Calendar-based graphics for visualizing people's daily schedules

by Earo Wang, Dianne Cook, Rob J Hyndman

Abstract Calendars are broadly used in society to display temporal information, and events. This paper describes a new R package with functionality to organize and display temporal data, collected on sub-daily resolution, into a calendar layout. The function `frame_calendar` uses linear algebra on the date variable to restructure data into a format lending itself to calendar layouts. The user can apply the grammar of graphics to create plots inside each calendar cell, and thus the displays synchronize neatly with **ggplot2** graphics. The motivating application is studying pedestrian behavior in Melbourne, Australia, based on counts which are captured at hourly intervals by sensors scattered around the city. Faceting by the usual features such as day and month, was insufficient to examine the behavior. Making displays on a monthly calendar format helps to understand pedestrian patterns relative to events such as work days, weekends, holidays, and special events. The layout algorithm has several format options and variations. It is implemented in the R package **sugrrants**.

Introduction

We develop a method for organizing and visualizing temporal data, collected at sub-daily intervals, into a calendar layout. The calendar format is created using linear algebra, giving a restructuring of the data, that can then be piped into grammar of graphics definitions of plots, as used in **ggplot2** (Wickham et al. 2018). The data restructuring approach is consistent with the tidy data principles available in the **tidyverse** (Wickham 2017) suite. The methods are implemented in a new package called **sugrrants** (Wang, Cook, and Hyndman 2018) .

The purpose of the calendar-based visualization is to provide insights into people's daily schedules relative to events such as work days, weekends, holidays, and special events. This work was originally motivated by studying foot traffic in the city of Melbourne, Australia (City of Melbourne 2017). There have been 43 sensors installed that count pedestrians every hour across the downtown area (Figure 1). The dataset can shed light on people's daily rhythms, and assist the city administration and local businesses with event planning and operational management. A routine examination of the data would involve constructing conventional time series plots to catch a glimpse of temporal patterns. The faceted plots in Figure 2, give an overall picture of the foot traffic at 3 different sensors over 2016. Further faceting by day of the week (Figure 3) provides a better glimpse of the daily and sub-daily pedestrian patterns. The sensor data, like many temporal datasets on human behaviors, lends itself to a number of exploratory data visualization challenges:

1. Variations exist on multiple time scales including time of day, day of week, and day of year (such as public holiday and recurring events).
2. Since the data are often collected at sub-daily frequencies, they typically involve a large number of observations.
3. Measurements of a single type are made at multiple locations at a given time point, which creates the need for comparing and contrasting between locations.

The conventional ways to display time series and constructing faceted displays are insufficient to address these challenges.

The work is inspired by Wickham et al. (2012), which uses linear algebra to display spatio-temporal data as glyphs on maps. It is also related to recent work by Hafen (2018) which provides methods in the **geofacet** package to arrange data plots into a grid, while preserving the geographical position. Both of these show data in a spatial context.

In contrast, calendar-based graphics unpack the temporal variable, at different resolutions, to digest multiple seasonalities, and special events. There is some existing work in this area. For example, Van Wijk and Van Selow (1999) developed a calendar view of the heatmap to represent the number of employees in the work place over a year, where colors indicate different clusters derived from the days. It contrasts week days and weekends, highlights public holidays, and presents other known seasonal variation such as school vacations, all of which have influence over the turn-outs in the office. The calendar-based heatmap was implemented in two R packages: **ggTimeSeries** (Kothari and Ather 2016) and **ggcal** (Jacobs 2017). However, these techniques are limited to color-encoding graphics and are unable to use time scales smaller than a day. Time of day, which serves as one of the most important aspects in explaining substantial variations arising from the pedestrian sensor data, will be neglected through daily aggregation. Additionally, if simply using colored blocks rather than curves, it may

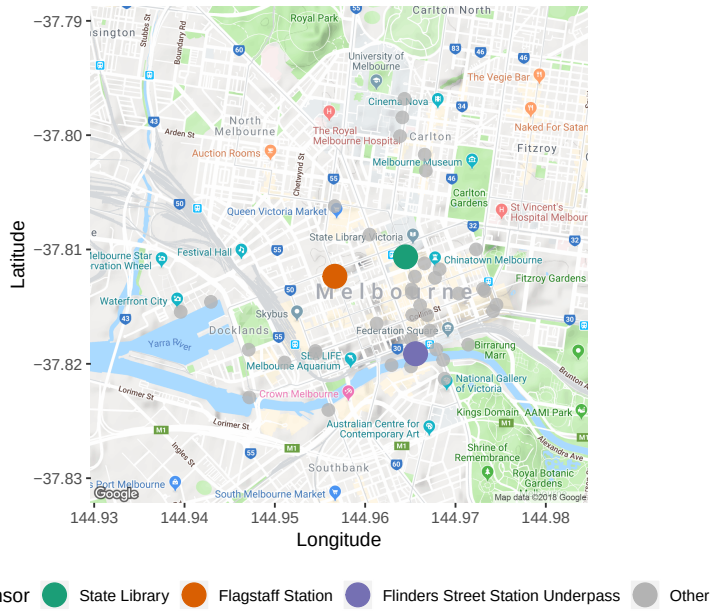

Figure 1: Map of the Melbourne city area with dots indicating sensor locations. 3 sensors have been highlighted to give a closer look at in the paper.

Figure 2: Time series plots showing the number of pedestrians in 2016 measured at 3 different sensors in the city of Melbourne. Colored by the sensors, small multiples of lines show that the foot traffic varies from one sensor to another in terms of both time and number. The weekly patterns look distinctive across these 3 sensors. There is an eye-catching spike which occurred at the State Library, caused by the annual White Night event on 20th of February.

