

Interactively Visualizing Multivariate Market Segmentation Using pytourr

Matthias Medl 

Institute of Statistics
BOKU University
Vienna

Dianne Cook 

Econometrics and Business Statistics
Monash University
Melbourne

Ursula Laa 

Institute of Statistics
BOKU University
Vienna

Abstract

Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum

Keywords: interactive graphics, tourr, exploratory data analysis, R, python.

1. Introduction

Clustering algorithms are often used to find a smaller number of observations (the cluster means) that adequately summarize a much larger number of observations. For market segmentation, clustering can allow partitioning of observations into a small number of groups, by incorporating associations between the variables. Market segmentation supports targeted approaches to different groups of customers based on common traits. It provides a data-driven solution to partitioning customer data. Leisch, Dolnicar, and Grün (2018) provides an extensive overview of using clustering for market segmentation.

A difference between clustering analysis and partitioning is typically the nature of the data. With cluster analysis, we usually envision data that contains separated clusters, and a successful clustering result is one that divides the data based on these gaps. With partitioning, it is usual that there are no gaps in the data, but it is still useful to partition the data. Figure 1 illustrates how the k -means algorithm would partition a 2D data set into four groups depending on the correlation between the two variables. When the correlation is high, the partitioning will be along the combination of variables that produces the highest variance. With lower correlation, it will segment the bottom and top, and divide the middle into two parts in the opposite direction. When there is no association, the partitioning is radial like a windmill.

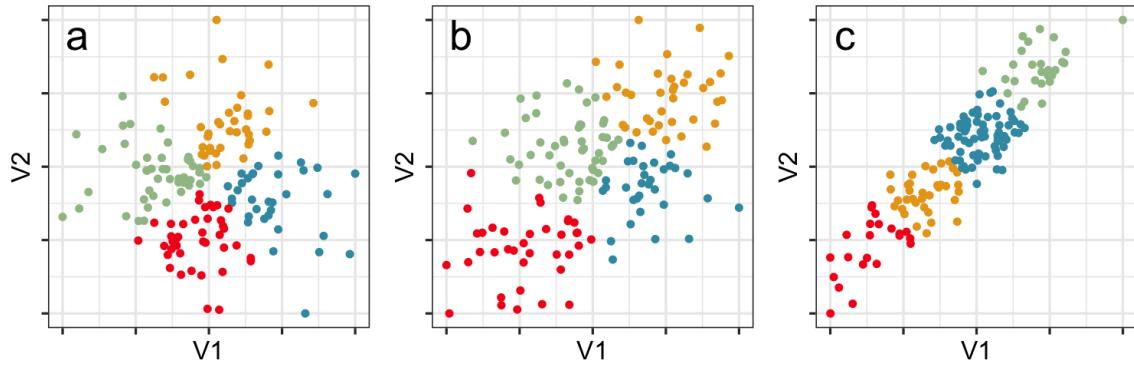


Figure 1: Examples of how k -means partitions 2D data with different association structure. If the correlation is high (c), the partitioning happens along the primary direction of the association.

One can see that the data is perfectly divided into four parts. However, if one were to plot the two variables individually, this would not be obvious. Figure 2 shows histograms of the two variables, V1, V2, with the colour matching the four partitions. From the histograms, we can see some differences in the partitions for the four different association structures, but they are all overlapping. The distinct border between the partitions can only be seen from the scatterplots of both variables.

When there are more than two variables, histograms of the individual variables are commonly used to display the partitioning results. This means that the analyst likely cannot understand how the partitioning divides the data. All that they can observe is roughly how the individual variables relate to the partition, which is useful but inadequate. Here is where using tour methods to view high dimensions can be helpful. A tour (XXX general tour and recent tour) can be used to show scatterplots of combinations of variables and thus provide views like that in Figure 1 where distinct differences between partitions can be observed. High dimensions are still tricky, and a combination of animations of the linear combinations, and interactive control (XXX REF radial/manual tour) over the combinations is important. A scatterplot of a combination of variables can be considered to be a projection of the data, and thus like a shadow of a 3D object, some aspects of the data (object) can be obscured. Using slices of the projected data (XXX REF) can be a useful addition to projections. This paper illustrates how to do this to better understand partitioning results for multivariate data.

This paper is organised as follows XXX The methods are illustrated using Austrian and Australian tourism data provided in [Leisch et al. \(2018\)](#).

2. Interactive interface for partitioning

The aim of this work is to build an interface that allows for an interactive exploration of partitioning solutions of multivariate data. The interface should enable

- visualizing a clustering solution with different tour methods, including manual changes to the viewing direction (manual tour)
- brushing of (groups of) points for manual refinement of the cluster solution (spin-and-brush capabilities)
- additional linked displays for comprehensive exploration of the solution
- ability to update the displays based on user selections

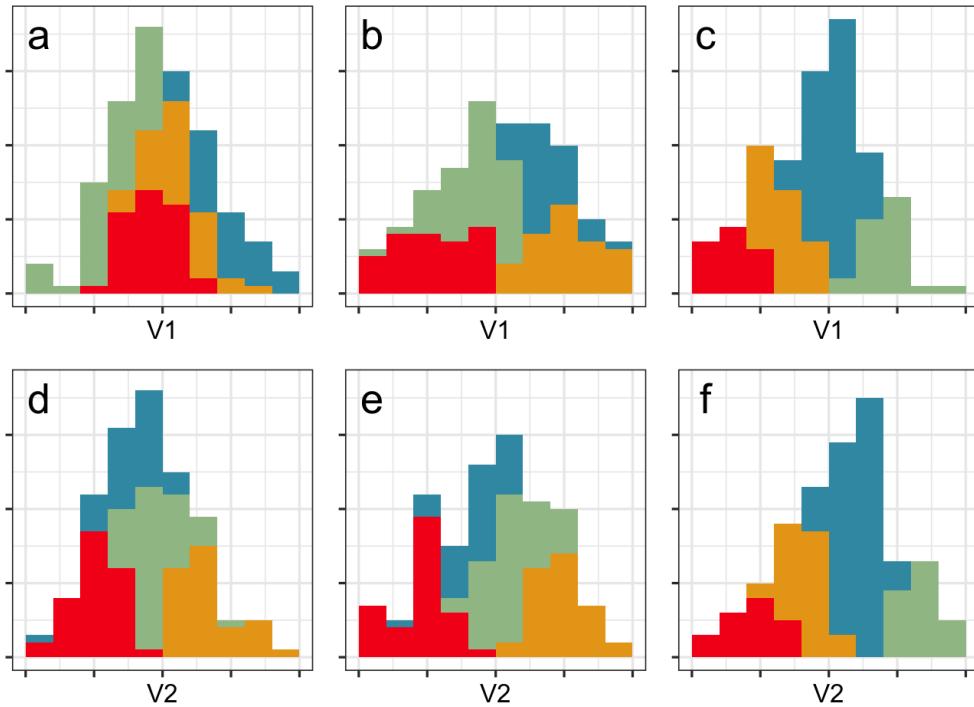


Figure 2: The four k -means partitions plotted as histograms of the individual variables. While differences between groups can be seen, the clear separation cannot. This is important for understanding why high-dimensional visualisation methods are useful for summarising partitioning results.

While tour animations are best obtained within R using the `tourr` package (Wickham, Cook, Hofmann, and Buja 2011), it does not enable the interactivity required for example for a manual tour (Laa, Aumann, Cook, and Valencia 2023). Interactive graphics are available when using Javascript, as implemented in the `detourr` package (Hart and Wang 2022). This allows to replay a recorded tour path with interactive graphics, and can also be linked with additional displays, but lacks capabilities for manual tours.

To integrate user interactions with the capabilities of the `tourr` package an active communication with the interface is required. For example, we may wish to explore the local neighbourhood of a projection selected by the user with a local tour animation provided by `tourr`, or we may want to optimize a guided tour path using groups identified via brushing. Our solution is using Python for high-performance interactivity, through the packages `TKinter` (Lundh 1999), `CustomTKinter` (Schimansky 2024), and `matplotlib` (Hunter 2007), with integration to the `tourr` package (Wickham *et al.* 2011) via `reticulate` (Ushey, Allaire, and Tang 2024), a framework that facilitates seamless interoperability between Python and R. *also rpy2?*

The interface was implemented in `pytourr` and offers a variety of linked interactive plot types, providing users with the flexibility to visualize their data from multiple perspectives. The ability to navigate through various projections of the displayed tours directly within the GUI enables users to explore different aspects of the dataset. Furthermore, users can initiate new tours directly from the interface. The GUI also supports interactive variable selection, allowing users to specify which subset of variables should be visualized in the plots. Once users have identified interesting views or settings, `pytourr` allows them to save the displayed projections, subsets, and plots. This functionality ensures that analysis states can be preserved for further examination or reporting, making the package particularly useful for iterative analysis where findings may need to be revisited or shared with collaborators.

With its high level of interactivity, performance, and ease of use, `pytourr` streamlines the

exploration of complex datasets, offering a powerful tool for researchers working with high-dimensional data.

