SHORT TECHNICAL NOTE

New and simplified manual controls for projection and slice tours, with application to exploring classification boundaries in high dimensions

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 July 12, 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 with 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 Fisherkeller, Friedman, and Tukey (1974)'s PRIM-9. PRIM-9 had user-controlled rotations on coordinate axes, allowing one to manually tour through low-dimensional projections. (A video illustrating the capabilities is available through video library of ASA Statistical Graphics Section (2022).) Steering through all possible projections is impossible, unlike Asimov's tours which allows one to quickly see many, many different projections. After Asimov there have been many, many tour developments, which are 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) describe controls for 1D (or 2D) projections, respectively 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) refined this algorithm and implements them to generate animation sequences.

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

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 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 faceted 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. How to construct a manual tour

A manual tour allows the user to alter the coefficients of one (or more) variables contributing to a $d - D$ projection. The initial ingredients are an orthonormal basis ($A_{p\times d}$) defining the projection of the data, and a variable id ($m \in \{1, ..., p\}$) specifying which coefficient will be changed. A method to update the values of the component of the controlled variable V_m is then needed.

2.1. *Existing methods*

The method for updating component values in Cook and Buja (1997) (and utilised in Spyrison and Cook (2020)) are prescribed primarily for a 2D projection, to take advantage of (then) newly developed 3D trackball controls made available for computer gaming. The first step was to construct a 3D manipulation space from a 2D projection. In this space the coefficient of the controlled variable ranges between -1 and 1. Movements of a cursor are recorded and converted into changes in the values of V_m thus changing the displayed 2D projection. The algorithm also provided constraints to horizontal, vertical, radial or angular motions only. The construction of the manipulation space overly complicates the manual controls, especially when considering a technique that will generally apply to arbitrary d.

2.2. *A new simpler and broadly applicable approach*

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. Provide A, and m. (Note that m could also be automatically chosen as the component that is closest to the cursor position.)

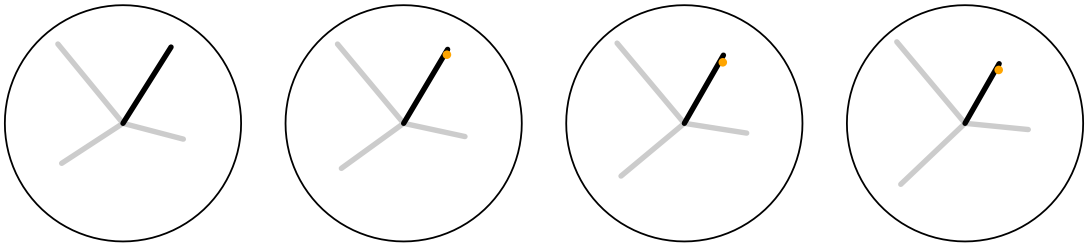

Figure 1. Sequence of projections where contribution of one variable is controlled (black) is changed using unconstrained orthonormalisation. The dot (orange) indicates the chosen values for the controlled variable. It can be seen that the actual axis does not precisely match the chosen position, but it is close.

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 1 (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.

2.3. *Refinements to enforce exact position*

The problem with new simple method is that it is not faithful to the precise values for V_m because the orthonormalisation will change them. There are numerous ways that this can be enforced, and details are provided in the Appendix.

2.4. *Manual control for slices*

To better explore the space we combine the manual controls for the projection with manual controls for slicing. A slice is defined by a projection, a center point that is anchoring it in the high-dimensional space and the slice thickness h (Laa, Cook, and Valencia 2020).

2.4.1. Shifting the center

In the case of a single orthogonal direction on the projection plane we can pick a sequence of center points in steps along that direction to move the slice and fully cover the data space. This no longer works in higher-dimensional spaces, and we can think of picking one direction and shifting the slice along the component orthogonal to the projection plane. In practice we define the center point explicitly and calculate the slice display as explained in Laa, Cook, and Valencia (2020). The current implementation allows the user to individually enter the components of the vector defining the center point, and updates the display by jumping to the new slice.

