SHORT TECHNICAL NOTE

A new simplified manual tour, with examples in mathematica and R

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

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

ARTICLE HISTORY
Compiled April 19, 2022

ABSTRACT
Something here

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

1. Introduction

From a statistical perspective it is rare to have data that are strictly 3D, and so unlike in most computer graphics applications, the more useful methods for data analysis show projections from an arbitrary dimensional space. These are dynamic data visualizations methods and are collected under the term *tours*. Tours involve views of high-dimensional (p) data in low-dimensional (d) projections. In his original paper on the grand tour, Asimov (1985) provided several algorithms for tour paths that could theoretically show the viewer the data *from all sides*. Prior to Asimov's work, there were numerous preparatory developments including **?**'s PRIM-9. PRIM-9 had user-controlled rotations on coordinate axes, allowing one to manually tour through low-dimensional projections. It is impractical to impossible to steer through all possible projections, unlike Asimov's tours which allows one to quickly see many, many different projections. After Asimov there have been many, many tour developments, as summarized in Lee et al. (2021).

One such direction of work develops the ideas from PRIM-9, to provide manual control of a tour. Cook and Buja (1997) describes controls for 1D (or 2D) projections, in a 2D (or 3D) manipulation space, allowing the user to select any variable axis, and rotate it into or out of or around the projection through horizontal, vertical, oblique, radial or angular changes in value. Spyrison and Cook (2020) refines this algorithm and implements them to generate animations.

Manual controls are especially useful for assessing sensitivity of structure to particular elements of the projection. There are many places where it is useful. In exploratory data analysis, where one sees clusters in a projection, can some variables be removed

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

from the projection without affecting the clustering. For interpreting models, one can reduce or increase a variable's contribution to examine the variable importance. These controls can also be used to interactively generate facetted plots (**?**), or spatiotemporal glyphmaps (**?**). Having the user interact with a projection is extremely valuable for understanding high-dimensional data. However, these algorithms have two problems: (1) the pre-processing of creating a manipulation space overly complicates the algorithm, (2) extending to higher dimensional control is difficult.

Through experiments with the relatively new interactive graphics capabilities in mathematica(?), we have realized that there is a simpler approach, which is more direct, and extensible for generating user interaction. This paper explains this, and is organized as follows. The next section describes the new algorithm for manual control. This is followed by details on implementation. The applications section illustrate how these can be used.

2. Manual tour

An orthonormal basis ($A_{p\times d}$) and a variable id ($m \in \{1, ..., p\}$) to control are provided to initialise a manual tour. A method to update the values of the component of the controlled variable V_m is needed.

2.1. *Background*

In the original work, the method for updating component values, for a 2D projection, was built trackball controls in 3D. A 3D manipulation space is created, as illustrated in Figure 1, where the controlled variable has full range of motion from -1 to 1. Movements of a cursor are recorded and converted into changes in the values of V_m to change it's values in the displayed 2D projection. Movement could also be constrained to be only in horizontal, vertical, radial or angular motions.

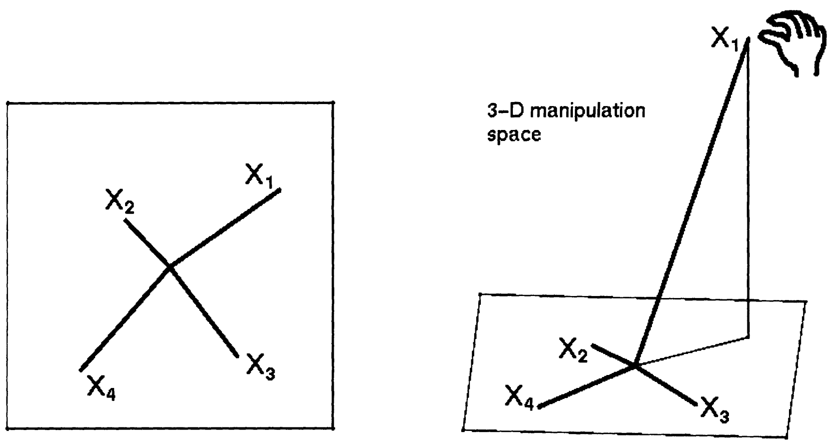


Figure 1. Original contruction of the manual tour designed for 2D projections and created a 3D space from which to utilise track ball controls to change it's contribution. (Figure 3 from Cook and Buja (1997).)