2.1. The graphical user interface

The `pytourr` GUI can be launched using the function `interactive_tour()`. At a minimum, the user needs to provide both the dataset and the instructions for constructing the desired plots. The dataset must be supplied as a `data.table`, while the plotting instructions should be passed as a list containing the named elements `type` and `obj`. The `type` element specifies the type of display to generate, such as "scatter" for a scatterplot or "2d_tour" for a 2-dimensional tour. The `obj` element further defines the properties of the chosen display. For example, to create a 2-dimensional tour, the user must provide a `tour_history` object, which can be generated using the `tourrr` package. For a scatterplot, the user needs to provide a vector of strings specifying the names of the features to be displayed. The user can optionally specify the feature names, the arrangement of the plots, predefined subsets of the data (e.g., cluster solutions), custom names for these subsets, the number of available subsets, and the size of each plot.

The GUI is divided into two main sections: a sidebar on the left, which contains a comprehensive set of interactive controls, and the display area on the right, where the selected plots are shown (see Figure 3).

At the top of the sidebar, users can select and deselect features using checkboxes (Figure 3C), controlling which features are displayed in the plots. Below this, each subset has its own checkbox to designate the active subset (Figure 3D). When data points are manually selected in the plots, they will be assigned to the active subset and colored accordingly. For scatterplots, data points can be selected by encircling them directly on the plot while holding down the left mouse button. For barplots, clicking on a specific bar selects the data represented by that bar. The colored boxes next to the subset names indicate the assigned colors for the data points. Clicking on these boxes adjusts the transparency of the points, which is helpful for highlighting and comparing subsets. Subsets can also be renamed using the provided text boxes. The "Reset original selection" button allows users to revert the subset selections to their initial state.

The frame selection interface (Figure 3E) displays the frames of the currently shown tours and enables users to jump between frames of the tour objects. Below this, users can save and load projections and subsets using the respective buttons (Figure 3F). New tours can be started using the current settings via the interface at the bottom (Figure 3G), and users can select different metrics for some plot types e.g. heatmaps (Figure 3H).

One can move through the projections of a tour by pressing the arrow keys or by specifying the index of the projection to be displayed and pressing the "Update frames" button. Users can animate the tours by toggling the "Animate" checkbox and specifying a time interval, so that the displays automatically move to the next projections after the specified time interval.

The "Save projections and subsets" button can be used to save the current state of the analysis. These states can be recovered by using pressing the "Load projections and subsets" button. Additionally, users can initiate new tours of various directly from the sidebar.

2.2. R/Python interface

The majority of `pytourr` was written in Python, while the R side of the package handles setting up the Python environment within the interactive interface is being run, launching the interactive tour, and generating new tours when initiated through the interface. To incorporate the functionality of the `tourrr` package without translating large portions of its code from R to Python, the `reticulate` package was used. This approach allows `pytourr` to automatically benefit from updates to `tourrr`.

However, to reduce inefficiencies caused by cross-language communication and to simplify

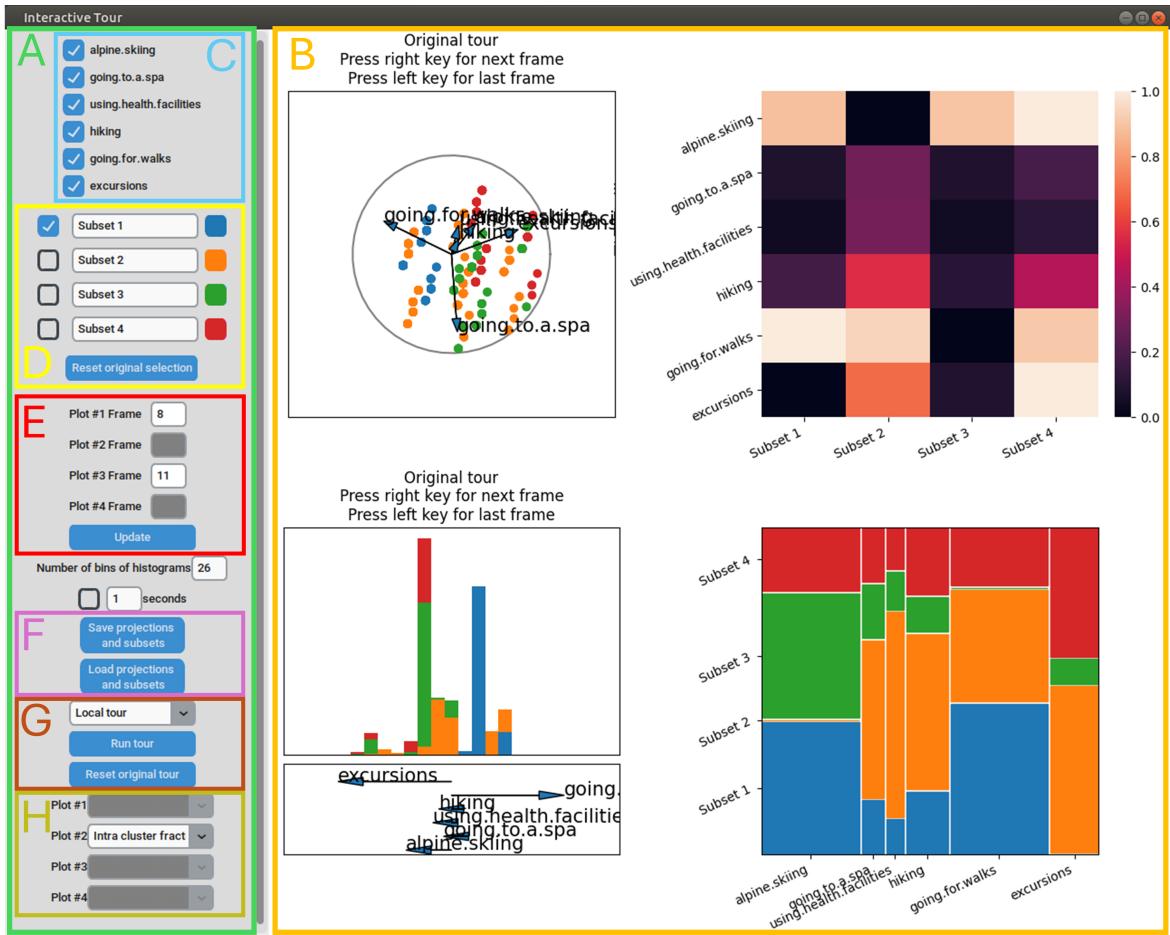


Figure 3: Overview of the GUI, consisting of the sidebar (left) and display area (right). A: Sidebar. B: Display area. C: Feature selection checkboxes. D: Subset selection with color indicators. E: Frame selection interface. F: Save and load buttons. G: Interface for starting new tours. H: Metric selection interface.

debugging, `tourr` functions were only accessed when necessary. Core functionalities, such as performing data transformations using projections, were implemented directly in Python. These transformations are based on well-established mathematical principles and are straightforward to replicate, ensuring they remain stable and efficient.

2.3. Structure of the Python code

The central component of the Python code is the `customTKinter` class `InteractiveTourInterface`. This class centrally stores attributes related to all plots, such as the dataset, subselections, feature selections, and other shared information. Plot-specific data is organized in dictionaries (the Python equivalent of named lists in R), including the display type, construction instructions, tour projections (only in case of tour displays), color schemes for the displayed data, and, where applicable, the selector and manual projection manipulation classes.

The selector classes handle the behavior when users manually select data points to move them to the active subset. After a selection is made, the selector class updates the centrally stored subselection attribute and ensures all other displays reflect these changes. The manual projection manipulation classes construct the arrows representing the projection axes in the displays and manage the manual adjustment of projections, and thus enable manual tours. Users can right-click and drag the arrowheads to modify the projections, after which the class orthonormalizes the projection axes and updates both the projection and the transformed data accordingly.

2.4. Bit blit

The implementation of bit blitting was crucial to ensuring fast plot updates and providing a smooth user experience. With bit blit, the static elements of the display, such as the outer frames of the plots, are stored as a background image. When a plot is manipulated, only the affected plot is updated, and within that, only the interactive elements, such as the data points and projection axes in a 2-dimensional tour, are rendered on top of the background image.

In practice, this means that the background, without the interactive elements, must be captured either during initialization or after major updates. The entire plot is first rendered without the interactive elements, the background is then saved, and finally, the plot is redrawn with the interactive elements blended in. Since this process is relatively slow, full updates are only triggered during initialization or after significant changes, such as modifications to the set of active features.

3. Workflow with the pytourr package

Before launching the pytourr interface the user should have performed a partitioning of their choice, and provide the initial clustering solution used by the interface. We launch the interface to explore and potentially refine this solution.

3.1. Variable selection

A first step may look into what variables are important for the cluster solution. The interface provides different capabilities that support variable selection. First, a 2D tour view can be used to understand the sensitivity of the grouping to individual variables, in particular when using the manual tour to remove one variable at a time. This is in particular useful when the starting view was obtained through a projection pursuit guided tour REF that optimized for the separation between the labeled groups REF.