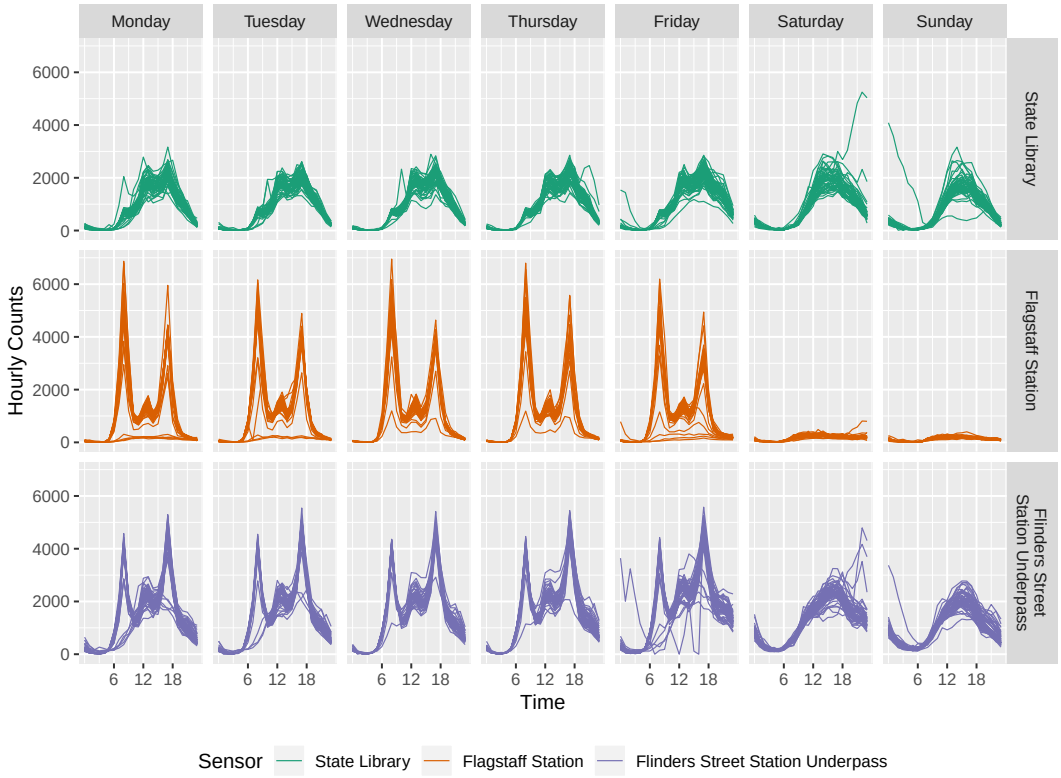

Figure 3: Hourly pedestrian counts for 2016 faceted by sensors and days of the week using lines. It features at least two types of seasons—time of day and day of week—across all the sensors. The White Night effect on increasing pedestrian counts can be seen on Saturday in February at the State Library.

become perceptually difficult to estimate the shape positions and changes, although using curves comes with the cost of more display capacity (Cleveland and McGill 1984; Lam, Munzner, and Kincaid 2007).

We propose a new algorithm to go beyond the calendar-based heatmap. The approach is developed with three conditions in mind: (1) to display time-of-day variation in addition to longer temporal components such as day-of-week and day-of-year; (2) to incorporate line graphs and other types of glyphs into the graphical toolkit for the calendar layout; (3) to enable overlaying plots consisting of multiple time series. The proposed algorithm has been implemented in the `frame_calendar` function in the **sugrrants** package using R.

The remainder of the paper is organized as follows. Section 2.2 demonstrates the construction of the calendar layout in depth. Section 2.3 lists and describes the options that come with the `frame_calendar` function. Section 2.4 presents some variations of its usage. Section 2.5 discusses the advantages and disadvantages of the method.

Construction

Figure 4 shows the line glyphs framed in the monthly calendar over the year 2016. This is achieved by the `frame_calendar` function, which computes the coordinates on the calendar for the input data variables. These can then be plotted using the usual **ggplot2** package (Wickham et al. 2018) functions. All of the grammar of graphics (Wilkinson 2005; Wickham 2009) can be applied.

The algorithm for constructing a calendar plot uses linear algebra, similar to that used in the glyph map displays for spatio-temporal data (Wickham et al. 2012). To make a year long calendar requires cells for days, embedded in blocks corresponding to months, organized into a grid layout for a year. Each month can be captured with 35 (5×7) cells, where the top left is Monday of week 1, and the bottom right is Sunday of week 5 by default. These cells provide a micro canvas on which to plot the data. The first day of the month could be any of Monday–Sunday, which is determined by the year of the calendar. Months are of different lengths, ranging from 28 to 31 days, and each month could extend over six weeks but the convention in these months is to wrap the last few days up to the top row of the block. The notation for creating these cells is as follows:

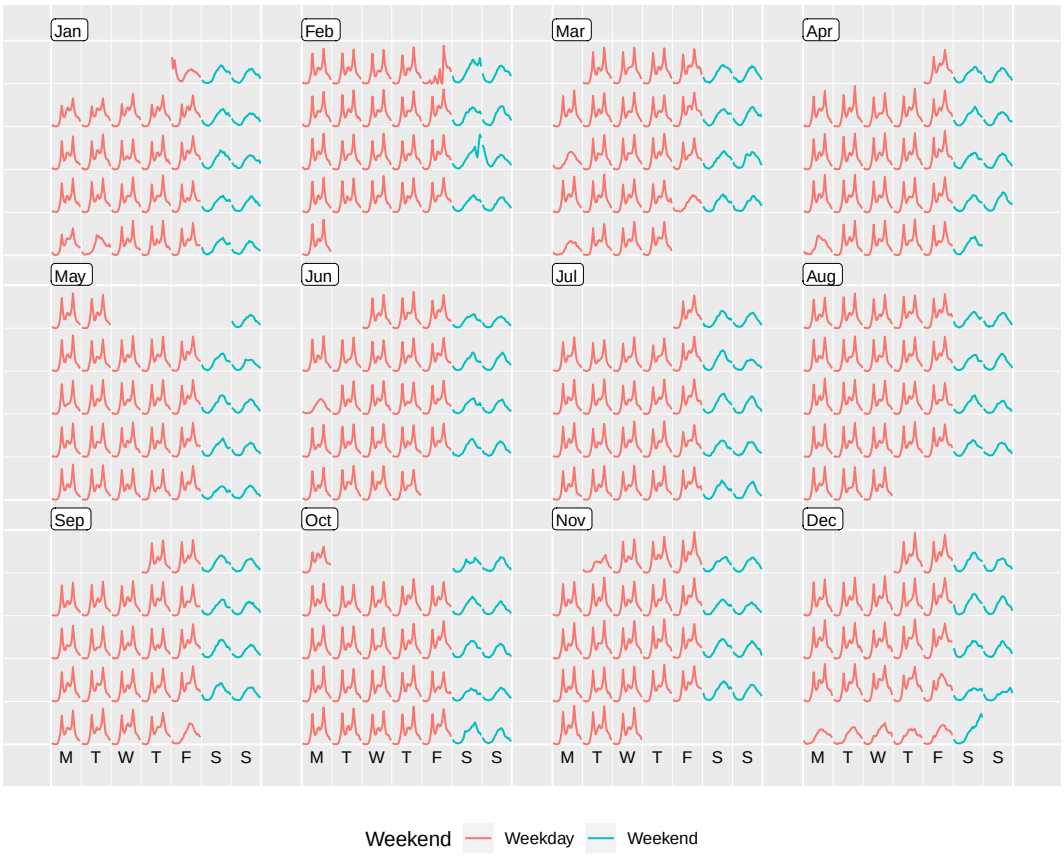

Figure 4: The calendar-based display of hourly foot traffic at Flinders Street Station using line glyphs. The arrangement of the data into a 3 by 4 monthly grid represents all the traffic in 2016. The disparities between week day and weekend along with public holiday are immediately apparent.

				g=k+5, g=5 i=1, j=5	g=k+1 i=1, j=6	g=k+2 i=1, j=7
g=k+3 i=2, j=1	g=k+4 i=2, j=2	g=k+5 i=2, j=3	g=k+6 i=2, j=4	g=k+7 i=2, j=5	g=k+8 i=2, j=6	g=k+9 i=2, j=7
g=k+10 i=3, j=1	g=k+11 i=3, j=2	g=k+12 i=3, j=3	g=k+13 i=3, j=4	g=k+14 i=3, j=5	g=k+15 i=3, j=6	g=k+16 i=3, j=7
g=k+17 i=4, j=1	g=k+18 i=4, j=2	g=k+19 i=4, j=3	g=k+20 i=4, j=4	g=k+21 i=4, j=5	g=k+22 i=4, j=6	g=k+23 i=4, j=7
g=k+24 i=5, j=1	g=k+25 i=5, j=2	g=k+26 i=5, j=3	g=k+27 i=5, j=4	...	...	g=k+d i=5, j=7

Figure 5: Illustration of the indexing layout for cells in a month, where k is day of the week, g is day of the month, (i, j) indicates grid position.

- $k = 1, \ldots, 7$ is the day of the week that is the first day of the month.
- $d = 28, 29, 30$ or 31 representing the number of days in any month.
- (i, j) is the grid position where $1 \leq i \leq 5$ is week within the month, $1 \leq j \leq 7$, is day of the week.
- $g = k, \ldots, (k + d)$ indexes the day in the month, inside the 35 possible cells.

The grid position for any day in the month is given by

$$\begin{aligned} i &= \lceil (g \bmod 35)/7 \rceil, \\ j &= g \bmod 7. \end{aligned} \tag{1}$$

Figure 5 illustrates this (i, j) layout for a month where $k = 5$.

To create the layout for a full year, (m, n) denotes the position of the month arranged in the plot, where $1 \leq m \leq M$ and $1 \leq n \leq N$. Between each month requires some small amount of white space, denoted by b. Figure 6 illustrates this layout where $M = 3$ and $N = 4$.

Each cell forms a canvas on which to draw the data. Initialize the canvas to have limits $[0, 1]$ both horizontally and vertically. For the pedestrian sensor data, within each cell, hour is plotted horizontally and count is plotted vertically. Each variable is scaled to have values in $[0, 1]$, using the minimum and maximum of all the data values to be displayed, assuming fixed scales. Let h be the scaled hour, and c the scaled count.

Then the final points for making the calendar line plots of the pedestrian sensor data is given by:

$$\begin{aligned} x &= j + (n-1) \times 7 + (n-1) \times b + h, \\ y &= i - (m-1) \times 5 - (m-1) \times b + c. \end{aligned} \tag{2}$$

Note that for the vertical direction, the top left is the starting point of the grid (in Figure 5) which is why subtraction is performed. Within each cell, the starting position is the bottom left.

In order to make calendar-based graphics more accessible and informative, reference lines dividing each cell and block as well as labels indicating week day and month are also computed before plot construction.

Regarding the monthly calendar, the major reference lines separate every month panel and the minor ones separate every cell, represented by the thick and thin lines in Figure 4, respectively. The major reference lines are placed surrounding every month block: for each m, the vertical lines are determined by $\min(x)$ and $\max(x)$; for each n, the horizontal lines are given by $\min(y)$ and $\max(y)$. The minor reference lines are only placed on the left side of every cell: for each i, the vertical division is $\min(x)$; for each j, the horizontal is $\min(y)$.

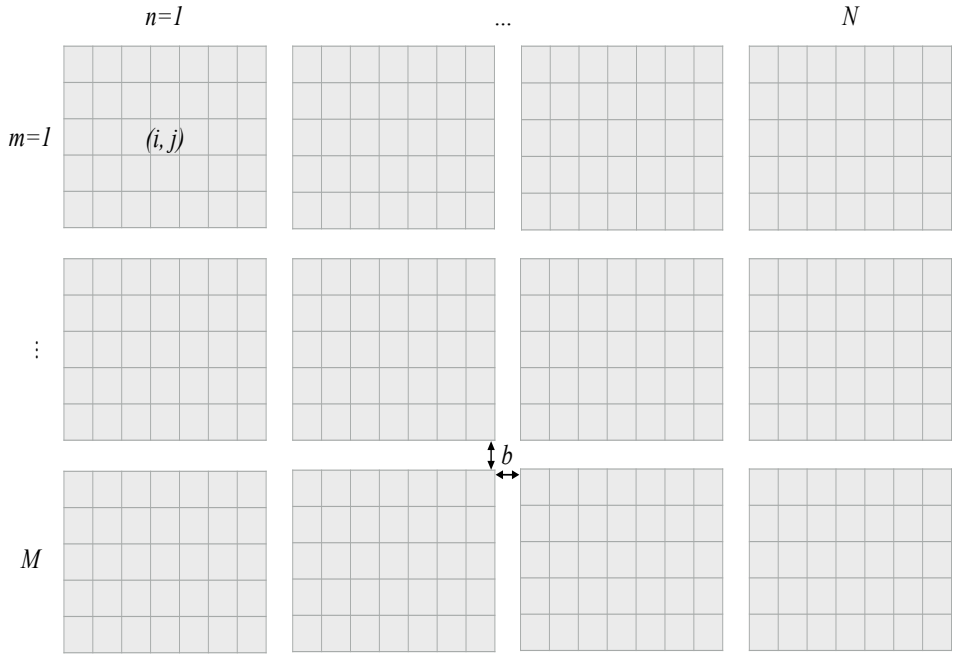

Figure 6: Illustration of the indexing layout for months of one year, where M and N indicate number of rows and columns, b is a space parameter separating cells.