Potential adjustments for future work include a better visualization of the current center point, potentially with interactive graphics to move the slice similar to how the manual tour was implemented here. An alternative would be a step-wise shift of the slice towards the new center point to get a better understanding of the distribution along the selected direction.

2.4.2. Changing the thickness

In addition it is also useful to interactively change the slice thickness, in particular when selecting the preferred thickness h (also called the slice radius) for exploring the input data. This was implemented as a slicer, and to allow the user to efficiently explore the interesting range we can provide the slider range as an input. For guidance the estimates of the number of points inside the slice as a function of the original sample size N and the number of dimensions p from Laa et al. (2022) can be used: in case of a uniform distribution inside a sphere of radius R a slice with thickness h will contain N_S points, with

$$N_S = \frac{N}{2}\left(\frac{h}{R}\right)^{p-2}\left(p-(p-2)\left(\frac{h}{R}\right)^2\right). \tag{1}$$

The current implementation contains a switch to change between the slice view and the projected view of the data, while keeping all other settings fixed.

3. Experimenting with new techniques using Mathematica

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

4. Application

To illustrate the usefulness of the manual controls we use the 4D penguins data (Horst, Hill, and Gorman 2020). We will show how classification boundaries can be explored and better understood on projections and slices through 4D space. Figure 2 shows a scatterplot matrix of this data. There are four variables (`bl = bill_length_mm, bd = bill_depth_mm, fl = flipper_length_mm, bm = body_mass_g`) measuring the size of the penguins from three species (Adelie, Chinstrap and Gentoo). The scatterplot matrix shows that the three species appear to be likely separable, and that at least the Gentoo can be distinguished from the other two species when `bd` is paired with `fl` or `bm`. The steps for exploring boundaries in this example are as follows:

1. Build your classification model.
2. Predict the class for a dense grid of values covering the data space.
3. Examine projections, using a manual tour so that the contribution of any variable is controlled.
4. Slice through the center, to explore where the boundaries will likely meet.
5. Move the slice by changing the center in the direction of a single variable to explore the extent of a boundary for a single group relative to a variable.

