animbook: Visualizing changes in performance measures and demographic affiliations using animation

by Krisanat Anukarnsakulchularp and Dianne Cook

Abstract Performance is often reported in quantiles or ranks and often repeatedly provided over time. Between elections, voters often switch party preferences, which produces categorical values over time. Inspired by the animation in the New York Times article "Extensive data show punishing reach of racism for black boys," we provide new visualization functions to generate animations that display this type of data in an R package called **animbook**. Functions to pre-process the data and prepare it for the animation are provided. The methods will be suitable generally for data where categorical variables are measured over time, and provide a way to engagingly view changes.

1 Introduction

The concept of "zombie companies" began to attract attention when Caballero, Hoshi, and Kashyap (2008) reported on their proliferation in Japan. Zombie companies are those with an interest coverage ratio of less than one for a period of more than three years, that is, companies taking space in the market but adding no life to the economy. We would detect their presence by a drop in performance, which is a movement pattern in their relative ranking over time. Generally, studying movement patterns is interesting for many problems, including rocket ship start-ups that rapidly perform well or in politics, to study voters who switch party affiliation between elections. Viewing changes between categories over time is an interesting challenge for visualization.

The New York Times provided a possible solution in the article titled "Extensive data show punishing reach of racism for black boys" (Badger et al. (2018)) to tell the story of how racism appears to inhibit socioeconomic change. This animation is the motivation for the new visualization presented here to be applied generally.

The challenge in producing an animation like that in the New York Times article animation is the transformation of the data and connection with elements of the plot that will be animated. The complexities include allowing the user to choose the number of categories, standardizing distributions, and allowing the user to input pre-computed categorical data. These considerations provide the objective for creating an R package that can generalize the animation to suitably apply to a wide range of data.

The structure of this paper is as follows. The next section explains the animation in The New York Times article and why it is relevant to the problem of studying zombie companies. The next section describes the expected data format. Following this is an explanation of available animation tools and how they are employed for this problem. A section on the functions of the package and how the visualization is designed demonstrates how the data is mapped to the animation. The last two sections illustrate the usage of the package and applications to company performance and changing political allegiance.

2 Explanation of the New York Times visualization

The interactive chart featured in the New York Times article (Figure 1) unveils the issue of income disparities between black and white children who were raised in families with comparable income according to the Chetty et al. (2020). This visualization reveals that, compared to white children, black children are more likely to drop down to the lower-income group, given that they both grew up in wealthy families.

In the visualization, each observation is initially classified into one group at the start and potentially transitions into either the same group or a different group. This dynamics visualization constructs questions on the broader use of this visualization to other types of data. The potential use of this visualization on accounting data is to convey a message, as reported by the McGowan, Andrews, and Millot (2017), that the concept of zombie companies is not unique to Japan alone. It is also present in the United States, which has a faster metabolize rate (more new listings and exits) relative to Japan.

The political data that exhibits the movement of voters switching party affiliations between elections can be a valuable insight into the behavior of the voters. This data could be extended to

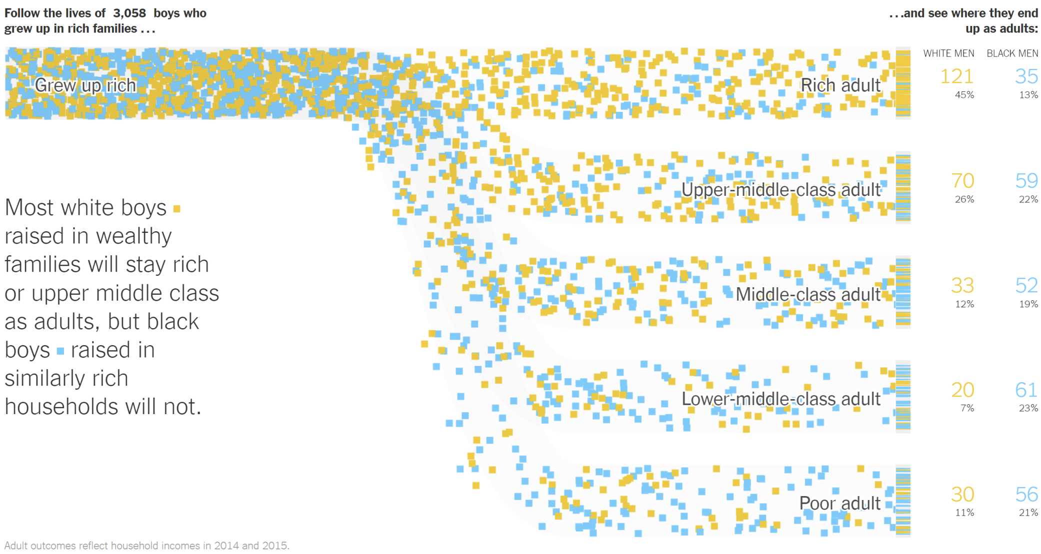


Figure 1: Screenshot of the New York Times animation, which is the motivation for this visualization package.

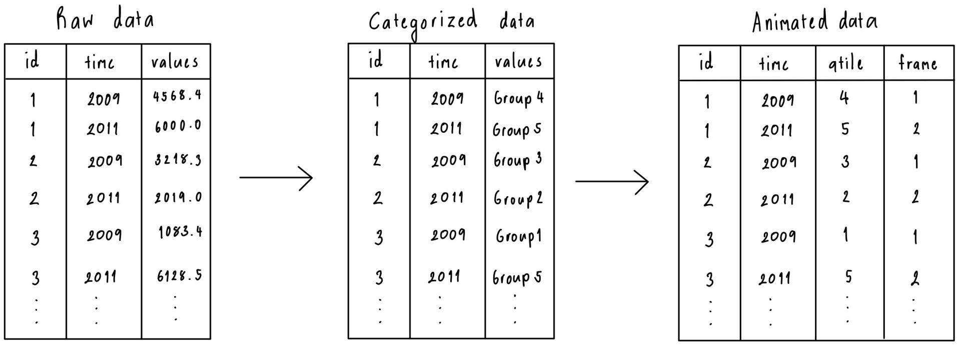


Figure 2: The animation expects data with an ID and a time variable, along with a numerical variable (raw form), which is possibly converted to categorical (categorized). The data can be provided in the raw or categorized form and will be processed into the format needed for the animation, where the categorical variable is treated as a quantile and an animation frame variable is created.

incorporate demographic information about the voters, providing analysts with a significant insight into voter behavior. This allows them to consolidate effective campaigns for their political party. This also applies to marketing data, where customers shift their product interest to the competitor, providing the marketing analysts with an understanding of both the company's products and the overall market.

This animation was developed using the software based on JavaScript, D3.js (Bostock (2012)), and WebGL (Mozilla Foundation (2011)). The D3 JavaScript is one of the most widely known libraries for creating an interactive and dynamic visualization. It enables the designers to bind both the data and graphical elements to the DOM (Document Object Model). On the other hand, WebGL functions as a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. For the animation in this paper, the programming language that will be used for recreating and revising the visualization done by The New York Times articles is R (R Core Team (2021)).