2.2. ***New simpler definition***

The new approach emerged from experiments in mathematica. The components corresponding to V_m are directly controlled by cursor movement, which updates row m of A. The updated matrix is then orthonormalised.

2.2.1. Algorithm 1

1. Provide A, and m. (Note that m could also be automatically chosen as the component that is closest to the cursor position.)
2. Change values in row m, giving A^*. A large change in these values would correspond to making a large jump from the current projection. Small changes would correspond to tracking a cursor, making small jumps from the current projection.
3. Orthonormalise A^*, using Gram-Schmidt. For $d = 2$, and $A^* = [a_{.1}\ a_{.2}]$, the steps are:
   i. Normalise $a_{.1}$, and $a_{.2}$.
   ii. $a^*_{.2} = a_{.2} - a^T_{.1} a_{.2} a_{.1}$.
   iii. Normalise $a^*_{.2}$.

This algorithm will produce the changes to a projection as illustrated in Figure **??** (top row). The controlled variable, V_m, corresponds to the black line, and sequential changes to row m of A can be seen to roughly follow a specified position (orange dot). Changes in the other components happen as a result of the orthonormalisation, but are uncontrolled.

2.2.2. Algorithm 2

The problem with Algorithm 1 is that the precise values for V_m cannot be specified because the orthonormalisation wil change them. This modification will maintain the components of V_m precisely (Figure **??** (bottom row)). The algorithm 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

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

.

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 normalised after each adjustment.

2.2.3. *Algorithm 3*

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 $x_{m1}, x_{m2} \rightarrow a, b$
(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 normalised), and then proceeding as usual $e_1 \rightarrow e_1 - (e_1 \cdot e_m)e_m$, $e_1 \rightarrow e_1/(e_1 \cdot e_1)$, etc.
(3) this results in the orthonormal basis A^* and a new projection matrix with $P_{m1} = a,\ P_{m2} = b$.

The random choice for the components of A not in the projection matrix allows the exploration of dimensions perpendicular to the projection plane.

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

2.2.4. *Potential Algorithm 3*

For now just sketching the idea:

- first step is to click on the axis display to change the contribution of one variable m
- capture that position and replace the corresponding row in the projection matrix
- for 2D projection this gives components m_1 along "x" direction and m_2 along y direction
- next step: rotate basis such that direction of row m now corresponds to the first basis vector, this means we apply 2x2 rotation matrix to each row of the projection matrix, where the rotation angle is $tan(\theta) = m_2/m_1$ (that matrix can