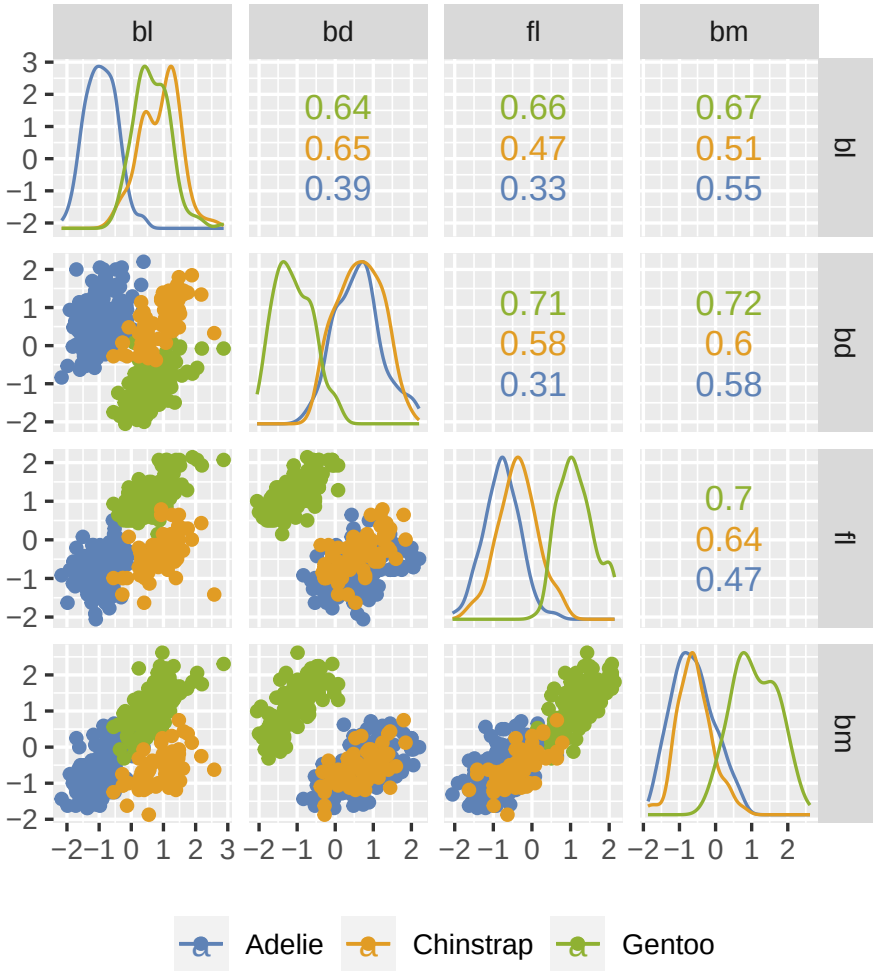

Figure 2. Scatterplot matrix of the (standardised) penguins data. The three species are reasonably different in size, with Gentoo distinguised from the other two on body depth relative to flipper length and body mass.

4.1. *Constructing the 4D prediction regions*

We use the `classifly` package (Wickham 2022) to generate predictions across the 4D cube spanned by the data, with two classification models: linear discriminant analysis (LDA) and random forest (RF).

4.2. *Exploring projections manually*

We start by first exploring the projections of the model prediction. Figure 3 summarises the process. Through manual rotation of the view we can get a feeling for where in the space we primarily predict each of the three species, and we also get a sense of the difference between the two models. To illustrate this difference we have manually rotated the projection for the RF model (left plot) to identify a projection that shows the non-linear but block-type structure that is typical of this type of model. This particular projection (A_1) is exported so that the same projection can be used to show the LDA model (middle plot) and the actual data (right plot). What can be seen is the linearity of the LDA model, where the boundaries are linear and oblique to the variable axes. And, interestingly this particular projection of the original data shows very distinct clusters of the three species. That means, the obscuring of the boundaries between groups for both of the models is driven by what is happening in the orthogonal space to the plane of the selected projection.

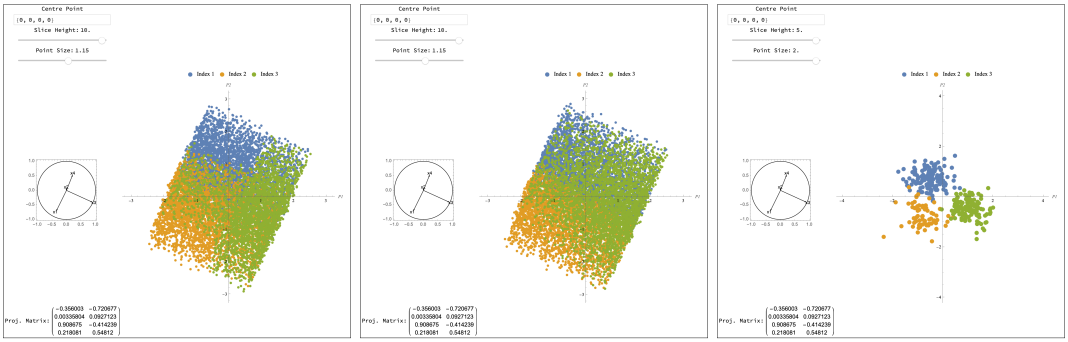

Figure 3. Projection identified using the manual tour, because it reveals an interesting structure in the predictions from the RF model (left). We can clearly see a block structure, while the LDA model (middle) produces linear boundaries. The three groups are nicely separated in this projection of the data (right)

.

4.3. *Slicing through the center*

XXXX currently in the notebook A_1 is called A5, and A_2 is called A6

We continue the investigation by now slicing orthogonally to the projection A_1. For both models we look at a thin ($h = 0.5$) slice through the center, $S_1^0 = (0, 0, 0, 0)$. At first we explore how changing the projection, and thus the slice) away from A_1 can help with understanding the boundary better. For our example, notice that A_1 does not contain any contribution from the second variable (`bd`), so we will first rotate this variable into the view.