3 Data

In the data structure, there are requirements that must be followed for reproducing the animation. First, the data set needs to be in the `tidy data` format (Wickham (2014)). The data then must have at least ID and time variables, in addition to the measured variables, which would usually be numeric

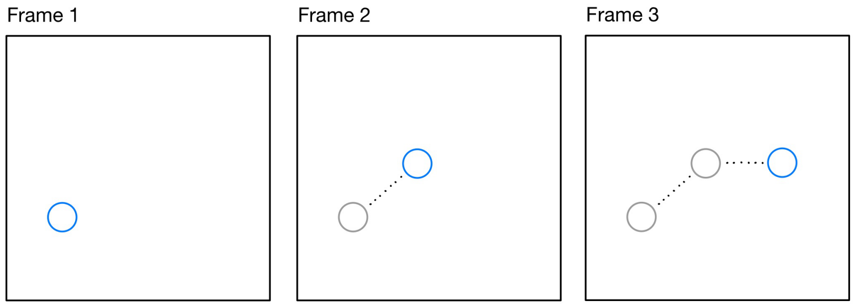


Figure 3: The diagram shows how the animation is done using the successive pictures. When small changes are seen in quick succession, it will appear as if the objects are in motion.

but can also be categorical as well. The ID variable indicates the individual, which is followed over time, such as the company name. There may also be a grouping variable, such as the country where the company is registered.

Figure 2 illustrates the expected format of the data and variables created to prepare it for the animation. We start with the raw data structure. The values are presented in the numerical format, which we call the 'raw' form. In most cases, the measured variable will be numerical and require transformation. The second form is categorized data, which involves transforming numerical variables into categories, typically quantiles. This transformation may not be necessary if it is already provided in the categorized format. The last form is animated data, where the frame is assigned to each unique ID.

4 Animation tools

A principle important for designing a useful animation is called persistence of vision (Webster (2005)). When an image disappears, the brain will retain the previous images for a brief period of time. It is this slight period of retention that allows humans to separate sequential images. If this is seen in quick succession, it will appear as if the objects are in motion. This is illustrated in Figure 3.

There are multiple ways to create an animation in the R environment (R Core Team (2021)), including the packages **gganimate** (Pedersen and Robinson (2020)) and **plotly** (Sievert (2020)).

The **gganimate** package is an extension from the **ggplot2** package (Wickham (2016)) to include the description of an animation. It added new grammar classes to the plot object, allowing it to understand how the plot should change over time. The use of `transition_*()` functions allows it to achieve this by specifying how the data evolves and how it relates to itself across time. This includes `gifski_renderer()` from the **gifski** package (Ooms (2023b)) to save animation in GIF format or `av_renderer()` from the **av** package (Ooms (2023a)) to save it into a video file format.

The `plotly` software is a graphic library that provides tools for creating an interactive plot in multiple programming languages, such as R, JavaScript, Python, and Julia. In R, plotly can be accessed through the **plotly** package, which integrates plotly.js from the JavaScript graphing library. The usage of this library can be from a converting function, `ggplotly()`, or a standalone function, `plot_ly()`. The conversion is accomplished by taking the elements from the `ggplot` object and then redrawing them using the plotly.js.

In the context of data, as shown in Figure 4, observations are positioned at specific points in time. The further the distance between these points, the less smooth the animation becomes. This issue can be eliminated by interpolating additional points in between the observations. In the **gganimate** package, the interpolation is achieved using the **tweenr** package (Pedersen (2022)), while in **plotly**, it utilizes d3.interpolate (Bostock (2012)).

Figure 4 demonstrates how the frame variables are applied in an animated plot. The frame variable within the animated data structure allows the animation function to determine the position of observations on the plot at any given frame.

From Hasler, Kersten, and Sweller (2007), it suggests that having control options for the animation can improve the efficiency of the learning process. Additionally, the length and speed of the animation should also be taken into consideration. According to R. Mayer (2010), the working memory, responsible for selecting and processing information from sensory memory, only holds a processed version of

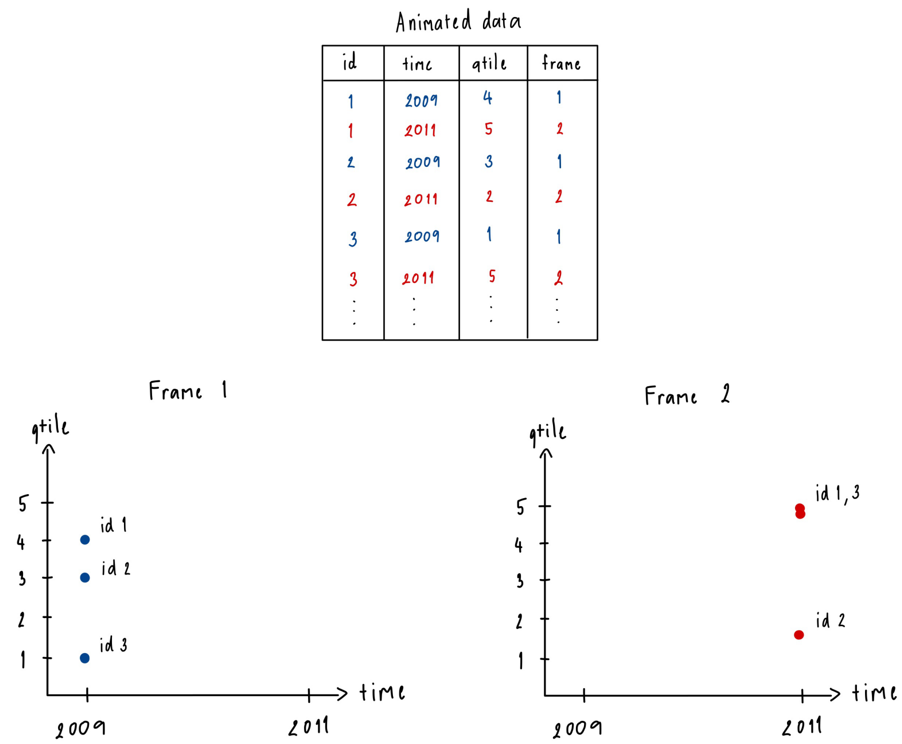


Figure 4: The diagram shows how the frames were used in the animated plot. Frame one is depicted in blue, and frame two is represented in red. Each blue component is mapped exclusively to the Frame 1 plot, while all the second-frame elements are excluded, and vice versa.

what was presented for generally less than thirty seconds.

In **gganimate**, the issue of integrating controls can be addressed by setting the `renderer` argument to be `av_renderer()`, which allows the animation output to be in media applications provided in their systems. As for adjusting the length and speed of the animation, the `nframes` and `fps` arguments can be utilized. The `nframes` dictates the number of frames to be rendered, while `fps` controls how many frames are displayed in one second. Using these two parameters, the duration of the animation in seconds can be calculated as follows: length = nframes/fps.

In the case of **plotly**, control integration is already implemented by default. The `frame` and `transition` arguments within the `animation_opts()` function can be specified to set the length and speed of the animation.