In our applications we are considering market segmentation analysis on binary variables. For this setting we have developed a heatmap display that we have found to be particularly useful in variable selection. The idea is to understand differences in the binary ratings between clusters, where we use different normalizations to better explore the data. For c_{ij} the count (number of true values in the binary variable) in the i -th cluster and j -th variable, we define

$$\text{Overall count fraction } f_{ij} = \frac{c_{ij}}{n}$$

with normalization to the total number of observations n for a general overview,

$$\text{Cluster count fraction } f_{ij} = \frac{c_{ij}}{n_i}$$

showing the counts relative to the cluster's size n_i , to see the fraction of observations in cluster i for which variable j was true, and

$$\text{Variable count fraction } f_{ij} = \frac{c_{ij}}{c_j}$$

to see how the true values for variable j are distributed across the different clusters relative, with normalization to the overall count of that feature $c_j = \sum_i c_{ij}$.

different symbol for each? indicate on the index value somehow?

3.2. Subset selection

The spin-and-brush approach suggests to cluster data manually when using a tour: we run a tour animation, stop when we see a group of points that are different from the rest of the

distribution, brush them, and then continue. Different projections will enable the separation of different groups, and for well-separated clusters we will be able to recover a full cluster solutions in this manner.

A similar approach can be used to refine a partitioning solution. This is in particular useful to integrate prior knowledge or business interests in a given cluster solution. In the interface we can keep the provided clustering, but separate out new subsets via manual selection, for example after we found a group of particular interest via a manual tour.

3.3. Reproducibility

Ensuring the reproducibility of data analysis is a fundamental principle in scientific research. It allows others to verify the validity of the findings and is key to the integrity of the scientific process. Reproducibility not only builds trust in the research outcomes but also enables the scientific community to build upon existing work. When analyses can be replicated, it can be validated whether the conclusions drawn from the data are robust and not dependent on the specific conditions or idiosyncrasies of the original analyst. Moreover, reproducible research can serve as a foundational building block for subsequent studies, fostering incremental advancements in knowledge.

One challenge in the context of interactive data analysis is that not all steps of the analysis are precisely documented in the form of code, especially when using graphical user interfaces (GUIs) where user-driven interactions might not leave a traceable history. This lack of documentation can hinder the ability of others to reproduce the analysis or to understand how specific results were obtained. To mitigate this challenge, it is essential to implement mechanisms that allow users to easily save and share intermediate snapshots of their analyses.

One measure to combat this is to make saving intermediate snapshots of the analysis easy and accessible. Specifically, the "Save projections and subsets" button enables users to take snapshots of their analysis, including visual representations, selected data, and parameter settings. Upon pressing this button, a file browser is triggered, allowing users to specify the destination for saving these snapshots. Each save operation generates multiple files:

- A `.png` file containing the currently displayed graphics,
- `.csv` files that capture the feature and subset selection as well as projections of the tours displayed at the time of the snapshot,
- two `.pk1` files that contain state variables of the GUI, allowing for complete recovery of the snapshot.

These files provide dual utility. First, they allow users to fully recover the state of the analysis within the GUI. This can be achieved either by using the "Load projections and subsets" button, or by launching a new GUI instance with the `load_interactive_tour()` function. The latter approach, using `load_interactive_tour()`, has the added flexibility of only requiring the original dataset and the directory containing the saved files. This function also allows users to modify display settings, such as adjusting the size of the interactive plots or changing the arrangement of the display grid. In contrast, when loading the saved state directly from within the GUI, it is crucial that the active session was initiated with the same dataset and plot objects that were present at the time of saving. This ensures that the analysis environment is accurately replicated.

Second, the saved `.csv` files provide a way to inspect and further analyze the data outside of the original interface. This opens up opportunities for deeper analysis and extensions of the work.

This level of interactivity and documentation is crucial for reproducibility, as it ensures that even exploratory, interactive data analysis can be retraced and validated by others. Ulti-

mately, these features facilitate a reproducible workflow that balances the flexibility of interactive exploration with the rigor of reproducible research.

4. Applications

The Austrian Vacation Activities [Dolnicar and Leisch \(2003\)](#) and the Australian Vacation Activities [Cliff \(2009\)](#) datasets are used to illustrate the methods.

4.1. Austrian Vacation Activities dataset

The Austrian Vacation Activities dataset comprises responses from 2,961 adult tourists who spent their holiday in Austria during the 1997/98 season. Participants were asked to evaluate the importance of 27 different activities during their vacation. The survey categorized responses based on four levels of importance: "totally important," "mostly important," "a bit important," and "not important." For analysis, the responses were binarized: a value of 1 was assigned if the activity was rated as "totally important," and a value of 0 if any of the other categories were selected. The survey was conducted by the Europäisches Tourismus Institut GmbH at the University of Trier and focused exclusively on tourists who did not stay in the country's capital cities.

To gain further insight into the dataset a k-means clustering as described in the book "Market Segmentation Analysis" has been performed ([Leisch et al. 2018](#)). Therefore, the function `stepcclust` of the R package `flexclust` ([Leisch 2006](#)) with $k=6$ and $nrep=20$ was used.

Feature selection

The original dataset contains $p=27$ features. Some of these features are more informative than others. We only want to keep the most informative ones. Additionally, interacting with the pytourr GUI becomes cumbersome when handling more than ~ 15 features, making it necessary to reduce the dimensionality of the dataset. An effective and intuitive way to perform feature selection is by using the heatmap display within pytourr.

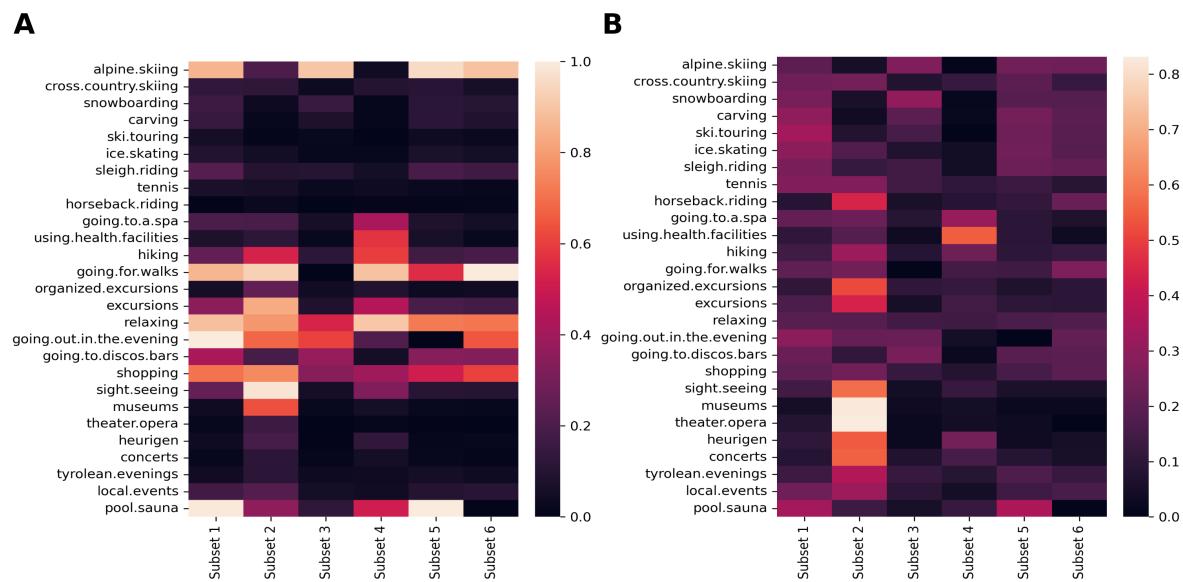


Figure 4: A: heatmap displaying the intra cluster fraction. B: heatmap displaying the intra feature fraction.

In Figure 4A, we can observe the general interests of tourists within each cluster. For instance, tourists in clusters 1, 3, 5, and 6 predominantly engaged in alpine skiing, while those in clusters 2 and 4 did not. Additionally, we see that activities such as ski touring and horseback riding

were generally unpopular across all clusters.

In Figure 4B, we can determine whether tourists in a particular cluster had a strong preference for certain activities. For example, nearly all tourists who visited museums are grouped in cluster 2, and those who used health facilities are primarily attributed to cluster 4. We can also identify activities that were similarly popular across all clusters, such as relaxing.

By using the heatmap with these metrics, we can perform feature selection by removing unpopular activities and those that were consistently similar across all clusters. After performing the feature selection by unchecking the corresponding checkboxes in the GUI using this strategy, the following 12 activities remained: alpine skiing, going to a spa, using health facilities, hiking, going for walks, excursions, going out in the evening, going to discos/bars, shopping, sightseeing, museums, and pool/sauna.