Figure 4 shows snapshots of the exploration. The rop row is the initial projection (A_1 and S_1^0), and the bottom row is a later projection containing more of `bd` (call them A_2 and S_2^0). The columns correspond to the two models, with Rf on the left and LDA on the right.

Both models have some overlap at the center, even for this thin slice (S_1^0). The second slice (S_2^0) mostly resolves this overlap and reveals the primary differences between the models. The boundary between Adelie (blue) and Chinstrap (yellow) is very similar for both models. The boundary between Chinstrap and Gentoo (green) is where they differ.

The RF is almost straight in this view, as something we might expect from a tree model where splitting occurs on single variables only. However, it is straight in a combination of the second and third variable (`bd` and `fl`). By examining the scatterplot matrix, we can understand how this boundary was built: on each of `bd` and `fl` it is possible to cut on a value where most of the Gentoo penguins are different from the other two speciesfor any tree, and likely only one is needed. Thus the forest construction is providing a sample of trees that use one variable or the other variable to split, producing the blocky boundary on the combination of the two.

In contrast, the LDA boundary has an oblique split between the two, and roughly divides the space into three similarly sized areas. It is almost like a textbook illustration of how LDA works for two variables (2D) where by assuming the clusters have equal variance-covariance, it places a boundary half-way between the group means.