5 Visualization design

The animated visualization can be an effective communication tool (R. E. Mayer and Moreno (2002); Robertson et al. (2008)). It helps with communicating changing data values, enhancing the narrative, and keeping it engaging for the audience. According to R. E. Mayer and Moreno (2002), animation can improve learning, especially when the goal is to promote deep understanding.

R. E. Mayer (2005) states that designing multimedia requires the designer to understand how people learn. One of the principles in R. E. Mayer (2005), Redundancy, suggests that a piece of excess information could overload the learners. By this principle, the animation must be carefully designed to avoid this pitfall.

In the animation, the proportional shaded area has been incorporated to facilitate a quick grasp of the proportion information. It displays the proportion of observation within each group. The design also needs to account for situations where the visualization lacks an adequate number of data points. In such cases, it can be challenging to visually discern the movement pattern of the subgroup, requiring the implementation of observation interpolation.

The paths of observations need to be interpolated to create small changes in the position of points to produce the appearance of motion, as mentioned in the Animation tools section. These

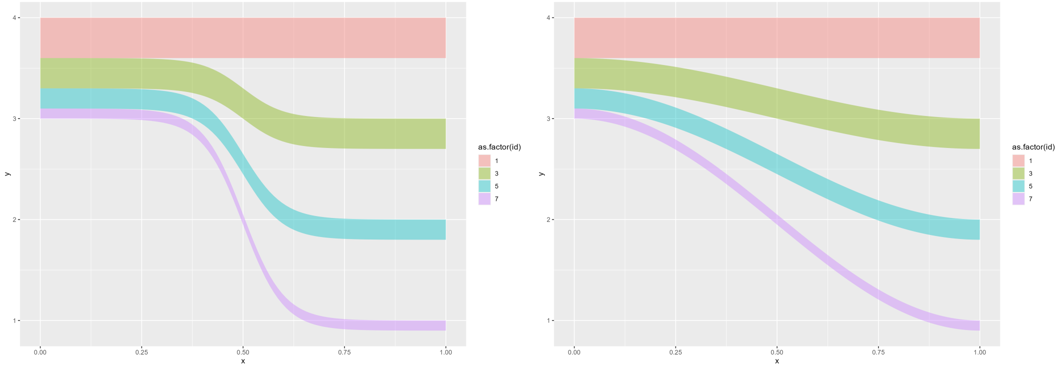


Figure 5: The plot shows the difference between the sigmoid (shown in the left figure) and the sine curve (shown in the right figure). In the sigmoid, as the curve progresses, it becomes narrower, resulting in a less accurate representation of proportions compared to the sine curve.

interpolate functions in **gganimate** and **plotly** only generate a linear path between points. However, linear paths are visually less appealing than non-linear paths. Sigmoid curves are commonly used in Sankey diagrams (Hvitfeldt (2018)). However, Shaffer (2019) argues that a sine curve more accurately represents proportion throughout the path. This is illustrated in Figure 5. The sigmoid accurately represents the proportion at the beginning and end, but as it curves, the shape gets narrower, leading to a less accurate proportion representation than the sine curve.

6 Software

Installation

The development version of **animbook** can be installed from `https://github.com/KrisanatA/animbook` with:

```
# install.packages("devtools")
devtools::install_github("KrisanatA/animbook")
```

Overview of functions

In designing the **animbook** package, a three-step structured approach was developed to create an animation. The initial step is to reformat the data into a categorized data structure, as seen in Figure 2. The second stage is creating a `ggplot` object, which can be subsequently passed into the animation function. During the second stage, an internal function will turn the categorized data into an animated data structure. The final step involves transforming the `ggplot` to a `gganimate` object by integrating the animation settings. This three-step structure was implemented to ensure that users, regardless of their level of experience, can produce the animations with simplicity while retaining customization for more experienced users.

Data preprocessing

From Figure 2, there is a need to map numerical value to a category. One way to handle this is by ranking the sales and grouping the rankings into quantiles. In some cases, this may not be the best option. When the observation is moved up by quantile, one is bound to move down. This issue can be resolved by using an alternative method, which is grouping values based on their absolute values. Users may also be interested in grouping the data based on different groups, for example, ranking within a specific country. This generalization leads to four different scaling methods for the numerical data.

```
#> # A tibble: 12 x 8
#>    id     time gp    values  rank rank_group absolute absolute_group
#>    <fct> <int> <fct>  <dbl> <int>      <int>    <int>          <int>
#>  1 1      2020 X      4255.     3          3        3              3
#>  2 1      2023 X      3357.     2          2        4              4
```

```
#>  3 14     2020 X      6763.    2           2          2              2
#>  4 14     2023 X      1197.    4           5          5              5
#>  5 100    2020 X      6864.    2           2          2              2
#>  6 100    2023 X      2321.    3           3          4              4
#>  7 21     2020 Y       698.    5           5          5              5
#>  8 21     2023 Y      3970.    2           2          4              3
#>  9 106    2020 Y      4110.    3           3          3              3
#> 10 106    2023 Y      2866.    3           3          4              4
#> 11 148    2020 Y      7174.    2           2          2              2
#> 12 148    2023 Y      4217.    1           1          3              3
```

1. Ranking by year. (rank)
2. Ranking by year within a group. (rank_group)
3. Fix bins relative to absolute values by year. (absolute)
4. Fix bins relative to absolute values by year within a group. (absolute_group)

For the first and second scaling methods, it is necessary to rank the values based on time, and in cases where a group is provided, they are ranked based on both time and groups. To ensure that the rank scales among different groups are the same, the variables are first normalized to a range between 0 and 1. The third and fourth scaling methods also involve a normalization step, but they are based on raw values instead of rank values.

All of these scaling methods utilized the `cut()` function from the base R package (R Core Team (2021)) to split the values into quantiles. The `cut()` function requires the specification of the `breaks` argument. If it is not provided, the prep function in this package defaults to using the `seq()` function, which sets the minimum and maximum values to 0 and 1, respectively, and then increments by equal steps based on the number of groups of interest. Additionally, the users have the option to specify the breaks themselves if they choose to do so, noting that the breaks provided need to be between 0 and 1, and the length of the vector needs to be the number of groups plus 1.

All of the pre-processing steps mentioned above are completed using the `anim_prep()` or `anim_prep_cat()` function, depending on the stages of the data structure. The `anim_prep()` function is used for raw data format, while the `anim_prep_cat()` function is for categorized data format. There are additional arguments that allow users for more customization.

These are only the initial steps in formatting the data into a category. Now that there is a method to transform the data from the raw into a categorized format, the next step is to modify it into an animated data structure. It is carried out by assigning the frame to each individual observation, ensuring that each ID does not contain repeat frame values. It lets the **gganimate** or **plotly** perceive where the observation would be on the plot at a given frame, as seen in Figure 4.

The frame variable is assigned by sorting the data based on the ID and time using the `arrange()` function, followed by applying the `group_by()` function on the ID, allowing the `row_number()` function to be performed within each group. The functions mentioned in this paragraph are from the **dplyr** package (Wickham et al. (2023)). These are done by the internal function for each of the plots provided in the packages.