We can now repeat the k-means clustering with `stepcclust` on the reduced dataset. Silhouette plots of both cluster solutions can be seen in Figure 5. By comparing both silhouette plots, we can see that the cluster solution with the reduced dataset results in a clustering of higher quality. Thus, we will continue with the analysis on the reduced dataset with the corresponding cluster solution. We can also see in Figure 5B that cluster 3 is of comparatively low quality. It is important to note that the silhouette scores were generally quite low, reflecting the lack of clearly separable clusters in the data. Consequently, there is no objectively correct or optimal solution. The primary aim of this analysis, therefore, is to gain insights into the data rather than to identify the optimal clustering configuration.

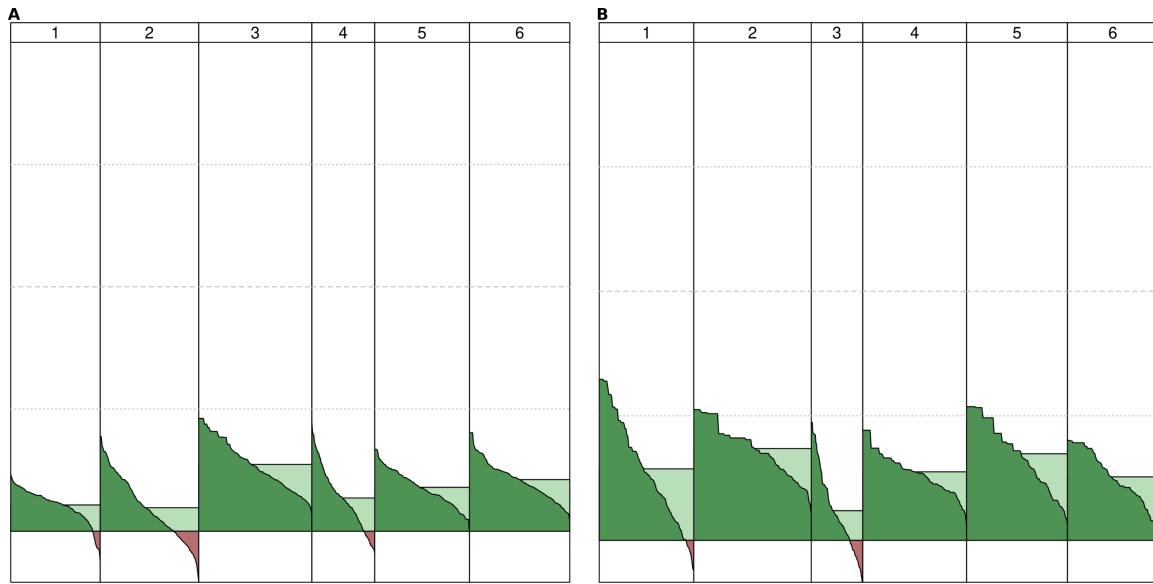


Figure 5: A: silhouette plot of the k-means solution of the full dataset. B: silhouette plot of the k-means solution with the reduced dataset.

We can further explore the similarities between the clusters by initializing an `interactive_tour()` with a 2D tour based on the linear discriminant analysis (LDA) projection pursuit index. By navigating through the tour, we can observe various projections, and when a projection that separates the clusters is found, we highlight each cluster sequentially. The different highlighted clusters can be seen in Figure 5.

This process allows us to visually assess the separation and similarities between the clusters, providing insight into the structure of the dataset. By highlighting each cluster individually, we can evaluate their distinctiveness in different projections. The most influential features shown in Figure 6 are pool/sauna, alpine skiing, museums, and going to the spa. The projection roughly separates clusters 1, 2, and 3 from each other and the other three clusters, which appear to be quite similar.

By manually manipulating the projection axes or initiating a local tour, we can gain further

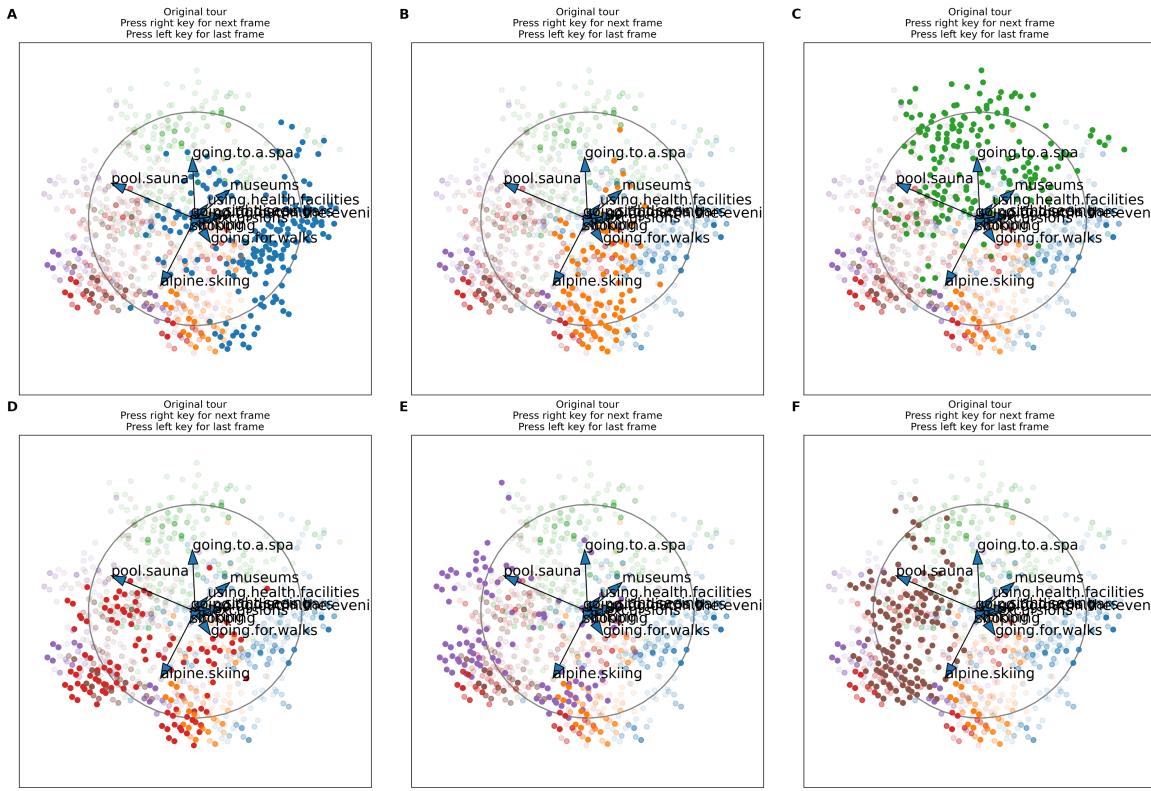


Figure 6: Cluster highlights: **A** - Cluster 1 (blue), **B** - Cluster 2 (orange), **C** - Cluster 3 (green), **D** - Cluster 4 (red), **E** - Cluster 5 (violet), **F** - Cluster 6 (brown).

insight into the similarities between the different clusters. This interactive exploration allows for a more nuanced understanding of the relationships between clusters and the influence of key features on the separation of the data.

Redefining cluster assignments - learning more about museum goers

There are several reasons why we might want to manually modify a clustering solution. One is to capture observations that do not fit well within their assigned clusters. Another reason is to explore specific features in more detail. The advantage of manual cluster selection is that it preserves most of the current clustering structure, allowing us to adjust specific parts of the solution without starting from scratch. This approach is particularly useful when we already have a cluster solution that reveals interesting patterns in the data.

In Figure 5, we observed that clusters 1 and 3 contained data points that did not fit well into their respective clusters. To further investigate this, we can initialize an `interactive_tour()` with the following components:

- A 2D tour using the linear discriminant analysis projection pursuit index,
- A heatmap showing the intra-cluster fraction,
- A 1D tour with the linear discriminant analysis projection pursuit index, and
- A mosaic plot.