The month labels located on the top left using $(\min(x), \max(y))$ for every (m, n). The week day texts are uniformly positioned on the bottom of the whole canvas, that is $\min(y)$, with the central position of a cell $x/2$ for each j.

Options

The algorithm has several optional parameters that modify the layout, direction of display, scales, plot size and switching to polar coordinates. These are accessible to the user by the inputs to the function `frame_calendar`:

```
frame_calendar(
  data, x, y, date, calendar = "monthly", dir = "h",
  sunday = FALSE, nrow = NULL, ncol = NULL, polar = FALSE,
  scale = "fixed", width = 0.95, height = 0.95, margin = NULL
)
```

It is assumed that the `data` is in tidy format (Wickham 2014), and `x`, `y` are the variables that will be mapped to the horizontal and vertical axes in each cell. For example, the `x` is the time of the day, and `y` is the count (Figure 4). The `date` argument specifies the date variable used to construct the calendar layout.

The algorithm handles displaying a single month or several years. The arguments `nrow` and `ncol` specify the layout of multiple months. For some time frames, some arrangements may be more beneficial than others. For example, to display data for three years, setting `nrow = 3` and `ncol = 12` would show each year on a single row.

Layouts

The monthly calendar is the default, but two other formats, weekly and daily, are available with the `calendar` argument. The daily calendar arranges days along a row, one row per month. The weekly calendar stacks weeks of the year vertically, one row for each week, and one column for each day. The reader can scan down all the Mondays of the year, for example. The daily layout puts more emphasis on day of the month. The weekly calendar is appropriate if most of the variation can be characterized by days of the week. On the other hand, the daily calendar should be used when there is a yearly effect but not a weekly effect in the data (for example weather data). When both effects are present,

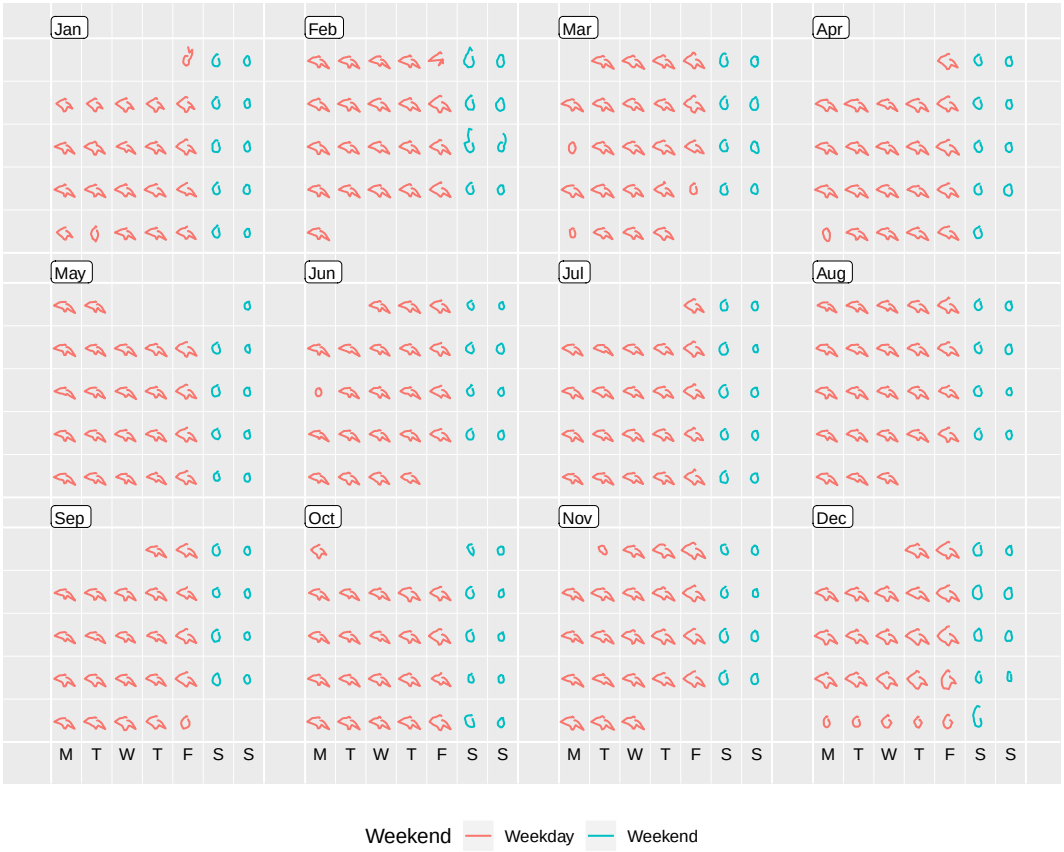

Figure 7: Figure 4 in circular layout, which is referred to as star plots. The daily periodicity on work days are clearly visible.

the monthly calendar would be a better choice. Temporal patterns motivate which variant should be employed.

Polar transformation

When `polar = TRUE`, a polar transformation is carried out on the data. The computation is similar to the one described in Wickham et al. (2012). Figure 7 shows star plots embedded in the monthly calendar layout, which is equivalent to Figure 4 placed in polar coordinates.

Scales

By default, global scaling is done for values in each plot, with the global minimum and maximum used to fit values into each cell. If the emphasis is comparing trend rather than magnitude, it is useful to scale locally. For temporal data this would harness the temporal components. The choices include: free scale within each cell (`free`), cells derived from the same day of the week (`free_wday`), or cells from the same day of the month (`free_mday`). The scaling allows for the comparisons of absolute or relative values, and the emphasis of different temporal variations.