The argument for the `anim_prep()` and `anim_prep_cat()` function are listed in the Table 1 and 2 respectively. The `data`, `id`, `values`, and `time` arguments are required for both functions.

The `ncat`, `breaks`, `group_scaling`, and `scaling` arguments are for the user to customize the data scaling calculation in the `anim_prep()` function. In the `anim_prep_cat` function, the users can customize the order of the category to be shown on the y-axis using the `order` argument.

Then, for further customization regarding how the final visualization looks. The `group`, `label`, and `time_dependent` arguments can be adjusted.

Plotting function

Once the data is prepared. The next step is to create the `ggplot` object as a basis for the animation. There are three plots available in this package. Two of the plots could be used for the animation, and another plot is used as a static visualization. All of the plots have an internal function that converts the categorized data format into the animated structure for each plotting function.

- `kangaroo_plot()`: plots the observation's movement over time.
- `wallaby_plot()`: the subset plot of the `kangaroo_plot` with the time limit to only start and end.
- `funnel_web_plot()`: the faceted static plot by time variable.

The main focus of this paper will be on the `wallaby_plot()`, which draws inspiration from the New York Times animation and combines the knowledge gained from the previous sections in creating

Table 1: The arguments for the anim_prep function.

Argument	Description
data	The numerical data to be prepared for visualization.
id	The column name in the data that will be used to identify an individual over time. For example, company name.
values	The column name in the data that will be used for the y-axis (must be numerical).
time	The column name in the data represents the time variable (x-axis).
group	The column name in the data for distinguishes between the values group (for example, country). In the visualization, this will be used as a color argument.
ncat	The number of categories to create for scaling values. The default for this function is 5 categories.
breaks	A vector of breaks for creating bins. If this is not provided, the bins will have equal size.
label	A vector of labels to be used for the y-axis in the visualization. If this is not provided, the labels will be the position on the y-axis.
group_scaling	The column name in the data that will be used for grouping. Allow the function to isolate the calculation between the groups.
scaling	The scaling method to be used; "rank" or "absolute". The default scaling is "rank".

Table 2: The argumenst for the anim_prep_cat function.

Argument	Description
data	The numerical data to be prepared for visualization.
id	The column name in the data that will be used to identify an individual over time. For example, company name.
values	The column name in the data that will be used for the y-axis (must be numerical).
time	The column name in the data represents the time variable (x-axis).
group	The column name in the data for distinguishes between the values group (for example, country). In the visualization, this will be used as a color argument.
ncat	The number of categories to create for scaling values. The default for this function is 5 categories.
breaks	A vector of breaks for creating bins. If this is not provided, the bins will have equal size.
label	A vector of labels to be used for the y-axis in the visualization. If this is not provided, the labels will be the position on the y-axis.
group_scaling	The column name in the data that will be used for grouping. Allow the function to isolate the calculation between the groups.
scaling	The scaling method to be used; "rank" or "absolute". The default scaling is "rank".

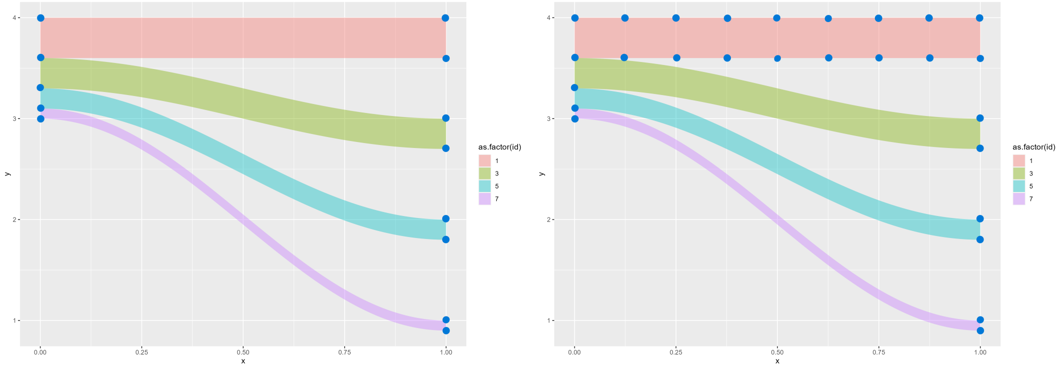


Figure 6: The plot shows how the algorithm for the proportional_shade() function works. The left figure represents the initial step of the algorithm, which calculates all the corner points. The right figure demonstrates the subsequent step, where points in-between the left and right are interpolated using the sine() function.

Table 3: The arguments for the wallaby_plot function.

Argument	Description
data	The categorized data returned from the prep function.
group_palette	The vector of the palette used by the function to supply the color to each group.
shade_palette	The vector of the palette used by the function to supply the color to the shaded area.
rendering	The choice of method used to create and display the plot, either gganimate or plotly. Default is gganimate.
subset	A character string specifying the variable used for subsetting the data. The "top" and "bottom" strings can also be used in this argument. Default is "top".
relation	The choice of relationship for the values to display on the plot, either "one_many." or "many_one". Default is "one_many".
total_point	The number of points the users want for the wallaby plot. Default is NULL, which is the number of points equal to the original.
height	The proportion of the area occupied by the observations in the shaded areas. Default is 0.6.
width	The distance between the first and last observation in the animation. Default is 50.
size	The point size. Default is 2.
alpha	The opacity of the proportional shaded areas. Default is 0.1.

the visualization.

The `wallaby_data()` function is responsible for performing data manipulation and formatting tasks on the original object. It includes creating additional data components for labeling and shading. This function also responds by interpolating the non-linear path. It performs this task by mapping the `sine()` function provided in this package to each observation using the `map()` function from the **purrr** package (Wickham and Henry (2023)). The frame is recalculated using the same method mentioned in the data preprocessing section. The `proportional_shade()` function is called to generate the proportional shaded data.

The algorithm behind the `proportional_shade()` function for generating the proportional shaded data is illustrated in Figure 6. It started by calculating the corner points for all of the shaded areas. Then, use the `sine()` function to interpolate the point between the left and right.

Now that the data are in the right format for the wallaby's plot, the `geom_point()` function is used for plotting the observations, `geom_polygon()` is used for creating the proportional shaded areas, and `geom_text()` is used for creating labels. These three functions are from the **ggplot2** package. During this process, the aesthetics mapping will be different depending on the rendering tool to be used, which can be either `gganimate` or `plotly`. The difference between the two rendering tools is that for `plotly`, the `ids` and `frame` arguments need to be specified during the creation of the `ggplot` object.

Animating function

It is necessary to save the plot as a ggplot object before passing it to the final function, anim_animate(), to animate. This function will automatically detect which rendering method was specified in the previous steps and add the minimum requirement functions accordingly. By default, if the user specifies the rendering as gganimate, then it will add the transition_time() function from the **gganimate** package. Otherwise, the animation_opts() function will be added from the **plotly** package.