This setup produces the display shown in Figure 7. It can also be seen that both clusters 1 and 3 contain tourists that didn't go alpine skiing and the main difference between them is that tourists in cluster 3 enjoyed going to the spa and health facilities as well as going to the pool, while the ones in cluster 1 didn't. Other than that, the clusters were mostly similar. Now we might be interested in the subset of tourists that enjoy going to museums. Therefore,

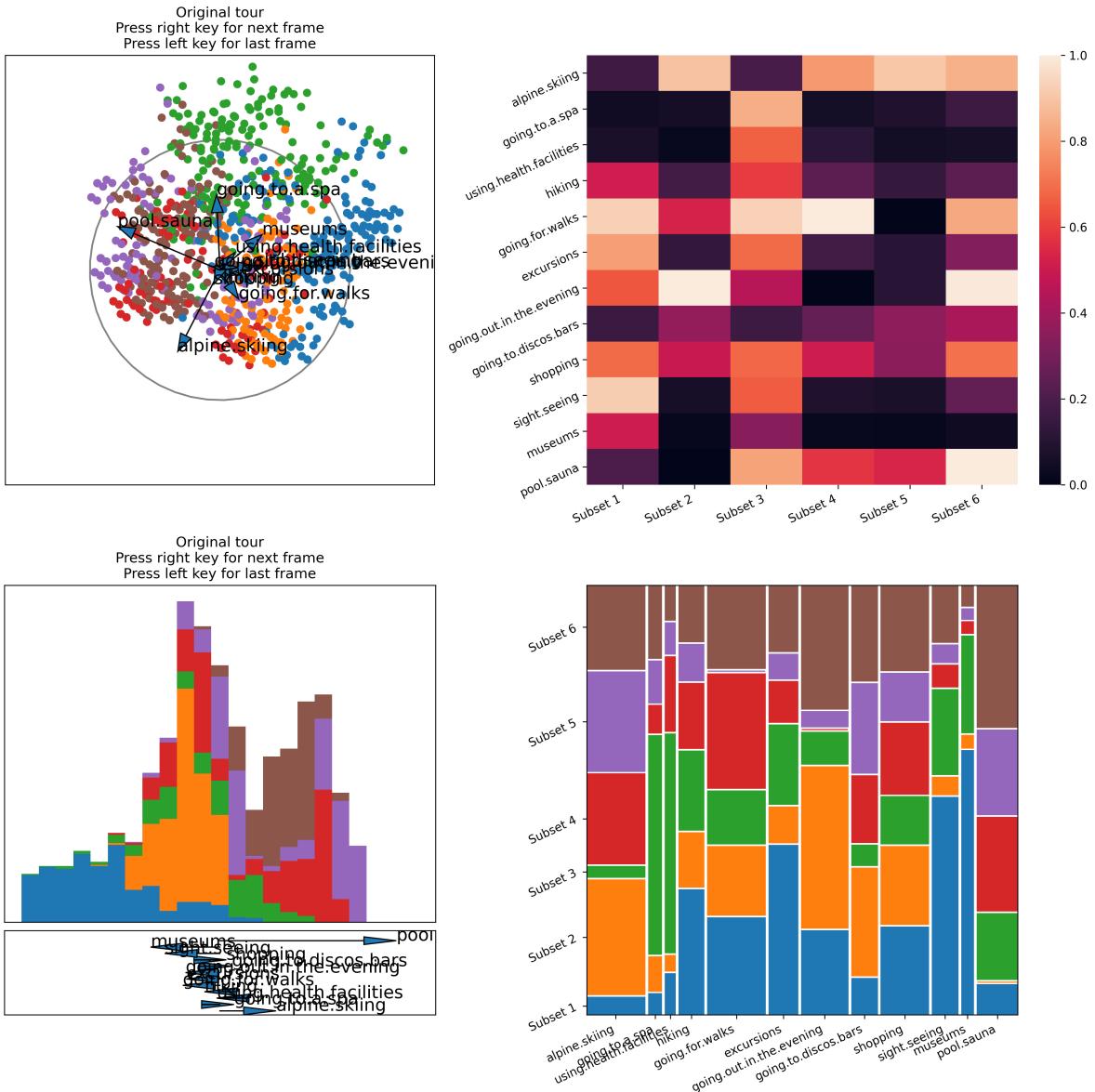


Figure 7: Top left: 2D tour with the linear discriminant analysis projection pursuit index. Top right: heatmap with the intra-cluster fraction. Bottom left: 1D tour with the linear discriminant analysis projection pursuit index. Bottom right: mosaic plot.

we can adjust the projection axes so that these axes are elongated and point into different directions. We can see that there is indeed overlap between clusters 1 and 3, as shown in Figure 8. As a next step we can reassign the overlapping section to a new cluster - cluster 7. By selecting the checkbox for subset 7 and manually selecting the region of overlap, we can form a new cluster, which is visualized in Figure 9.

We can observe slight behavioral differences between tourists in clusters 1 and 7. Tourists in cluster 7 all enjoyed both museums and sightseeing, whereas most tourists in cluster 1 engaged in sightseeing but showed no interest in museums. Instead, participants in cluster 1 exhibited a greater preference for hiking. Despite this, tourists in both clusters generally shared similar interests. This insight could be valuable for enhancing museum marketing strategies. While clusters 1 and 7 have overlapping interests, it appears that current marketing efforts may not effectively reach tourists in cluster 1. By increasing targeted marketing at hiking trails, popular excursion destinations, and shopping centers, it may be possible to attract more interest in museums from tourists in cluster 1.

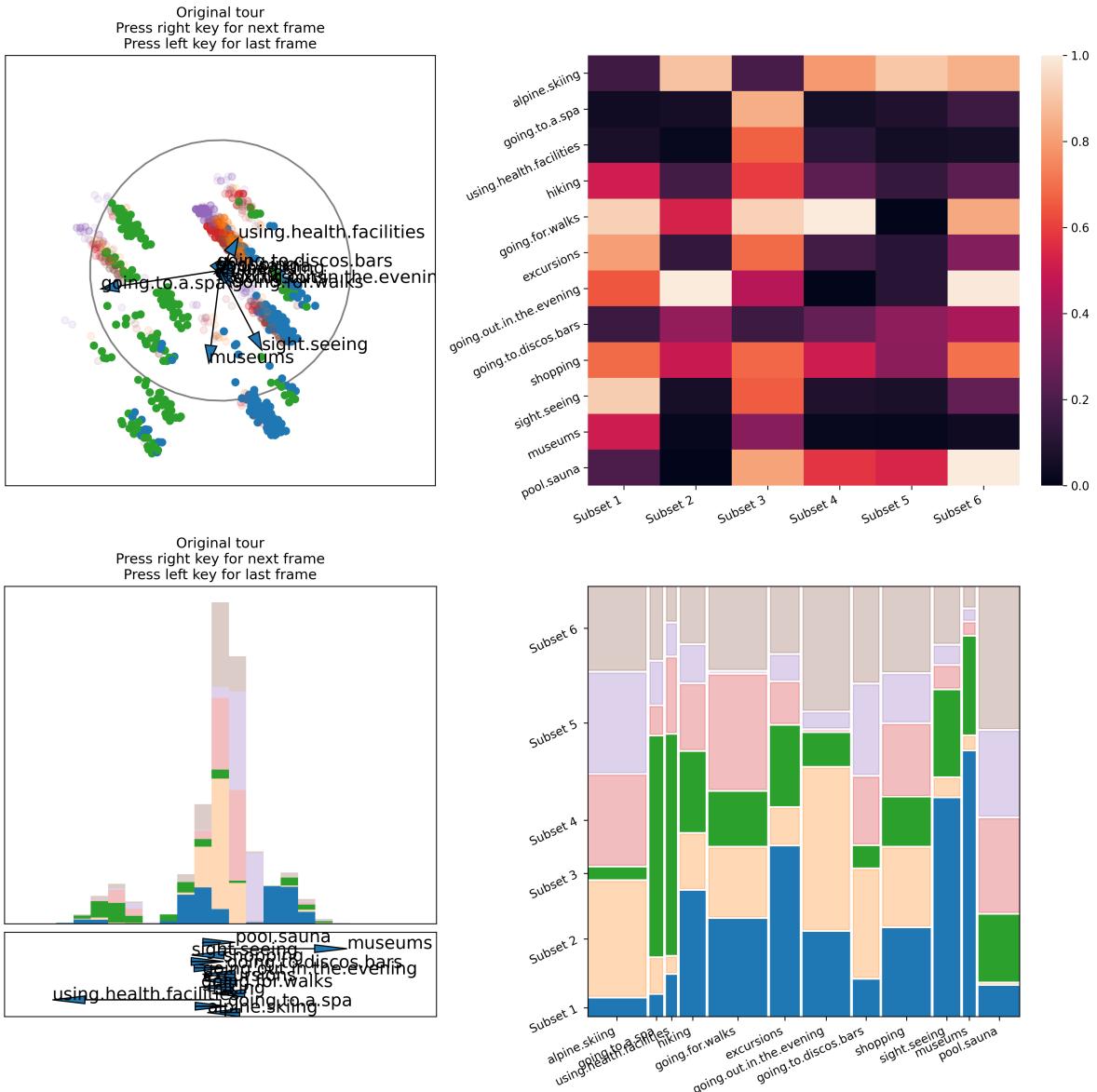


Figure 8: Top left: 2D tour with the linear discriminant analysis projection pursuit index. Top right: heatmap with the intra-cluster fraction. Bottom left: 1D tour with the linear discriminant analysis projection pursuit index. Bottom right: mosaic plot.

4.2. Australian Vacation Activities dataset

The second dataset, the Australian Vacation Activities dataset, includes responses from 1,003 adult Australians who were surveyed through a permission-based internet panel. The survey was conducted in 2007. Participants were asked whether they engaged in 44 specific vacation activities during their most recent vacation within Australia. Similar to the Austrian dataset, responses were binarized: a value of 1 indicates that the participant took part in the activity, while a value of 0 signifies they did not. Surveys where participants claimed they partook in more than 40 activities or no activity at all were removed as they are considered faulty.