With local scaling, the overall variation gives way to the individual shape. Figure 8 shows the same data as Figure 4 scaled locally using `scale = "free"`. The daily trends are magnified.

The `free_wday` scales each week day together. It can be useful to comparing trends across week days, allowing relative patterns for weekends versus week days to be examined. Similarly, the `free_mday` uses free scaling for any day within a given month.

Orientation

By default, grids are laid out horizontally. This can be transposed by setting the `dir` parameter to `"v"`, in which case i and j are swapped in the Equation 1. This can be useful for creating calendar layouts for countries where vertical layout is the convention.

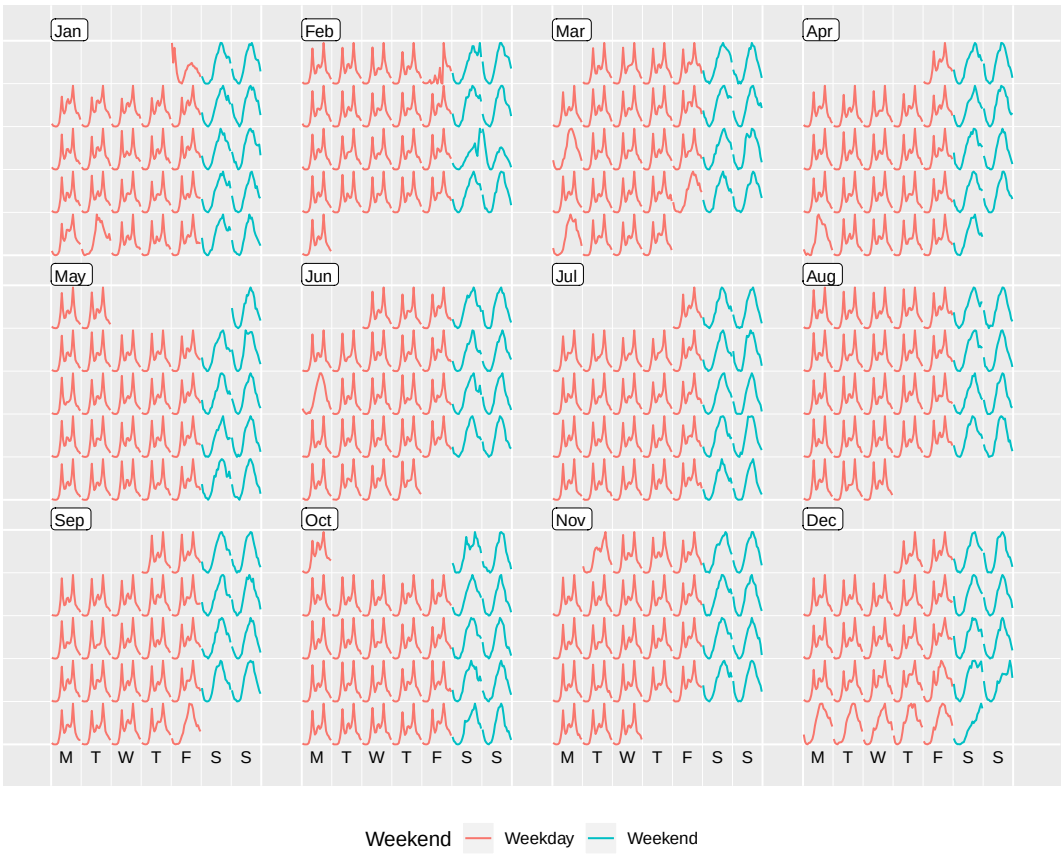

Figure 8: Line glyphs on the calendar format showing hourly foot traffic at Flinders Street Station, scaled over all the days. The individual shape on a single day becomes more distinctive, however it is impossible to compare the size of peaks between days.