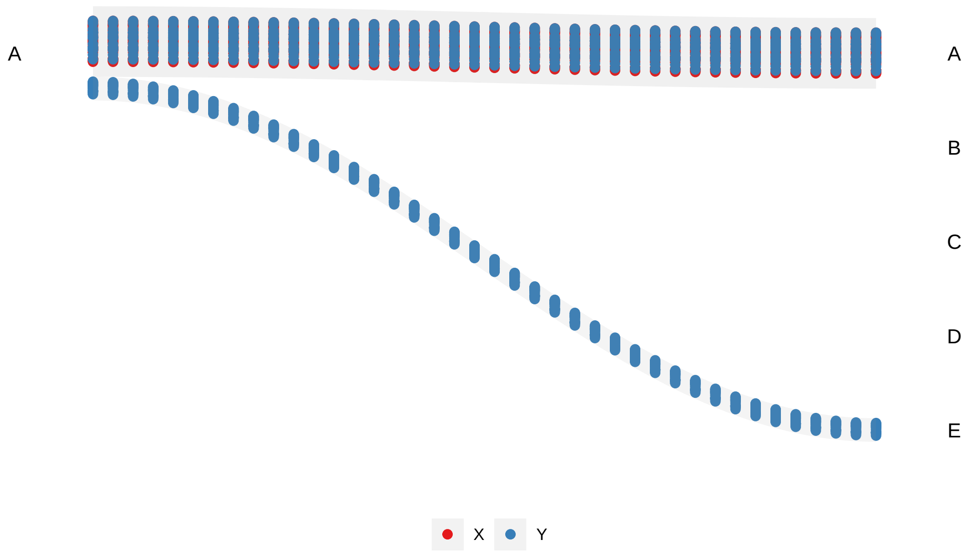
Example usage

In this section, the package usage will be demonstrated using the cat_change dataset included in this package. This dataset is simulated data that has changed from category A to E between two time points. Since this data is already in the categorized data structure, the anim_prep_cat() function is utilized. The output produced from this package is referred to as the categorized data.

```
animbook <- anim_prep_cat(cat_change,
                          id = id,
                          values = qnt,
                          time = time,
                          group = gp)

head(animbook, 10)

#> # A tibble: 10 x 5
#>    id     time qtile label group
#>    <fct> <int> <dbl> <chr> <fct>
#>  1 1      2020     5 A     X
#>  2 2      2020     5 A     X
#>  3 3      2020     5 A     X
#>  4 4      2020     5 A     X
#>  5 5      2020     5 A     X
#>  6 6      2020     5 A     X
#>  7 7      2020     5 A     X
#>  8 8      2020     5 A     X
#>  9 9      2020     5 A     X
#> 10 10     2020     5 A     X

str(animbook)

#> tibble[,5] (S3: tbl_df/tbl/data.frame/categorized)
#>  $ id   : Factor w/ 200 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
#>  $ time : int [1:400] 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
#>  $ qtile: num [1:400] 5 5 5 5 5 5 5 5 5 5 ...
#>  $ label: chr [1:400] "A" "A" "A" "A" ...
#>  $ group: Factor w/ 2 levels "X","Y": 1 1 1 1 1 1 1 1 1 1 ...
```

To obtain a ggplot object for the animation function, the wallaby_plot is used. For further customization, the shade_palette, subset, and relation arguments are applied to define the final appearance of the animation.

```
p <- wallaby_plot(data = animbook,
             group_palette = RColorBrewer::brewer.pal(9, "Set1"),
             shade_palette = c("#737373", "#969696", "#BDBDBD",
                               "#D9D9D9","#D9D9D9","#D9D9D9"),
             rendering = "gganimate",
             subset = "top",
             relation = "one_many",
             total_point = NULL)

p
```



Once the ggplot object is obtained, the anim_animate() function can be used to transform it into a gganimate object. This then can be passed onto the animate() function from the gganimate for rendering.

```
p2 <- anim_animate(p)

gganimate::animate(p2)
```

This is an example of how the users can apply the three-step process in creating an animation plot using the animbook package.

7 Application

Accounting database: osiris

The data included in the package contained a record of 790 companies from 2006 to 2018. The variables in this data are year, ID, country, sales, and japan. It is a subset of the 2021 sample data collected from the Bureau van Dijk ("Osiris" (2023)). The original data comprises 1000 companies from 1991 to 2020. It has 94 variables of information on listed and major unlisted/delisted companies worldwide.

An OECD report (McGowan, Andrews, and Millot (2017)) claims that the United States has a faster metabolic rate relative to Japanese companies, indicating that more US companies are listed and exited. The wallaby plot allows for the observation of the entry and exit behavior. It provides an insight into the number of companies involved in these actions. Additionally, the kangaroo plot animation displays a full view of the movement of the companies over time, helping to explain any behavior not depicted in the wallaby plot. It particularly looks at the movement between the beginning and end of the observation period.

The steps needed to create a visualization: 1. Filter the countries to include only the United States and Japan. 2. Prepare the data using the anim_prep function with the first scaling, which is a ranking method. 3. Use the wallaby_plot and kangaroo_plot functions to create ggplot objects and add default settings for gganimate rendering.

```
# library(animbook)
# library(dplyr)

data <- osiris |>
  filter(country %in% c("US", "JP"))

label <- c("Top 25%", "25-50", "50-75", "75-100", "Not listed")

accounting <- anim_prep(data,
```

