with algorithm 2. The precise location of the axis is maintained.

$$\sum_{j=1,j\neq m}^{p} a_{j1}^{**} a_{j2}^{**} + a_{m1} a_{m2} = 0$$

.

If $d > 2$ the process would be sequentially repeated in the same manner that Gram-Schmidt is applied sequentially to orthormalise the columns of a matrix. If $d = 1$ no orthonormalisation is needed, and the projection vector would simply need to be normalized after each adjustment.

1.2. *Adjustment method 2*

For $d = 2$ projections, the projection matrix is the sub-matrix of A formed by its first two columns. Whereas orthonormality of the basis for the p-dimensional space is given by $e_i \cdot e_j = \delta_{ij}, i, j, = 1, \cdots ,$, orthonormality of the projection matrix is expressed as $P_i \cdot P_j = \delta_{ij}, i, j = 1, 2$. Movement of the cursor takes the two components x_{m1}, x_{m2} into a selected new value a, b. Although the motion is constrained by $a^2 + b^2 \leq 1$, this is not sufficient to guarantee orthonormality of the new projection matrix. One possible algorithm to achieve this is

(1) Cursor movement takes $x_{m1}, x_{m2} \to a, b$
(2) The freedom to change the components $A_{i3} \cdots A_{ip}$ (the columns of A not corresponding to the projection matrix) is used to select a new orthonormal basis as follows:
   (a) For row m one chooses $A_{m3} = \sqrt{1 - a^2 - b^2},\ A_{m,k>3} = 0$
   (b) For other rows, $A_{i\neq m,j\geq 3}$ random selections in the range $(-1, 1)$ are made.
   (c) The Gram-Schmidt algorithm is then used to obtain an orthonormal basis taking e_m as the first vector (which is already normalized), and then proceeding as usual $e_1 \to e_1 - (e_1 \cdot e_m)e_m$, $e_1 \to e_1/(e_1 \cdot e_1)$, etc.
(3) this results in the orthonormal basis A^* and a new projection matrix with $P_{m1} = a,\ P_{m2} = b$.

The random completion of A outside the projection matrix provides an exploration of dimensions orthogonal to the projection plane. However, it renders projections in a way that is not continuouns and may be distracting. This can be alleviated by replacing the random completion with a rule restricting the size of the jumps.

$$A = \begin{array}{c} \\ e_1 \to \\ e_2 \to \\ \vdots \\ e_m \to \\ \vdots \\ e_p \to \end{array} \begin{array}{c} \begin{array}{cc} \overset{P_1}{\downarrow} & \overset{P_2}{\downarrow} \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdot & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdot & \cdots & x_{mp} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdot & \cdots & x_{pp} \end{pmatrix} \end{array}, \quad (x_{m1}, x_{m2}) \to (a, b) \tag{1}$$

1.3. *Adjustment method 3*

A very simple adjustment method works purely within the (hyper)plane, and is using the fact that the first direction will not be rotated when applying Gram-Schmidt. The following steps are needed:

1. Provide A, and m.
2. Change values in row m, giving A^*.
3. Rotate the basis within the projection (hyper-)plane, such that the first direction is aligned with the direction of m in A^*, giving $\hat{A}^*$. In a 2D projection this means finding the angle θ by which the basis should be rotated.
4. Normalise the first direction in $\hat{A}^*$ while keeping the contribution in row m fixed, giving $\hat{A}^{*0}$.
5. Orthonormalise $\hat{A}^{*0}$ using Gram-Schmidt, giving $\hat{A}^{**}$. By construction the contribution of m will remain fixed.
6. Rotate $\hat{A}^{**}$ back into the original basis to obtain A^{**}.

Diagram needed?
{Do we need three methods? is this one implemented somewhere?}

2. Software details

- **Why is this a good sandbox**
- **Explain the functionality available in the notebooks**

Here we describe the functions implemented in the Mathematica package `mmtour.wl`. The main function is

- `SliceDynamic[data, projmat, height, heightRange, legendQ(=1), flagQ(=1), colorFunc(=ColorData[97])]` with arguments
- `data`: A data matrix with an extra grouping that has a label (string) and a flag (numerical)
- `projmat`: The initial projection matrix. It is possible to start with a random projection with the input "random" (including the quotes) in this entry field.
- `height`: The initial height of the slice.
- `heightRange`: The range of slice heights to be explored.
- `legendQ`: A flag signaling whether the data matrix includes a column with group labels (1) or not (0).

- `flagQ`: A flag signaling whether the data matrix includes a numerical flag for the groups.
- `colorFunc`: specifies the colour mapping for the different groups.

The function renders a manual tour with controls that allow the user to navigate through different projections. For a given projection, a slider permits the user to change the slide width within its range and a separate box switches the view to a projection. The user can also change the centre point of slices; the size of the points rendered and the scale of the plot through sliders. An additional box displays the coordinates of the projection matrix explicitly.

This function can be used for ungrouped data setting legendQ and flagQ to 0.

To remove a group from the visual display, the user can choose specific colours for the entry colorFunc. For example, {Red,White,Green} would display groups 1 (red) and 3 (green) while making 2 invisible.

The Slice Tour* notebooks include the additional functions: ProjectionPlot, Projected2DSliderPlot, and VisualiseSliceDyanmic. ProjectionPlot is very similar to SliceDynamic, except it only displays projections and VisualiseSliceDynamic allows you to discern which points exist inside the slice and outside it. However, ProjectedLocatorPlot displays the interactive slightly differently. Several locator panes are displayed above the plot and each one corresponds to a row in the projection matrix but behaves the same as ProjectionPlot.

- `ProjectionPlot[data, projmat, legendNames(={}), colorFunc(=ColorData[97])]` with arguments
- `data`: A data matrix with one or more groups labeled by a flag (numerical)
- `projmat`: The initial projection matrix. It is possible to start with a random projection with the input "random" (including the quotes) in this entry field.
- `legendNames`: an optional list of labels for the groups.
- `colorFunc`: specifies the colour mapping for the different groups.

This function is similar to SliceDynamic but can only display projections. Using it instead of SliceDynamic simplifies the display and is more efficient. The data file should not contain labels for groups, if these are desired they are passed through the optional argument legendNames as {"group 1", "group 2",...}, for example.

The display allows the user to navigate through different projections, choose the point size and zoom the factor, and explicitly display the coordinates of the projection.

- `ProjectedLocatorPlot[data, projmat, legendNames(={}), colorFunc(=ColorData[97])]` with arguments
- `data`: A data matrix with one or more groups labeled by a flag (numerical)
- `projmat`: The initial projection matrix. It is possible to start with a random projection with the input "random" (including the quotes) in this entry field.
- `legendNames`: an optional list of labels for the groups.
- `colorFunc`: specifies the colour mapping for the different groups.

This function is very similar to ProjectionPlot but uses a different display for navigation of projections in which each coordinate is represented on a separate dial.

- `VisualiseSliceDynamic[data, projmat, height, heightRange]` with arguments
- `data`: A data matrix containg only one set.In this case the data matrix should not contain column labels.

- `projmat`: The initial projection matrix. It is possible to start with a random projection with the input "random" (including the quotes) in this entry field.
- `height`: The initial height of the slice.
- `heightRange`: The range of slice heights to be explored.

This function differs from SliceDynamic in that it looks at only one data set but displays both the points inside and outside the slice. The display panel allows the user to specify the size of both sets of points (inside and outside) along with the slice height, centre point and zoom level.