Feature selection

At first, hierarchical clustering using Ward's method (?) and the Jaccard index was applied to the features. The resulting dendrogram is shown in Figure ???. Based on this clustering, $k = 15$ clusters were identified, and generally, only one representative feature from each cluster was selected for further analysis. Clusters containing unpopular activities, such as "Adventure", which only had 42 participants, were discarded. The cluster containing popular

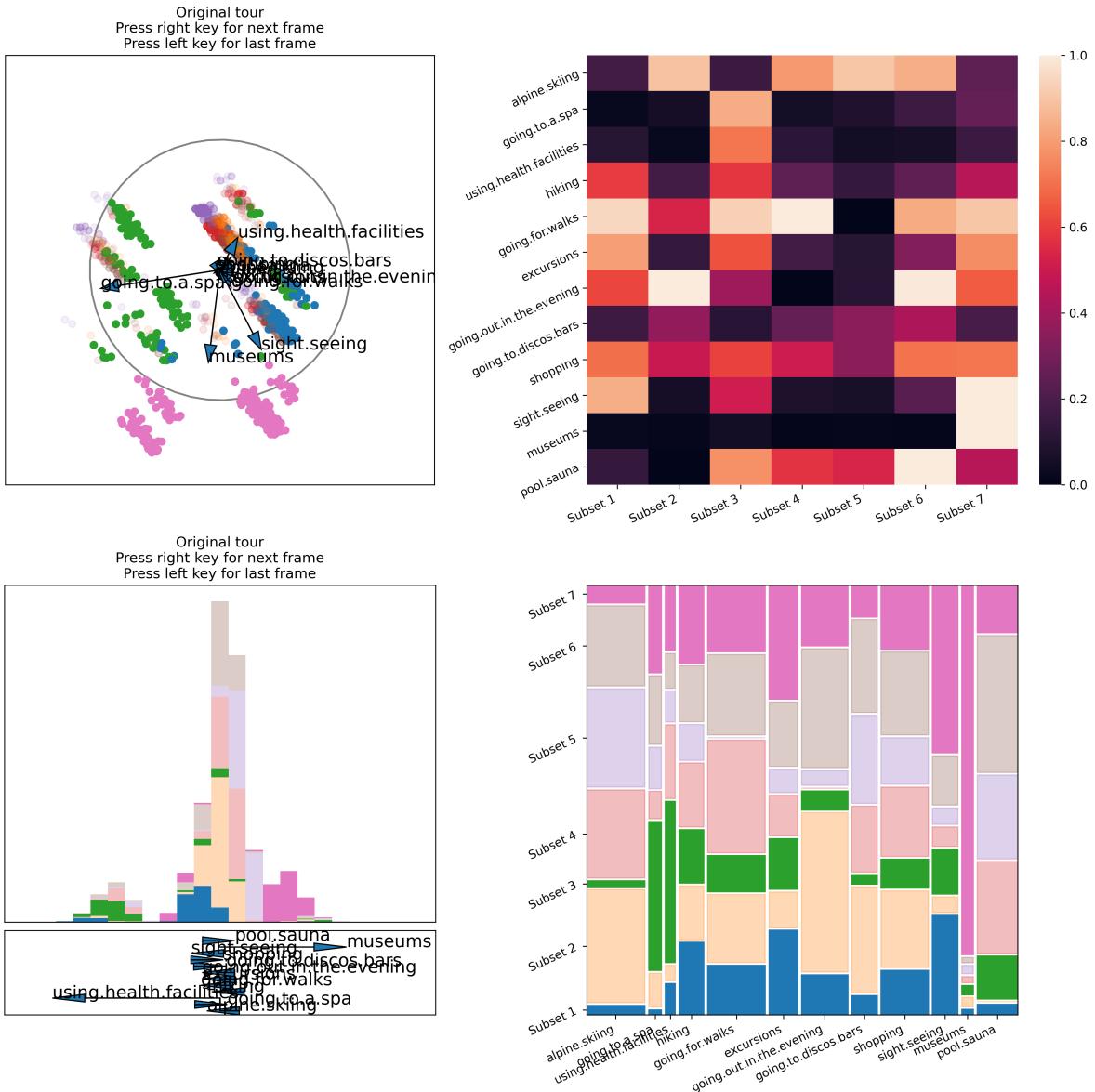


Figure 9: Top left: 2D tour with the linear discriminant analysis projection pursuit index. Top right: heatmap with the intra-cluster fraction. Bottom left: 1D tour with the linear discriminant analysis projection pursuit index. Bottom right: mosaic plot.

features like "Beach," "Swimming," "ScenicWalks," "Markets," "Sightseeing," "Friends," "Pubs," "BBQ," "Shopping," "Eating," "EatingHigh," "Movies," and "Relaxing" was treated differently. Multiple features from this cluster were retained to preserve as much information as possible. After feature selection, the observations were reclustered using k-means with $k = 15$.

Segmentation of solo travellers

In the heatmap displaying the intra-cluster fraction shown in Figure 11, we observe that subsets 1, 2, and 6 tend to prefer travelling alone. As a tourist agency, we might be interested in targeting solo travellers more effectively. However, subsets 2 and 6 also include individuals who enjoy spending time with their friends. To further explore the dataset, we can launch an `interactive_tour()` with the same configuration as in the previous example. To achieve better separation between the subsets, we can skip to the last frames of a 2 dimensional tour optimized for the LDA index.

Since we are particularly interested in tourists who did not spend time with friends, we can extend the projection axis "Friends" outward to separate the data based on that feature. Dat-

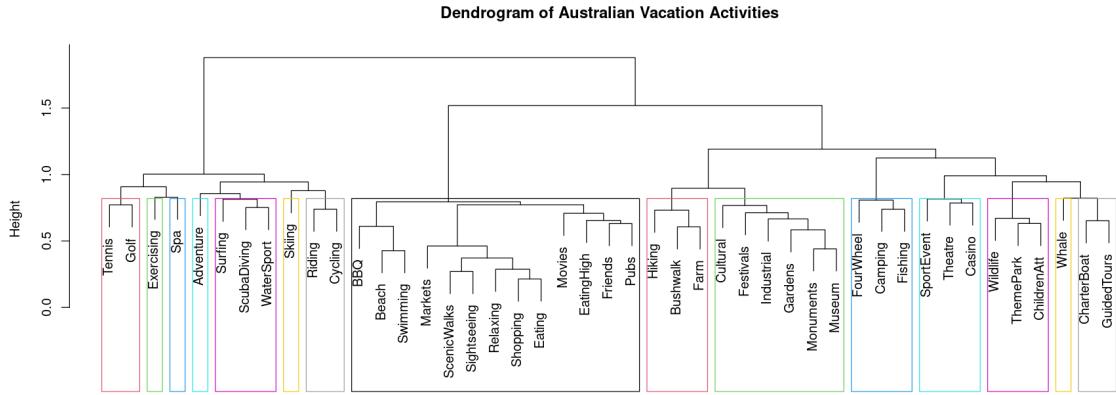


Figure 10: Dendrogram of the features of the Australian Vacation Activities dataset using Ward's method with the Jaccard index.

apoints in the opposite direction of the "Friends" axes are the solo travellers we are interested in, which can be seen in Figure 12 on the left side of the top left plot. Subsequently, we can pull all other projection axes in one direction to separate data points based on their overall activity level. Tourists that fall in the direction of the axes generally engaged in more activities compared to those in the opposite direction of the projection axes. This separation is evident, as observations in subset 6 are located opposite the axes, and the heatmap shows that they did not engage in many activities. Similarly we can also see that subset 2, which contains quite active tourists, is shifted towards the direction of the axes.

Using this logic, we can segment the data points on the left into three segments: active tourists (more upward), medium active tourists (center), and largely inactive tourists (bottom). The result of this segmentation can be seen in Figure 13. Analyzing the heatmap, we can identify several interesting patterns.

By comparing subset 2, which contains active tourists who spent time with their friends, with subset 7, the active tourists who traveled alone, we notice that subset 7 showed less interest in visiting the casino, theatre, and chartering a boat.

We also observe distinctive differences in the interests of the subsets that traveled alone. Subset 7 was interested in relaxing, shopping, sightseeing, wildlife, going to the beach, visiting pubs, and exploring farms. Given that subset 7 showed a notable interest in going to pubs, we can assume they want to meet new people. This insight can be utilized in a marketing campaign that bundles the activities subset 7 is interested in, creating a package tailored for solo travelers. This way, they can meet other solo travelers while engaging in activities they enjoy.

Although subset 8 was generally less active, almost everyone still engaged in sightseeing. For them, the focus was on sightseeing, relaxing, shopping, and going to the beach, with much less interest in other activities. It can be assumed that they value their time alone. These tourists could potentially be targeted more effectively by offering sightseeing options with minimal interaction, such as using a phone app to provide information about interesting locations, rather than relying on a tour guide.

Finally, since subset 9 is notably inactive, one might infer that they prefer spending much of their time in their accommodation. Consequently, these individuals might be most interested in accommodations that offer well-equipped, comfortable living spaces with amenities that cater to relaxation and leisure.