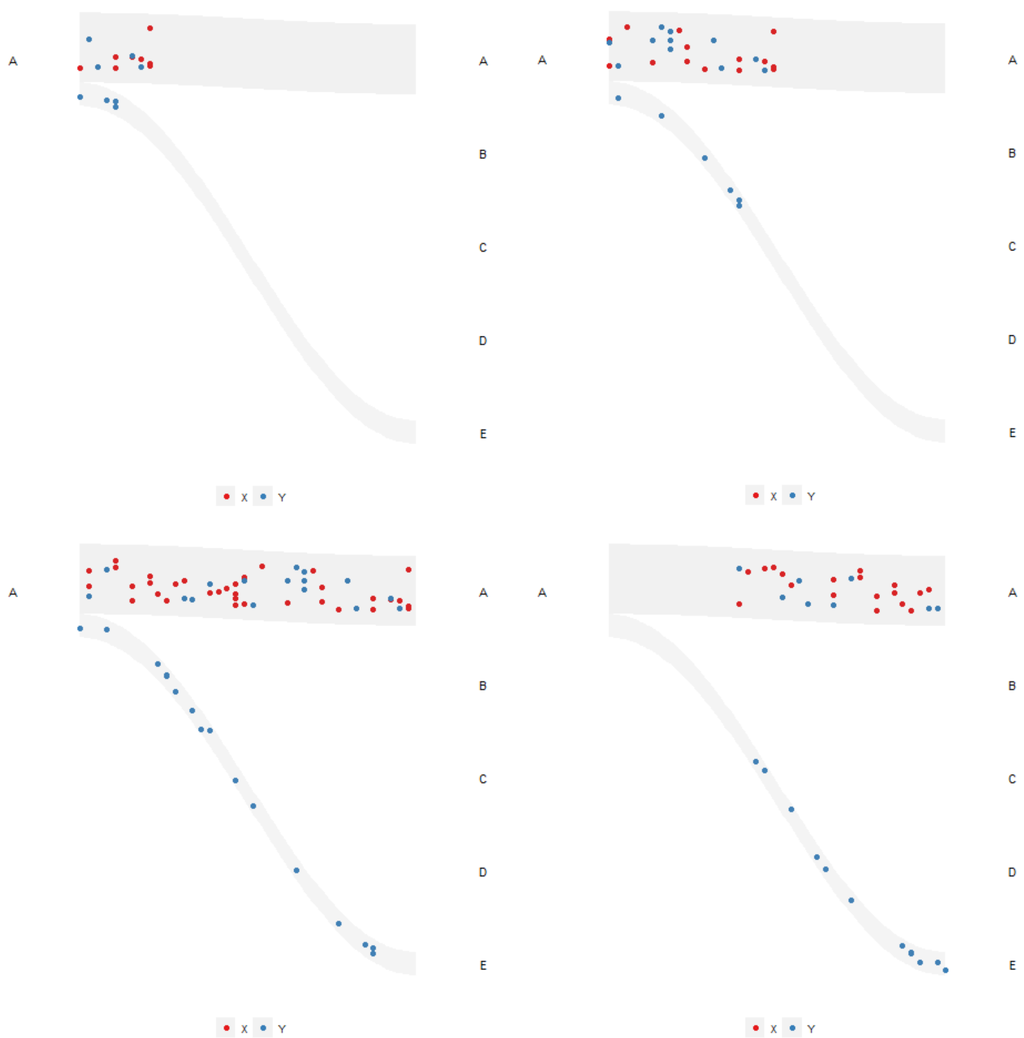


Figure 7: Animate visualization using example data. All of the X observations stay within the same group, while Y observations change from group A to group E.

```
                      id = ID,
                      values = sales,
                      time = year,
                      label = label,
                      ncat = 4,
                      group = country)

kan_p <- kangaroo_plot(accounting)

p <- wallaby_plot(accounting,
                  group_palette = RColorBrewer::brewer.pal(9, "Set1"),
                  shade_palette = c("#737373", "#969696", "#BDBDBD",
                                    "#D9D9D9","#D9D9D9","#D9D9D9"),
                  subset = "bottom",
                  relation = "many_one",
                  height = 1,
                  size = 2,
                  width = 100,
                  total_point = 1000)

kan_p2 <- anim_animate(kan_p)

p2 <- anim_animate(p)

gganimate::animate(kan_p2)

gganimate::animate(p2)
```

From the kangaroo plot (Figure 8), most companies are staying in the same quantile group, with only a small number moving upward or downward. Most of the movement is made by American companies, supporting the OECD report (McGowan, Andrews, and Millot (2017)) that the United States has a higher turnover rate compared to Japan.

Using the wallaby plot (Figure 9), it emphasize the high turnover rate from the kangaroo plot. There are clearly more American companies de-listed compared to Japanese companies, and it can be seen that there are no Japanese companies in the Top 25% in 2006 that were de-listed.

Voter behavior

The election survey dataset included in the **animbook** package focuses on the voter behavior of the 734 individuals who participated in the 2019 survey. The variables in this data are id, year, party, and gender. The ID variable has been de-identified to ensure the privacy of the responders, preventing the identification of the survey participants. The year column is derived from the two different questions in the survey, which answered how the top party performs in keeping the old voters of different genders relative to the 2016 Australian election results. This dataset was collected from ADA Dataverse (McAllister et al. (2023)).

```
voter <- anim_prep_cat(data = aeles,
                       id = id,
                       values = party,
                       time = year,
                       group = gender,
                       order = NULL)

p_voter <- wallaby_plot(data = voter,
                  group_palette = c("pink", "blue", "red"),
                  shade_palette = c("#737373", "#969696", "#BDBDBD",
                                    "#D9D9D9","#D9D9D9","#D9D9D9"),
                  rendering = "gganimate",
                  subset = "top",
                  relation = "one_many",
                  height = 1,
                  size = 2.5,
                  width = 100,
```

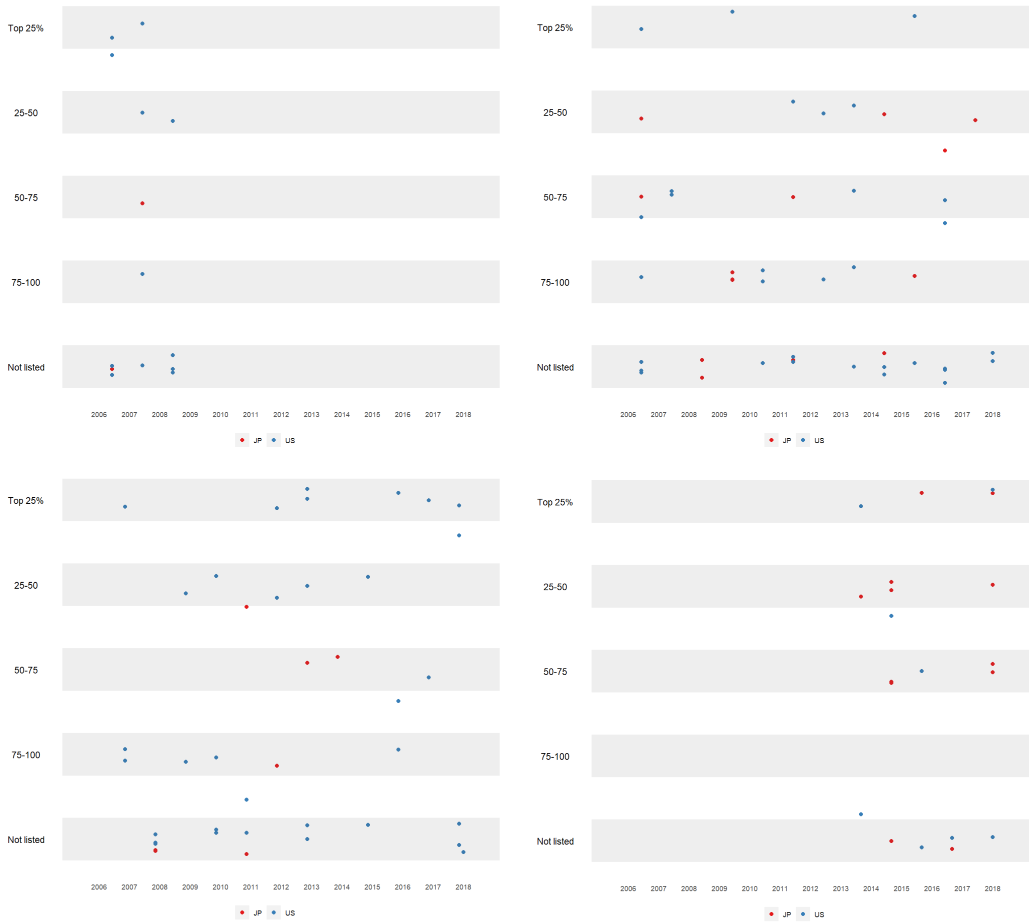


Figure 8: The kangaroo plot visualization shows the movement of the Japanese and US companies between the performance sales quantiles from 2006 to 2018 for a sample of data extracted from the Osiris database. The 'not listed' category indicates companies not yet listed or removed from the listing. Most companies stay in the same quantile group, with a small number moving up and down. Most of the movement is made by American companies, which supports the OECD report.