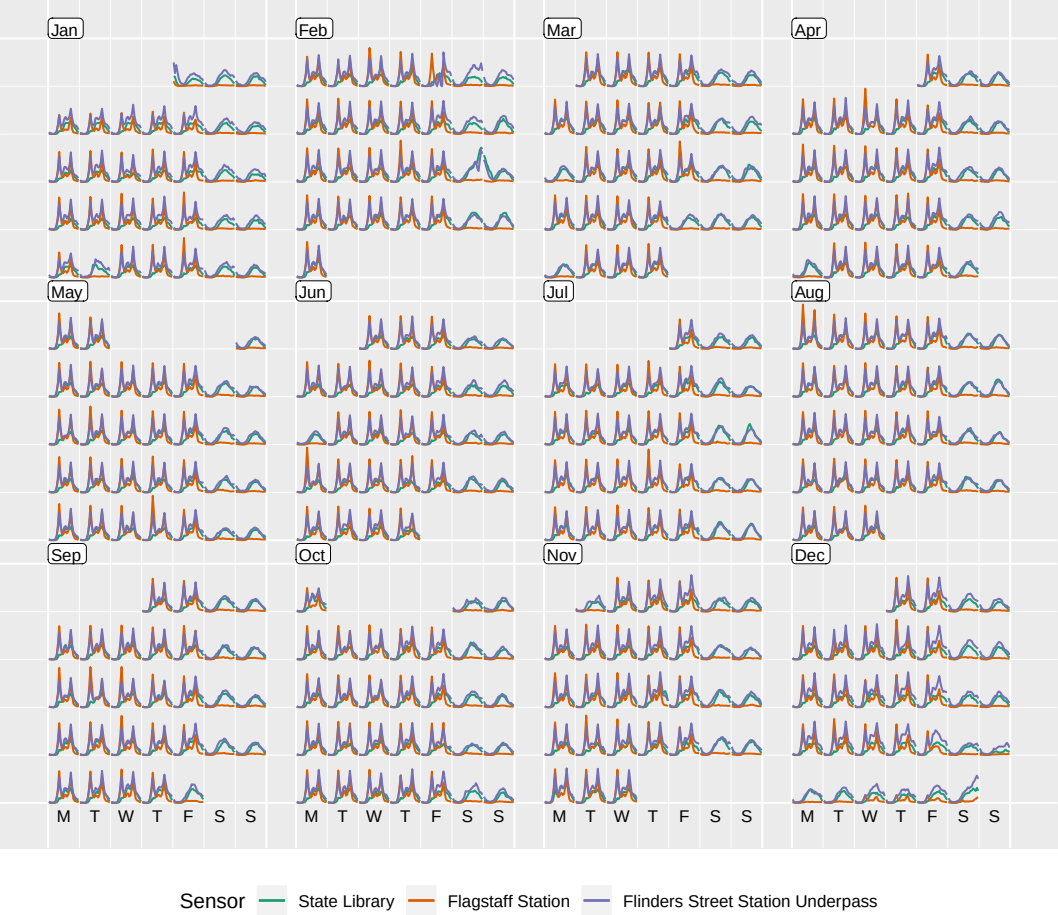

Figure 9: Overlaying line graphs of the 3 sensors in the monthly calendar. Flagstaff station is not as busy as the other two on non-work days.

Language support

Most countries have adopted this western calendar layout, while the languages used for week day and month would be different across countries. We also offer languages other than English for text labelling. Figure 13 shows the same plot as Figure 12 labelled using simplified Chinese characters.

Variations

Overlaying and faceting subsets

Plots can be layered. The comparison of sensors can be done by overlaying plot the values for each (Figure 9). Differences between the pedestrian patterns for Flinders Street and Flagstaff train stations can be seen. Both Flinders Street and Flagstaff train stations exhibit strong commuters patterns, however, Flinders Street has higher pedestrian counts during the weekends and public holidays. This suggests that Flagstaff Station has limited functionality on non-work days, but other activities occur around the Flinders Street Station. From Figure 9 it can be seen that the State Library has a similar temporal trend to Flinders Street Station on non-work days. The nighttime events, such as White Night and New Year's Eve, have barely affected the operation of Flagstaff Station but heavily affected the incoming and outgoing traffic to Flinders Street Station and the State Library.

To avoid the overlapping problem, the calendar layout can be embedded into a series of subplots for the different sensors. Figure 10 presents the idea of faceting calendar plots. This allows comparing the overall structure between sensors, while emphasising individual sensor variation.

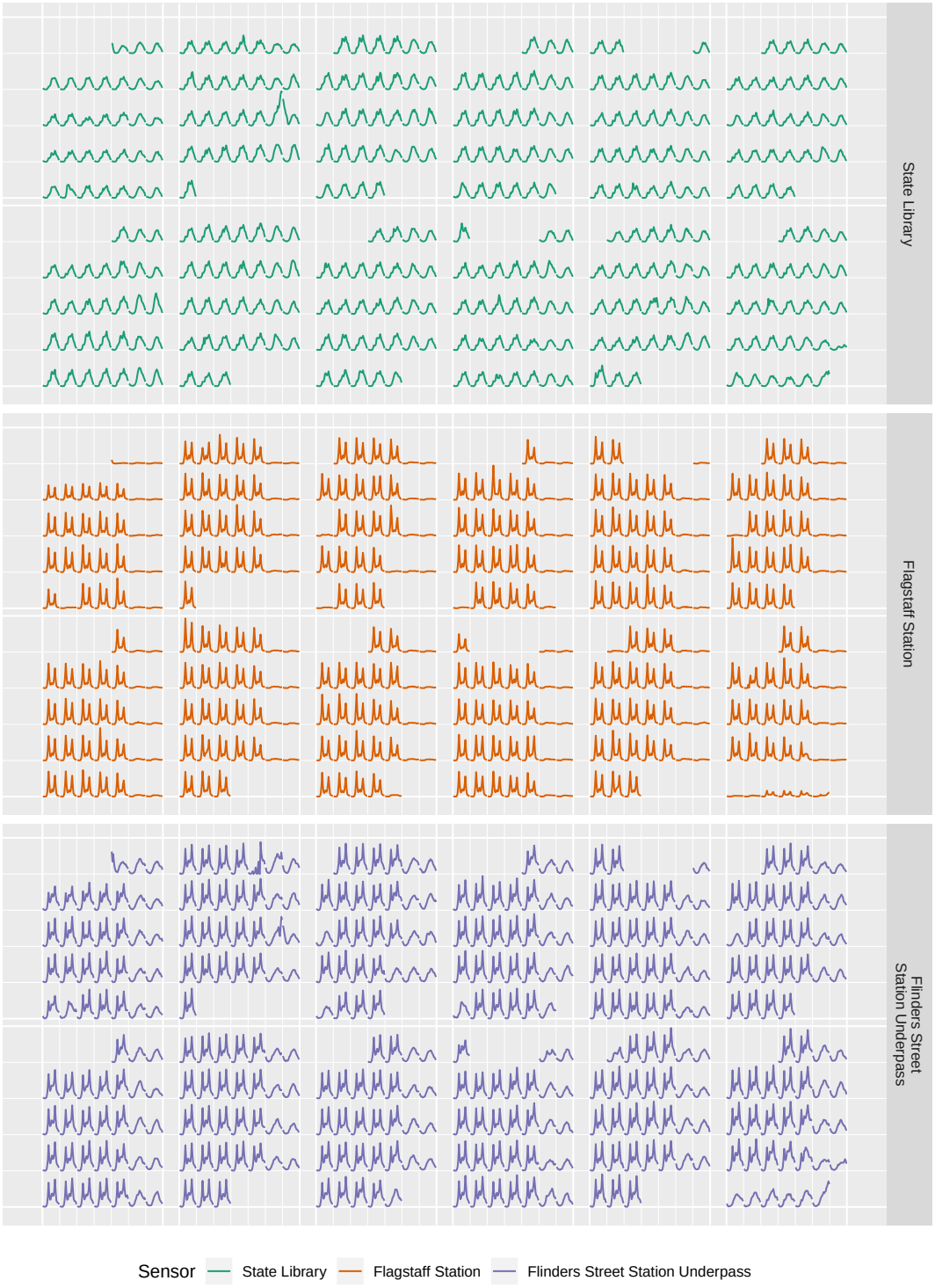

Figure 10: Line charts, embedded in the 6 by 2 monthly calendar, colored and faceted by the 3 sensors. The variations of an individual sensor are emphasised, and the shapes can be compared across the cells and sensors.