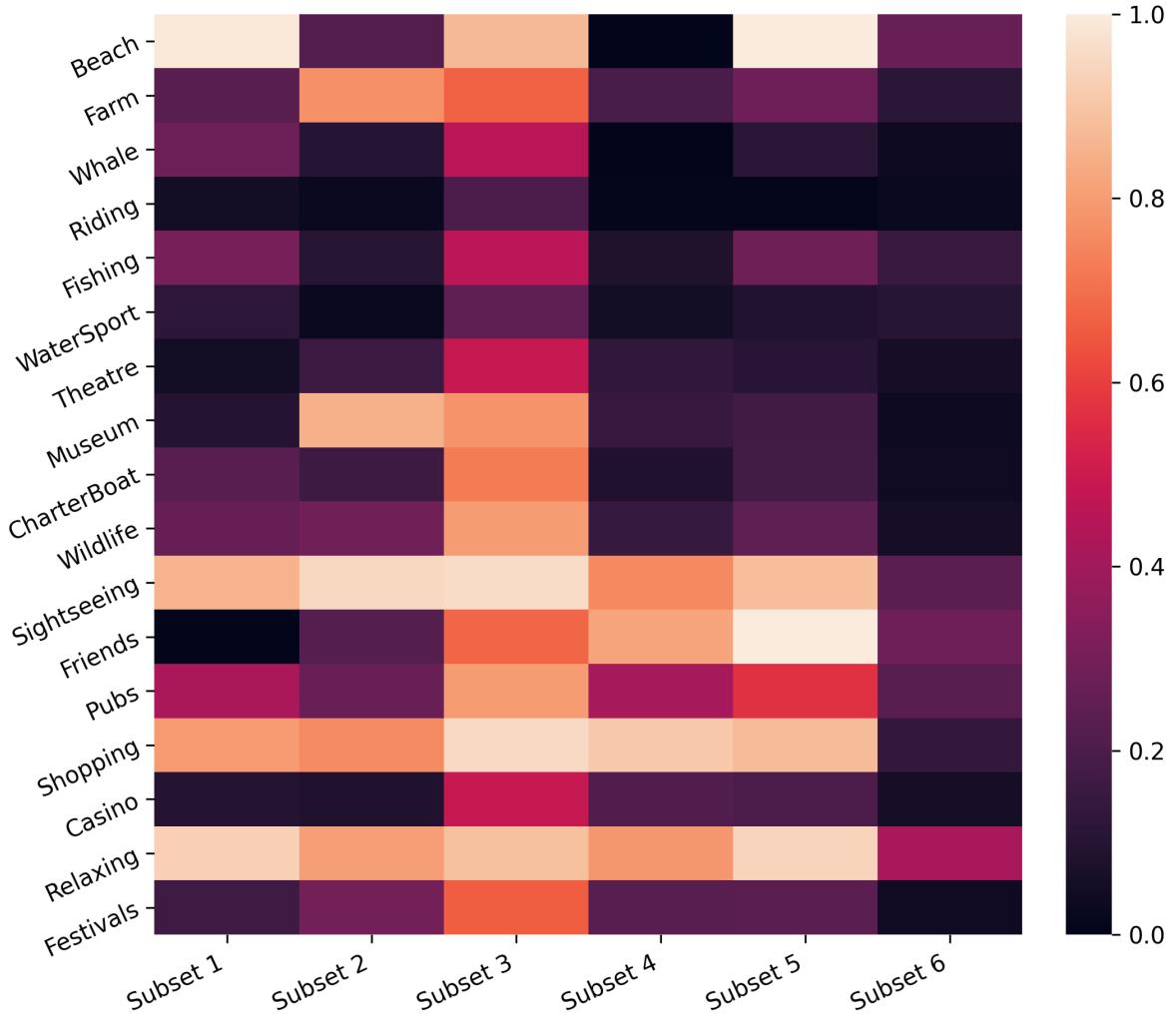


Figure 11: Heatmap of the intra cluster fraction of the k-means solution with $k = 6$ and the reduced feature subset.

5. Discussion

One might question the necessity of manual exploratory data analysis, considering it is inherently subjective and relies heavily on intuition. However, in situations like those presented here, where there is no clearly defined or optimal clustering solution or feature selection, manual exploration becomes indispensable. While it is possible to optimize clustering metrics to improve separation between clusters, this alone may not yield conclusions that are useful for practical applications. The lack of clear boundaries and the overlapping nature of clusters in the datasets underscore the limitations of purely automated methods in capturing the complexity and nuance of real-world data.

In such cases, manual exploration allows analysts to interweave expert knowledge, intuition, and specific objectives with the initial clustering solution, which serves as the backbone of the analysis. This approach is particularly valuable when no single solution can be deemed “correct.” For instance, in the Austrian dataset, leveraging the interactive GUI for feature selection enabled us to isolate the most informative activities and gain deeper insights into the preferences of tourists who might be interested in visiting museums. This understanding facilitated targeted recommendations for increasing museum attendance by focusing on tourists frequenting hiking trails, excursion spots, and shopping centers. Such insights are challenging to extract through automated optimization alone.

Similarly, in the Australian dataset, manual exploration allowed us to refine the understanding

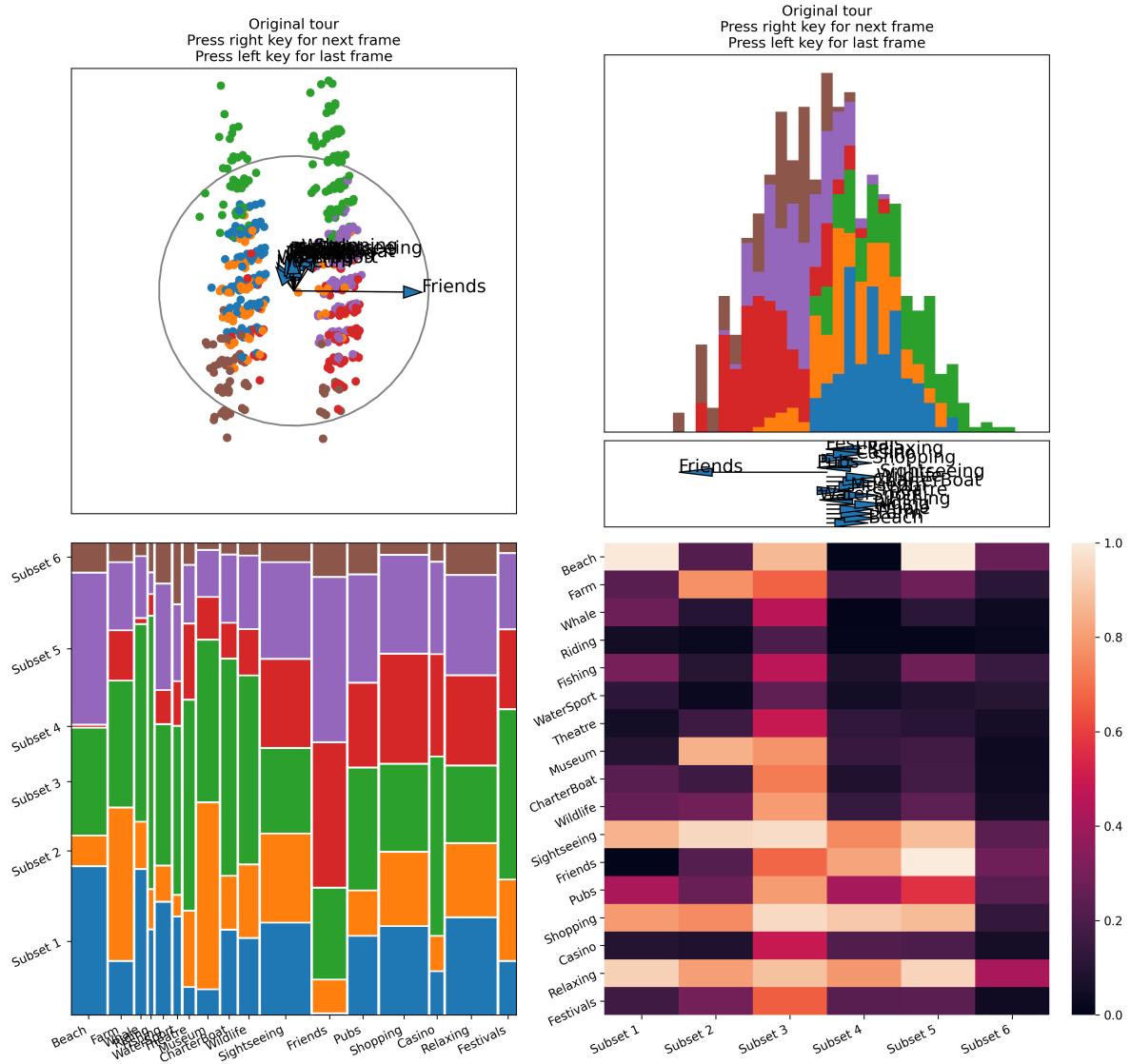


Figure 12: Separation of the data based on the feature "Friends" and the general activity. Top left: 2D tour. Top right: 1D tour. Bottom left: Mosaic plot. Bottom right: Heatmap with the intra cluster fraction.

of solo travelers by dividing them into three distinct subsegments: highly active, moderately active, and largely inactive tourists. This nuanced segmentation, derived from a blend of automated clustering and manual adjustments, offers a deeper understanding of the varied needs and behaviors within this group, enabling more effective marketing strategies.