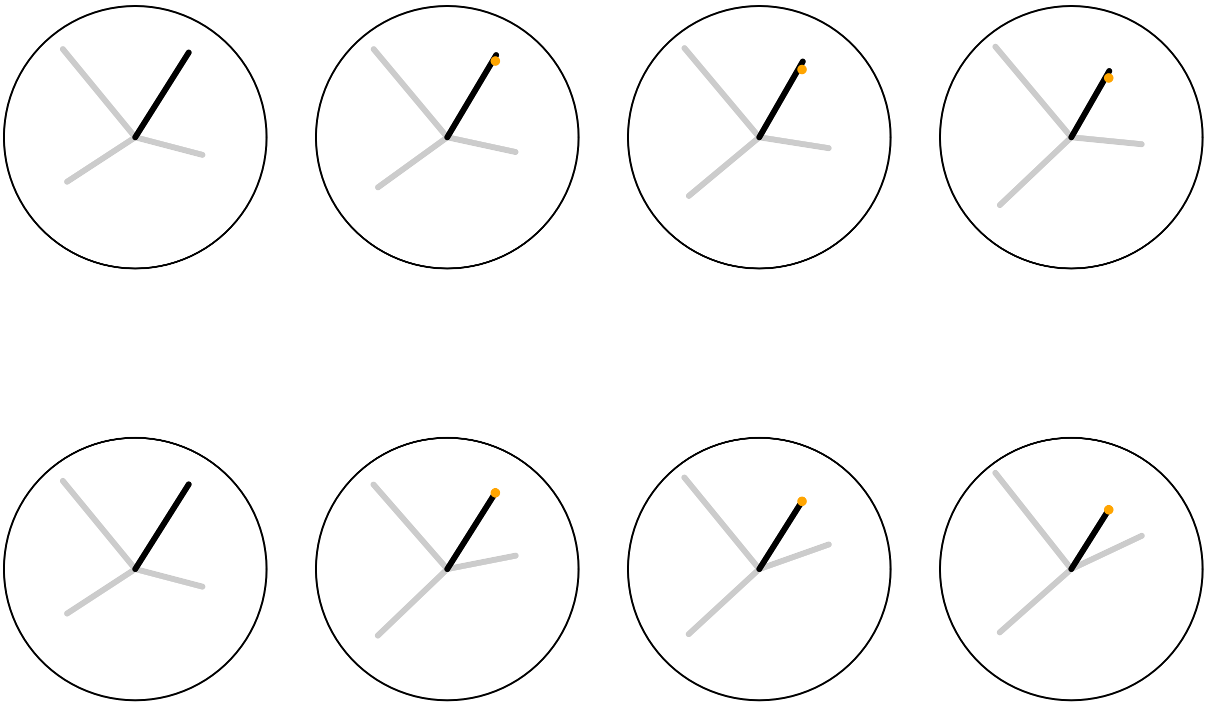
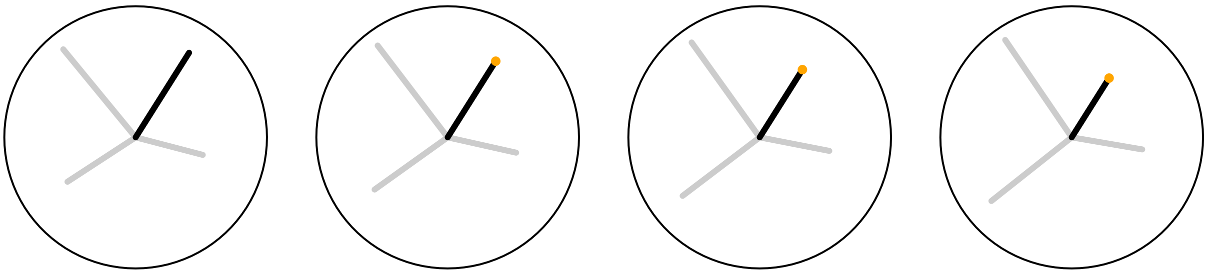


Figure 2. Sequence of projections where contribution of one variable is controlled (black) is changed: (top) unconstrained orthonormalisation, (bottom) constrained as specified. The dot (orange) indicates the chosen values for the controlled variable. For the constrained orthonormalisation it can be seen to precisely match the axis, but not so for the unconstrained orthonormalisation.

be written just in terms of that ratio, $x = m_2/m_1$, when $m_1 < 0$ I think we just need to translate $\theta \to \theta + \pi$)
- take the rotated basis, apply Gram-Schmidt, now the direction of m will not change (but the length will change during normalization)
- rotated back to the original xy basis to update the plots

XXX need to check calculation of angle when $m_1 < 0$



3. Implementation

4. Applications

5. Discussion

Acknowledgements

The authors gratefully acknowledge the support of the Australian Research Council. The paper was written in `rmarkdown` (Xie, Allaire, and Grolemund 2018) using `knitr` (Xie 2015).

Supplementary material

The source material and animated gifs for this paper are available at

References

Asimov, D. 1985. "The Grand Tour: A Tool for Viewing Multidimensional Data." *SIAM Journal of Scientific and Statistical Computing* 6 (1): 128–143.

Cook, Dianne, and Andreas Buja. 1997. "Manual Controls for High-Dimensional Data Projections." *Journal of Computational and Graphical Statistics* 6 (4): 464–480. `http://www.jstor.org/stable/1390747`.

Lee, Stuart, Dianne Cook, Natalia da Silva, Ursula Laa, Earo Wang, Nick Spyrison, and H. Sherry Zhang. 2021. "Advanced Review: The State-of-the-Art on Tours for Dynamic Visualization of High-dimensional Data." *arXiv:2104.08016 [cs, stat]* `http://arxiv.org/abs/2104.08016`.

Spyrison, Nicholas, and Dianne Cook. 2020. "spinifex: an R Package for Creating a Manual Tour of Low-dimensional Projections of Multivariate Data." *The R Journal* 12 (1): 243. `https://journal.r-project.org/archive/2020/RJ-2020-027/index.html`.

Xie, Yihui. 2015. *Dynamic Documents with R and knitr*. 2nd ed. Boca Raton, Florida: Chapman and Hall/CRC. `https://yihui.name/knitr/`.

Xie, Yihui, Joseph J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC. `https://bookdown.org/yihui/rmarkdown`.