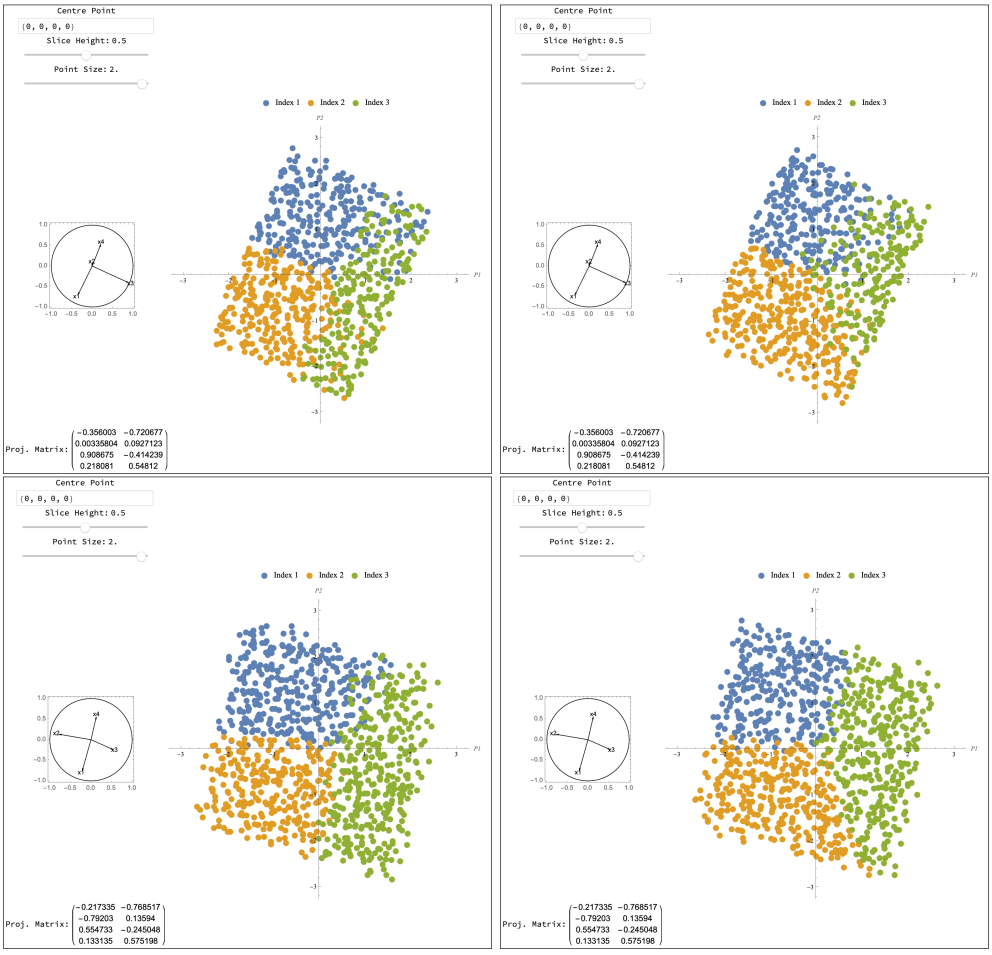

Figure 4. Comparing slices based on two projections A_1 (top row) and A_2 (bottom row), for the two models RF (left) and LDA (right). With A_1 we see two groups overlap (green - Gentoo with yellow - Chinstrap), while the rotation to A_2 results in clear boundaries inside the slice. The boundary between Adelie (blue) and Chinstrap (yellow) is similar for both models but very different between Chinstrap and Gentoo (green).

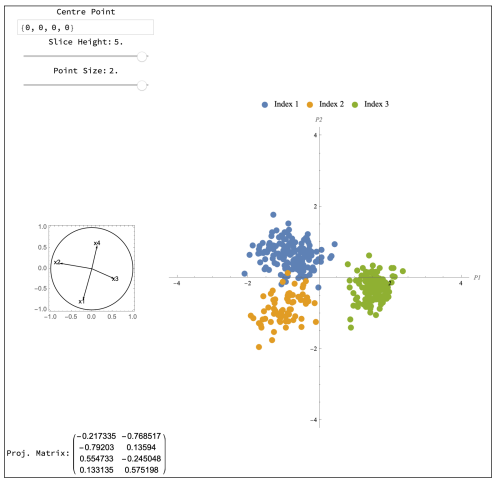

Figure 5. Projection of the data based on A_2. Compared to projecting onto A_1 we see that the green observations (Gentoo) are more separated from the other two species.

Finally, to determine which model does the boundaries better, compare them with the A_2 projection of the data (Figure 5). Gentoo is more separated from the other two in this projection, and one can imagine that the trees in the forest has greedily grasped any one of many places to make a split to separate the group. It might be argued though the that RF boundary is cut too close to the Chinstrap species, and might lead to some unnecessary misclassification with new data. The LDA boundary is better placed for all species.

4.4. *Shifting the slice center*

We have seen that starting from A_1 using the manual controls to change the contribution of the second variable we could find a clear separation boundary indicating the relation between this variable (bill depth) and the Gentoo penguin species. Instead of rotating to a different projection, we might also change the view by moving the slice along one axis in the 4D space. Here we will continue our exploration of the dependence on `bd` and move the slice defined by A_1, S_1^c to either large positive or negative values ($c_{bd} = \pm 1.5$ after centering and scaling). We will label these slices as $S_1^{\pm}$. Here we will also look at slices of the observed data points, using a thicker slice ($h = 1.5$) to capture enough points in a given view.

We start by a comparison of the two models and the data distribution in S_1^+, thus the slice is localized towards high values of bill depth in Figure 6. We can see that all three slices (the two models and the data) contain almost no points from the third class (green, Gentoo), and that the decision boundary between the two models is very similar.

A more interesting comparison is found for S_1^-, thus the slice localized towards low values of bill depth, shown in Figure 7. The RF model (left) predicts all three species within this slice, with an interesting boundary for the third class (green, Gentoo). On the other hand the LDA model (middle) predominantly predicts the third class within the slice, this appears to be enforced through the linear structure of the model. Looking finally at the thick slice through the data we see that there are primarily observations from this class within the slice we can conclude that the two models have filled in the "empty" space (where we do not have any training observations) in very