While some plots supported by the pytourr package are specifically designed for analyzing binary survey data, it is important to emphasize that its capabilities extend far beyond this data type. The versatility of tours has been demonstrated across various applications in the past, showcasing their effectiveness in exploring complex, high-dimensional datasets. The interactive GUI enables seamless exploration of both one- and two-dimensional tours, regardless of the data being analyzed, providing a powerful tool for uncovering patterns and insights across diverse domains.

6. Conclusion

Ultimately, manual data exploration serves as a crucial complement to automated methods,

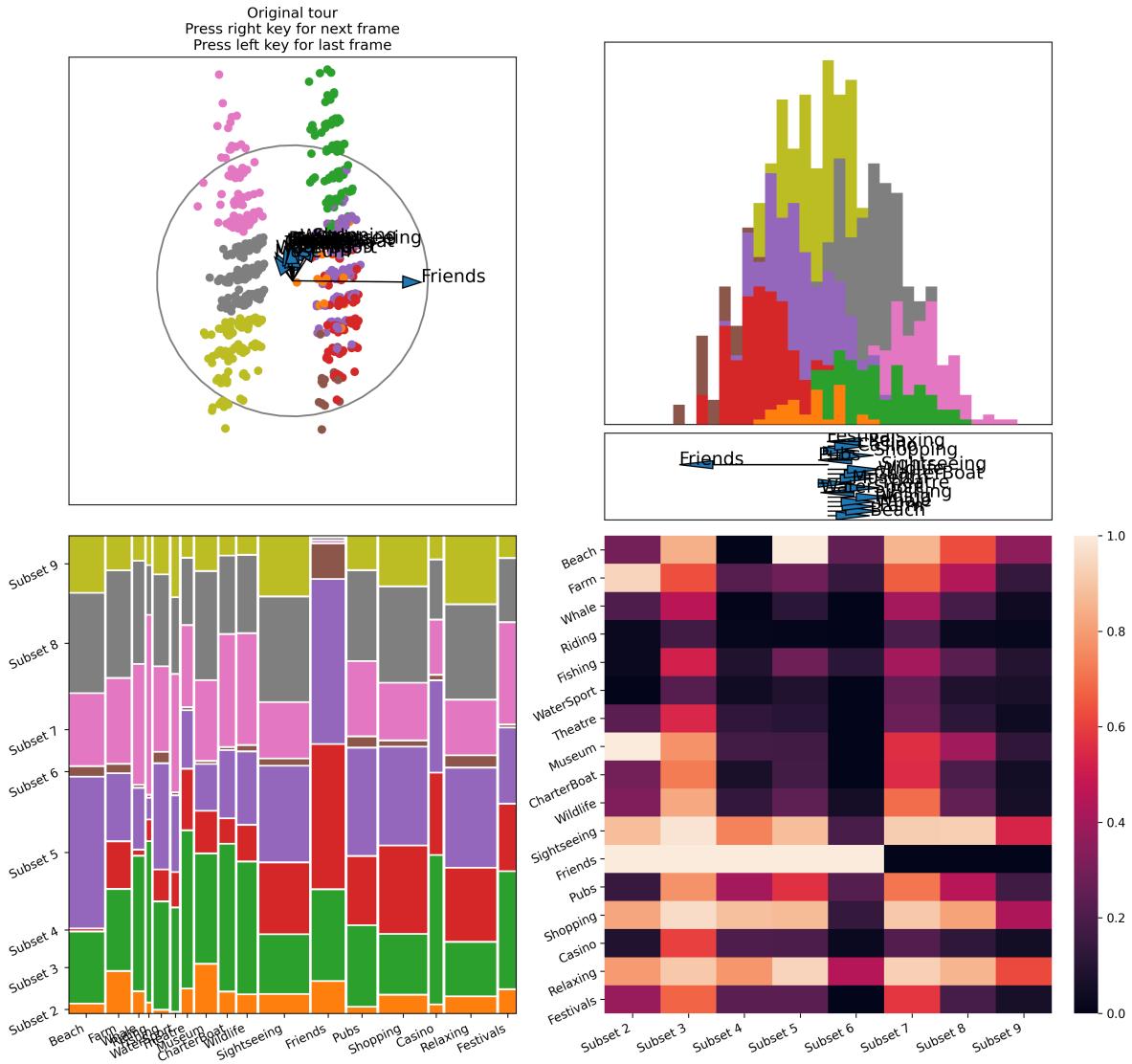


Figure 13: Separation of the data based on the feature "Friends" and the general activity after reselection of subsets. Tourists in subsets 7 (pink), 8 (grey) and 9 (gold) all responded they didn't engage in activities with their friends. Subset 7 was the most active, subset 8 medium active and subset 9 largely inactive. Top left: 2D tour. Top right: 1D tour. Bottom left: Mosaic plot. Bottom right: Heatmap with the intra cluster fraction.

providing the flexibility to incorporate context, expert judgment, and specific analytical goals. This approach enables analysts to refine initial results and adapt them to the complexities of real-world scenarios, leading to more nuanced interpretations and actionable insights. The pytourr package offers a user-friendly and powerful tool for conducting such analyses, bridging the gap between automated clustering and interactive exploration.

By integrating interactive visualization capabilities, the pytourr package empowers users to dynamically engage with their data, making it possible to uncover subtle patterns and relationships that might otherwise remain hidden. This flexibility is particularly valuable in complex datasets where traditional methods may fall short. Additionally, the package's broad applicability across various data types and analytical contexts highlights its potential as a versatile tool for a wide range of research and practical applications.

Looking forward, expanding the interactive features and supporting additional visualization and data types could further enhance the package's utility, making it an even more power-

ful tool for exploratory data analysis. Future developments could also include integrating advanced machine learning techniques, such as enabling users to generate new cluster solutions directly within the GUI. Another promising direction would be the implementation of sliced tours, which would enhance the package’s capability to analyze concave and nonlinear structures.

In summary, *pytourr* represents a significant advancement in the toolkit of data analysts, offering a novel way to balance automated analysis with human intuition and domain expertise, thereby facilitating a deeper and more comprehensive understanding of complex datasets.

References

- Chang W, Cheng J, Allaire J, Sievert C, Schloerke B, Xie Y, Allen J, McPherson J, Dipert A, Borges B (2024). *shiny: Web Application Framework for R*. R package version 1.9.1.9000, <https://github.com/rstudio/shiny>, URL <https://shiny.posit.co/>.
- Cliff K (2009). “A formative index of segment attractiveness: optimising segment selection for tourism destinations.”
- Dolnicar S, Leisch F (2003). “Winter tourist segments in Austria: Identifying stable vacation styles using bagged clustering techniques.” *Journal of Travel Research*, **41**(3), 281–292.
- Hart C, Wang E (2022). *detourr: Portable and Performant Tour Animations*. R package version 0.1.0, URL <https://casperhart.github.io/detourr/>.
- Hunter JD (2007). “Matplotlib: A 2D graphics environment.” *Computing in Science & Engineering*, **9**(3), 90–95. doi:[10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- Inc PT (2015). “Collaborative data science.” URL <https://plot.ly>.
- Laa U CD (2024). “galahr.” <https://github.com/uschiLaa/galahr>.
- Laa U, Aumann A, Cook D, Valencia G (2023). “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*, **32**(3), 1229–1236.
- Leisch F (2006). “A Toolbox for K-Centroids Cluster Analysis.” *Computational Statistics and Data Analysis*, **51**(2), 526–544. doi:[10.1016/j.csda.2005.10.006](https://doi.org/10.1016/j.csda.2005.10.006).
- Leisch F, Dolnicar S, Grün B (2018). *Market segmentation analysis: Understanding it, doing it, and making it useful*.
- Lundh F (1999). “An introduction to tkinter.” URL: www.pythonware.com/library/tkinter/introduction/index.htm.
- Schimansky T (2024). “CustomTkinter.” <https://github.com/TomSchimansky/CustomTkinter>.
- Ushey K, Allaire J, Tang Y (2024). *reticulate: Interface to 'Python'*. R package version 1.38.0, <https://github.com/rstudio/reticulate>, URL <https://rstudio.github.io/reticulate/>.
- Wickham H, Cook D, Hofmann H, Buja A (2011). “tourr: An R Package for Exploring Multivariate Data with Projections.” *Journal of Statistical Software*, **40**(2), 1–18. URL <https://doi.org/10.18637/jss.v040.i02>.

Affiliation:

Matthias Medl
Institute of Statistics
BOKU University Vienna
E-mail: matthias.medl@boku.ac.at