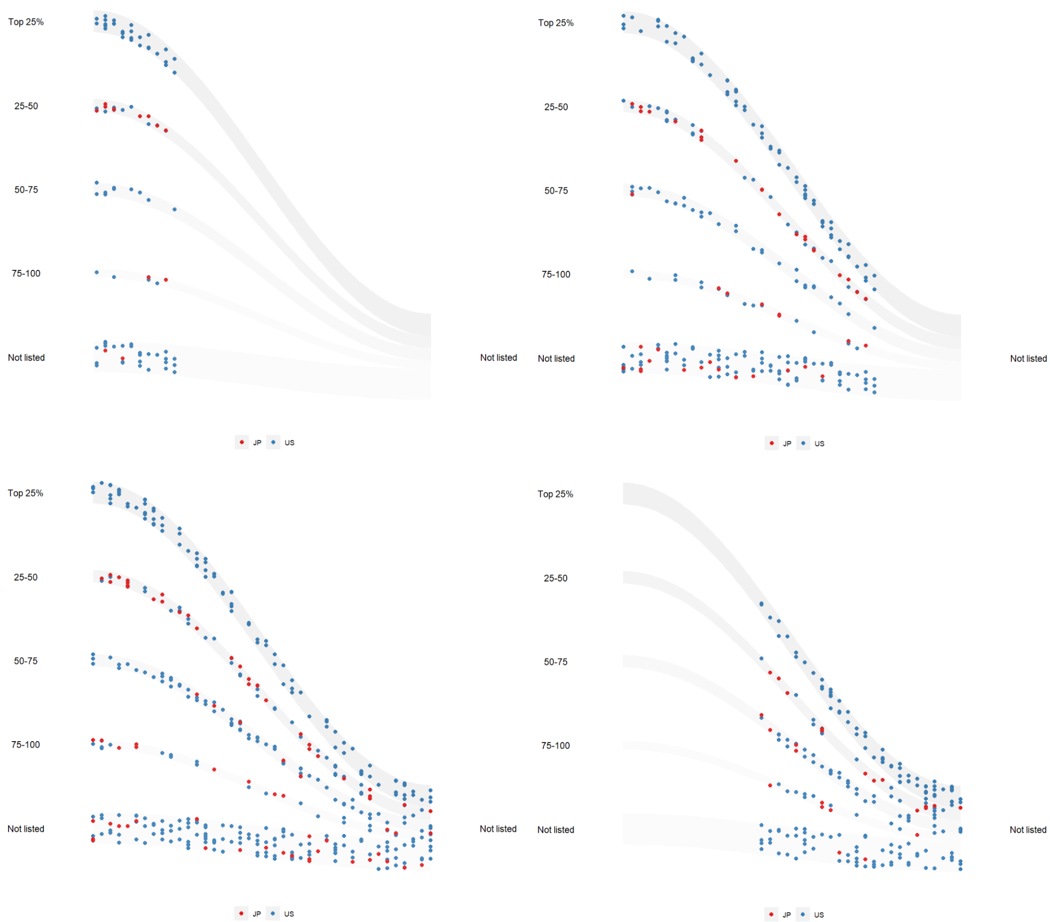


Figure 9: The wallaby plot visualization shows the companies that exited the market. There are more United States companies that fall down into a not listed group (got de-listed) compared to Japanese companies.

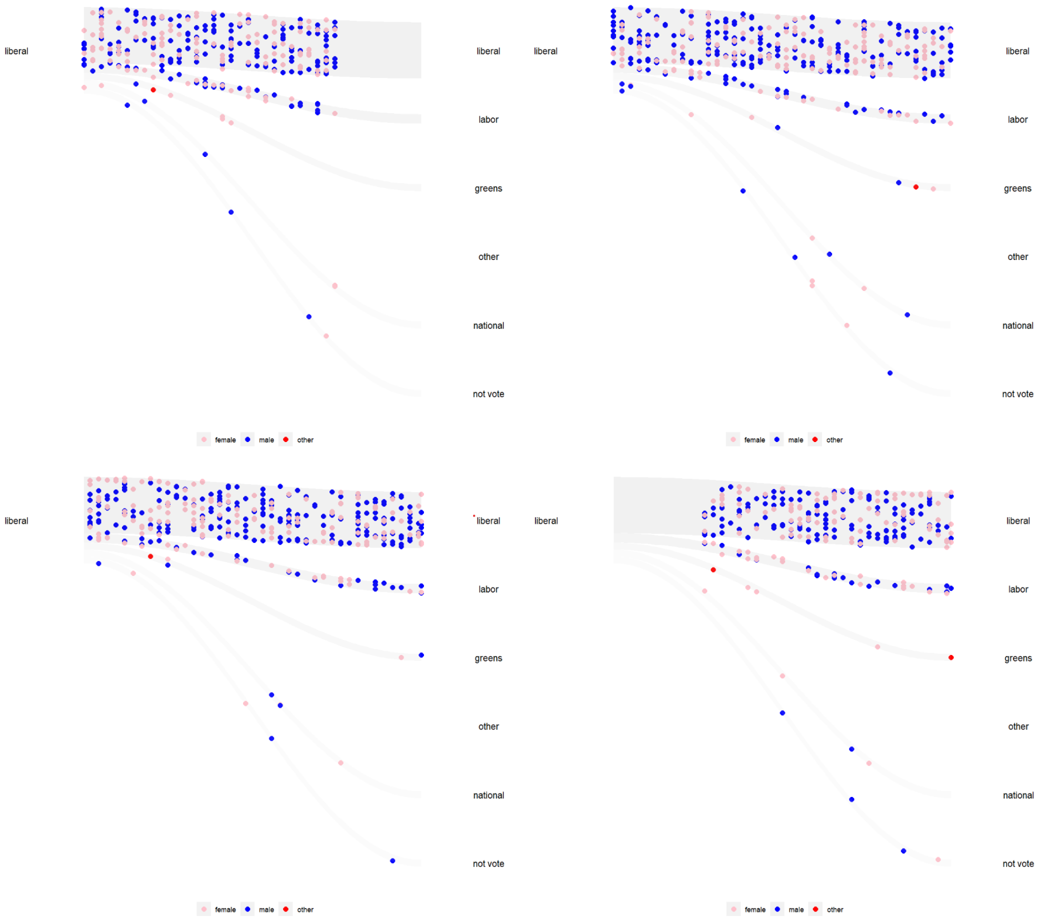


Figure 10: The wallaby plot visualization shows how the top party performs in keeping the old voters of different genders. Most voters remain loyal to the party, but a small fraction of voters with roughly equal male-to-female ratio switch primarily to the other major party. Interestingly, an individual who identified as others overwhelmingly shifted their party affiliations to the Greens party. However, not many of them are shown in the plot.

```
                  total_point = 1000)

p2_voter <- anim_animate(p_voter)

gganimate::animate(p2_voter)
```

From Figure 10, it reveals an intriguing pattern where individuals who identified their gender as 'others' have shifted their voting preference from the Liberal Party, the leading party in 2006, to the Greens Party. This change in behavior could be linked to the Green Party's "A FAIRER, MORE EQUAL COMMUNITY" campaign, which advocates for full equality under the law and communities for LGBTIQ+ individuals.