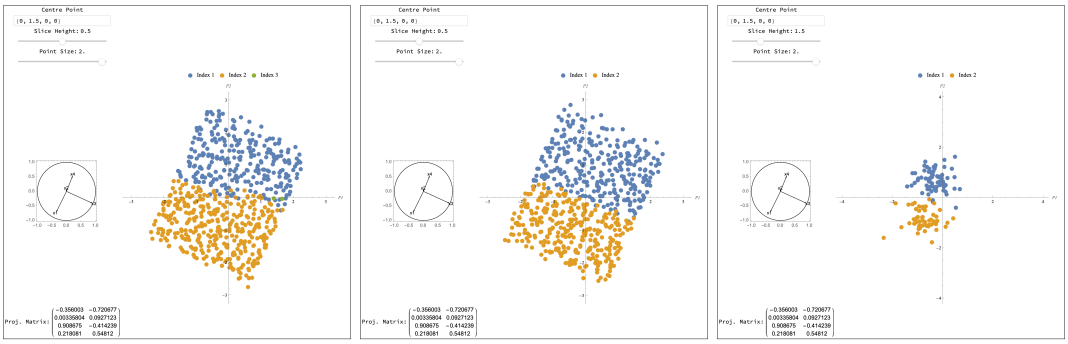

Figure 6. Shifting the slice center in the positive direction of bill depth produces regions that have no Gentoo (green). The two models have similar boundaries except that the RF is more of a step function.

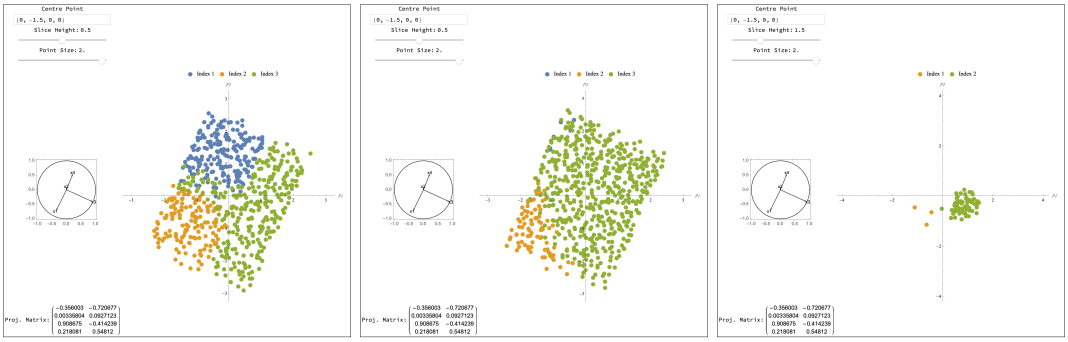

Figure 7. Shifting the slice center in the negative direction of bill depth produces regions that are mostly Gentoo (green). The two models have very different boundaries: the nonlinear potential of RF can be seen here where still some subspaces would predict to be Adelie and Chinstrap.

different ways and according to what we might expect given the model structure.

Finally it is interesting to compare the slice views to the projection of the models seen in Fig. 3 to better understand how the boundaries change along the `bd` direction and where the differences in the projections come from.

5. Extension added in the R package `tourr`

Explain `radial_tour`

6. What would be desirable for implementations in R?

7. 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

ASA Statistical Graphics Section. 2022. "Video Library." https://community.amstat.org/jointscsg-section/media/videos.

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.

Fisherkeller, M.A., J.H. Friedman, and J.W. Tukey. 1974. "PRIM-9, an interactive multidimensional data display and analysis system." In *The Collected Works of John W. Tukey: Graphics 1965-1985, Volume V*, edited by William S. Cleveland, 340–346.

Horst, Allison Marie, Alison Presmanes Hill, and Kristen B Gorman. 2020. *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*. R package version 0.1.0, https://allisonhorst.github.io/palmerpenguins/.

Laa, Ursula, Dianne Cook, Andreas Buja, and German Valencia. 2022. "Hole or Grain? A Section Pursuit Index for Finding Hidden Structure in Multiple Dimensions." *Journal of Computational and Graphical Statistics* 0 (0): 1–14. https://doi.org/10.1080/10618600.2022.2035230.

Laa, Ursula, Dianne Cook, and German Valencia. 2020. "A Slice Tour for Finding Hollowness in High-Dimensional Data." *Journal of Computational and Graphical Statistics* 29 (3): 681–687. https://doi.org/10.1080/10618600.2020.1777140.

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.

Wickham, Hadley. 2022. *classifly: Explore Classification Models in High Dimensions*. R package version 0.4.1, https://CRAN.R-project.org/package=classifly.

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.