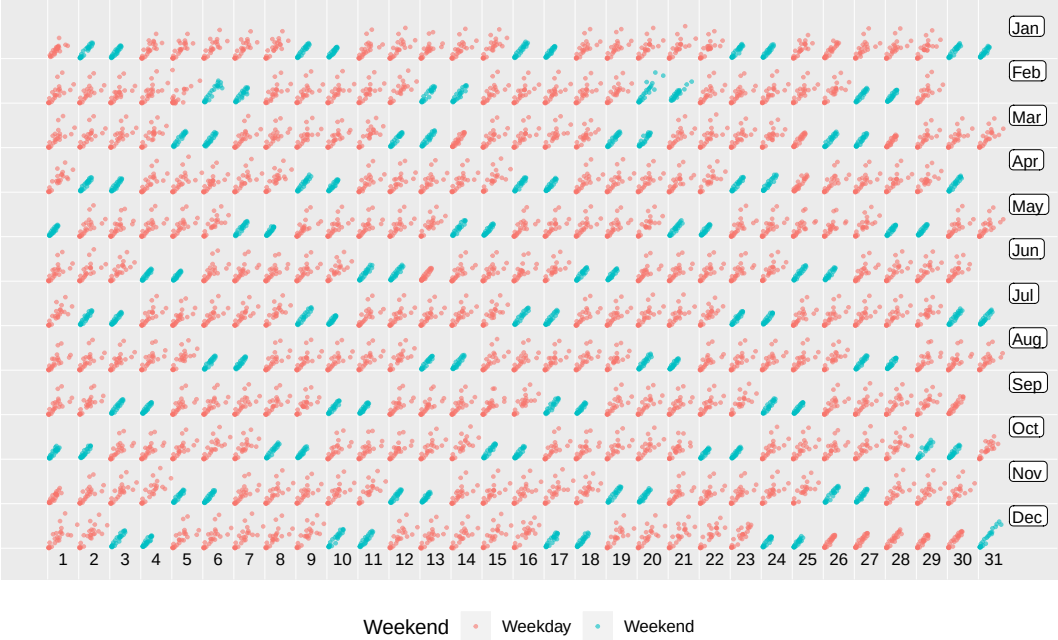

Figure 11: Lag scatterplot in the daily calendar layout. Each hour's count is plotted against previous hour's count at Flinders Street Station to demonstrate the autocorrelation at lag 1. The correlation between them is more consistent on non-work days than work days.

Different types of plots

The `frame_calendar` function is not constrained to line plots. The full range of plotting capabilities in **ggplot2** is essentially available. Figure 11 shows a lag scatterplot at Flinders Street Station, where the lagged hourly count is assigned to the `x` argument and the current hourly count to the `y` argument. This figure is organized in the daily calendar layout. Figure 11 indicates two primary patterns, strong autocorrelation on weekends, and weaker autocorrelation on work days. At the higher counts, on week days, the next hour sees possibly substantial increase or decrease in counts, essentially revealing a bimodal distribution of consecutive counts, as supported by Figure 4.

The algorithm can also produce more complicated plots, such as boxplots. Figure 12 uses a loess smooth line superimposed on side-by-side boxplots. It shows the distribution of hourly counts across all 43 sensors during December. The last week of December is the holiday season: people are off work on the day before Christmas, go shopping on the Boxing day, and stay out for the fireworks on New Year's Eve.

Discussion

The calendar-based visualization provides data plots in the familiar format of an everyday tool. Patterns on special events for the region, like Anzac Day in Australia, or Thanksgiving Day in the USA, more easily pop out to the viewer as public holidays, than they would on a more commonly used week day and month faceted layout.

The focus is on the western calendar layout, because most countries have adopted this format. However the language of labels would differ across countries, and so support was added for changing the label language simply.

This sort of layout will be useful for studying consumer trends, or human behavior, such as pedestrian patterns or pollution peaks. It will not be so useful for physical patterns like temperature, which are not typically affected by human activity. The layout does not replace traditional displays, but serves to complement to further tease out structure in temporal data. Analysts would still be advised to plot overall summaries and deviations, in order to study general trends.

The layout is achieved by utilizing linear combinations of temporal components and measured variables. This provides the data with the most screen real estate. It could be useful to develop this into a fully-fledged faceting method, with formal labels and axes. This is a future goal.