According to the survey ("LGBTIQ+ Health Australia Party Survey" (2022)), the Greens Party provided a detailed response to all nine priority areas that focus on changing systems and addressing health and well-being disparities among LGBTIQ+ communities. However, the Labor Party only give a broad statement of commitment to a range of LGBTIQ+ and human right issues. This includes working with LGBTIQ+ Australians and advocates to ensure equality before the law and full access to Medicare.

8 Summary

Beginning with inspiration from the New York Times articles, this package provides tools to facilitate the communication of complex data to general audiences. The paper outlines the anticipated data format for input into the prep function, covering raw, categorized, and animated formats. It delves into the animation concept, persistence of vision, achieved through point interpolation, and the visualization design choices. The software is explained in detail with examples and real-world applications. If more time were available, the features allowing a vary in speeds for each observation could enhance the visualization with an additional dimension. Additionally, the new animated and static plots, as demonstrated in the New York Times article, would enable comparisons of demographics at specific points in time.

Acknowledgements

This paper is created using the **rjtools** packages (O'Hara-Wild et al. (2023)) and utilized the **dplyr**, **knitr**, **gganimate**, and **kableExtra** for creating the visual included in this paper. This paper is based on the 0.0.0.9 version of the **animbook** package. This version is available on `https://github.com/KrisanatA/animbook`. I also acknowledge the use of ChatGPT (`https://chat.openai.com/`) to improve the text, grammar, and spelling.

I would like to acknowledge Chika Saka for providing the inspiration that led to the development of this R package. Additionally, thanks to Masayuki Jimichi for the efforts in collecting the sample Osiris data, which contributed to the practical application of the package.

References

Badger, Emily, Claire Cain Miller, Adam Pearce, and Kevin Quealy. 2018. "Extensive Data Shows Punishing Reach of Racism for Black Boys." *The New York Times*. The New York Times. `https://www.nytimes.com/interactive/2018/03/19/upshot/race-class-white-and-black-men.html`.

Bostock, Mike. 2012. "D3.js - Data-Driven Documents." 2012. `http://d3js.org/`.

Caballero, Ricardo J., Takeo Hoshi, and Anil K. Kashyap. 2008. "Zombie Lending and Depressed Restructuring in Japan." *The American Economic Review* 98 (5): 1943–77. `http://www.jstor.org/stable/29730158`.

Chetty, Raj, Nathaniel Hendren, Maggie Jones, and Sonya Porter. 2020. "Race and Economic Opportunity in the United States: An Intergenerational Perspective*." *The Quarterly Journal of Economics* 135 (May): 711–83. `https://doi.org/10.1093/qje/qjz042`.

Hasler, Béatrice, Bernd Kersten, and John Sweller. 2007. "Learner Control, Cognitive Load and Instructional Animation. Applied Cognitive Psychology, 21, 713-729." *Applied Cognitive Psychology* 21 (September): 713–29. `https://doi.org/10.1002/acp.1345`.

Hvitfeldt, Emil. 2018. "Recreate - Sankey Flow Chart." *Recreate - Sankey Flow Chart*. `https://emilhvitfeldt.com/post/2018-03-20-recreate-sankey-flow-chart/`.

"LGBTIQ+ Health Australia Party Survey." 2022. *LGBTIQ+ Health Australia*. `https://www.lgbtiqhealth.org.au/electionsurvey`.

Mayer, Richard. 2010. "Applying the Science of Learning to Medical Education." *Medical Education* 44 (June): 543–49. `https://doi.org/10.1111/j.1365-2923.2010.03624.x`.

Mayer, Richard E. 2005. "Cognitive Theory of Multimedia Learning." In *The Cambridge Handbook of Multimedia Learning*, edited by RichardEditor Mayer, 31–48. Cambridge Handbooks in Psychology. Cambridge University Press. `https://doi.org/10.1017/CBO9780511816819.004`.

Mayer, Richard E., and Roxana Moreno. 2002. *Educational Psychology Review* 14 (1): 87–99. `https://doi.org/10.1023/a:1013184611077`.

McAllister, Ian, Jill Sheppard, Sarah Cameron, and Jackman Simon. 2023. "Australian Election Study, 2022." ADA Dataverse. `https://doi.org/10.26193/W3U2S3`.

McGowan, Müge Adalet, Dan Andrews, and Valentine Millot. 2017. "The Walking Dead?" no. 1372. https://doi.org/`https://doi.org/https://doi.org/10.1787/180d80ad-en`.

Mozilla Foundation. 2011. "WebGL." Khronos WebGL Working Group. www.khronos.org/webgl/.

O'Hara-Wild, Mitchell, Stephanie Kobakian, H. Sherry Zhang, Di Cook, Simon Urbanek, and Christophe Dervieux. 2023. *Rjtools: Preparing, Checking, and Submitting Articles to the 'r Journal'*. `https://CRAN.R-project.org/package=rjtools`.

Ooms, Jeroen. 2023a. *Av: Working with Audio and Video in r*. `https://CRAN.R-project.org/package=av`.

———. 2023b. *Gifski: Highest Quality GIF Encoder*. `https://CRAN.R-project.org/package=gifski`.

"Osiris." 2023. *Bvd*. Bureau van Dijk. `https://www.bvdinfo.com/en-gb/our-products/data/international/osiris`.

Pedersen, Thomas Lin. 2022. *Tweenr: Interpolate Data for Smooth Animations*. `https://CRAN.R-project.org/package=tweenr`.

Pedersen, Thomas Lin, and David Robinson. 2020. "Gganimate: A Grammar of Animated Graphics." *R Package Version* 1 (7): 403–8.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Robertson, George, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. 2008. "Effectiveness of Animation in Trend Visualization." *IEEE Transactions on Visualization and Computer Graphics* 14 (6): 1325–32. https://doi.org/10.1109/TVCG.2008.125.

Shaffer, Jeffrey. 2019. "Sankey Diagrams: Why i Used the Sigmoid Function and Why You Probably Shouldn't." *Data + Science*. https://www.dataplusscience.com/Sigmoid.html.

Sievert, Carson. 2020. "Interactive Web-Based Data Visualization with r, Plotly, and Shiny." Chapman; Hall/CRC. https://plotly-r.com.

Webster, Chris. 2005. *Animation : The Mechanics of Motion*. Focal Press Visual Effects and Animation Series. Oxford ; Burlington, MA: Elsevier Focal Press.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10): 1–23. https://doi.org/10.18637/jss.v059.i10.

———. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*.

Wickham, Hadley, and Lionel Henry. 2023. *Purrr: Functional Programming Tools*. https://CRAN.R-project.org/package=purrr.

Krisanat Anukarnsakulchularp
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
ORCiD: 0009-0008-5638-7124
kanu0003@student.monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
ORCiD: 0000-0002-3813-7155
dicook@monash.edu