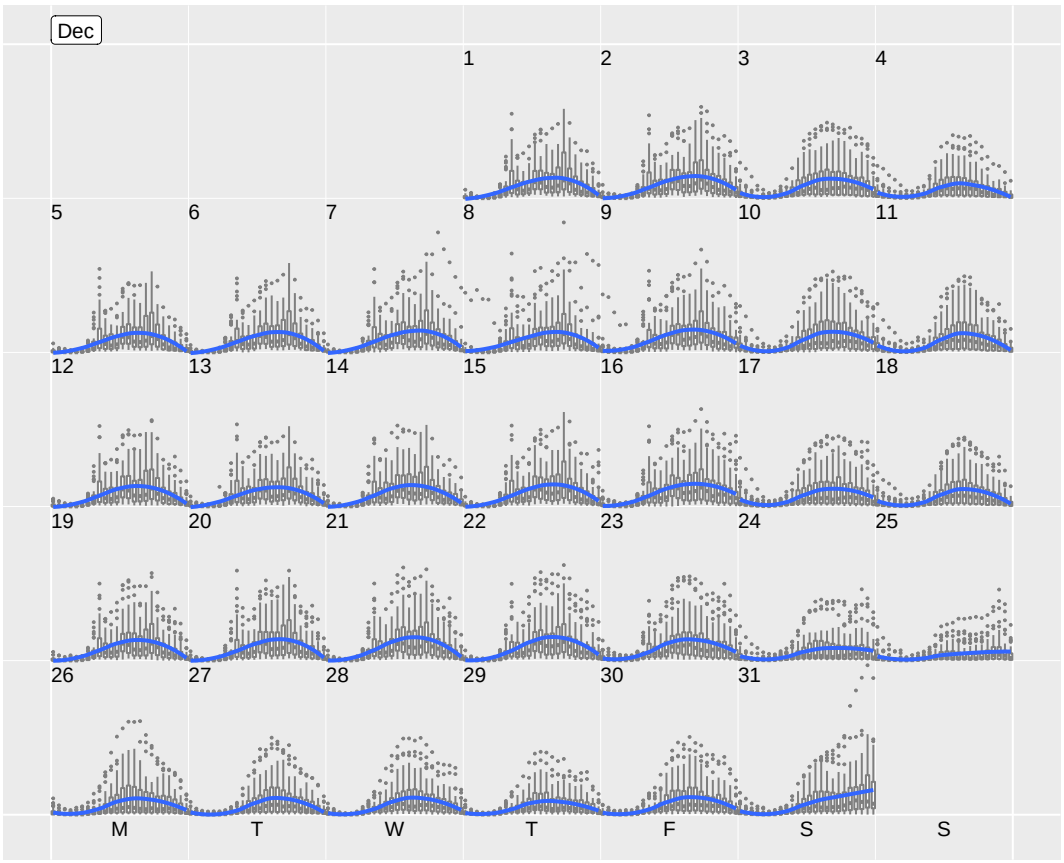

Figure 12: Side-by-side boxplots of hourly counts for all the 43 sensors in December 2016, with the loess smooth line superimposed on each day. It shows the hourly distribution in the city as a whole. There is one sensor attracting a larger number of people on New Year's Eve than the rest.

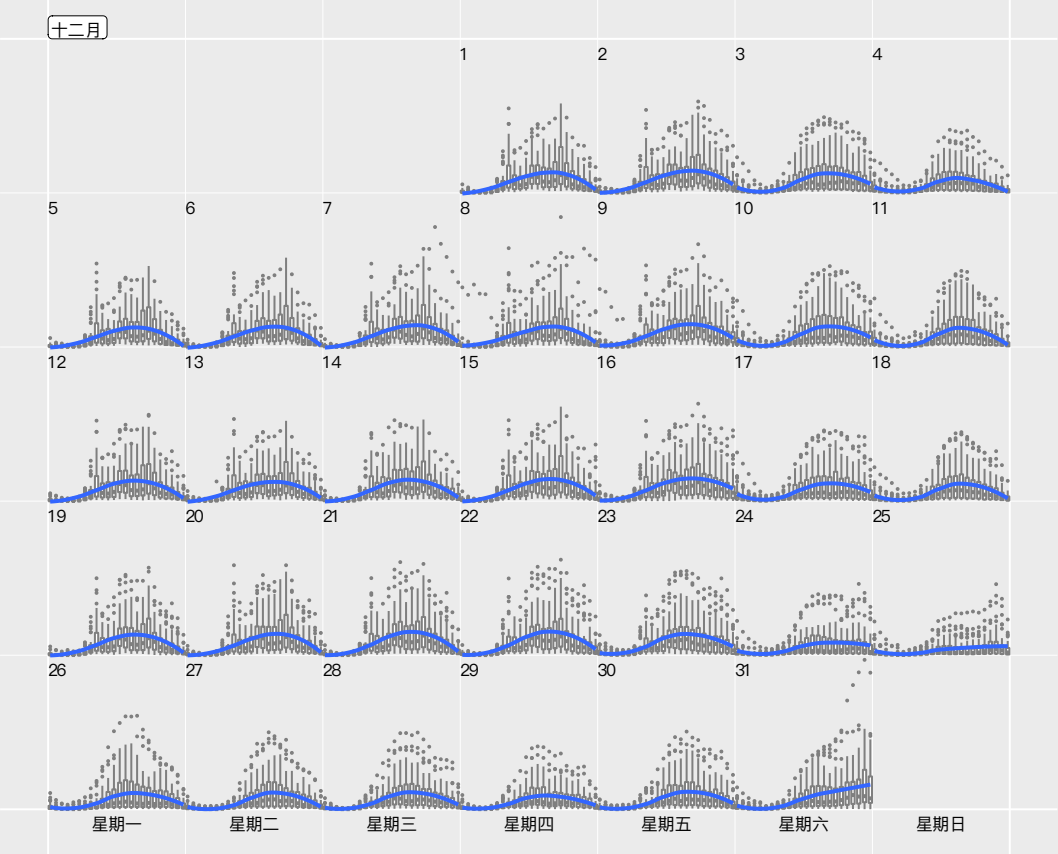

Figure 13: The same plot as Figure 12, but with the month and week day labels in Chinese. It demonstrates the natural support for languages other than English.

Bibliography

City of Melbourne. 2017. *Pedestrian Volume in Melbourne*. City of Melbourne, Australia. http://www.pedestrian.melbourne.vic.gov.au.

Cleveland, William S, and Robert McGill. 1984. "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods." *Journal of the American Statistical Association* 79 (387). Taylor & Francis: 531–54.

Hafen, Ryan. 2018. *Geofacet: 'Ggplot2' Faceting Utilities for Geographical Data*. https://CRAN.R-project.org/package=geofacet.

Jacobs, Jay. 2017. *Ggcal: Calendar Plot Using Ggplot2*. https://github.com/jayjacobs/ggcal.

Kothari, Aditya, and Ather. 2016. *ggTimeSeries: Nicer Time Series Visualisations with Ggplot Syntax*. https://github.com/Ather-Energy/ggTimeSeries.

Lam, Heidi, Tamara Munzner, and Robert Kincaid. 2007. "Overview Use in Multiple Visual Information Resolution Interfaces." *IEEE Transactions on Visualization and Computer Graphics* 13 (6). IEEE: 1278–85.

Van Wijk, Jarke J., and Edward R. Van Selow. 1999. "Cluster and Calendar Based Visualization of Time Series Data." In *Information Visualization, 1999. INFOVIS 1999 Proceedings. IEEE Symposium on*, 4–9. IEEE.

Wang, Earo, Di Cook, and Rob Hyndman. 2018. *Sugrrants: Supporting Graphs for Analysing Time Series*. https://pkg.earo.me/sugrrants.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. New York, NY: Springer-Verlag New York.

———. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10). Foundation for Open Access Statistics: 1–23.

———. 2017. *Tidyverse: Easily Install and Load the 'Tidyverse'*. https://CRAN.R-project.org/package=tidyverse.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, and Kara Woo. 2018. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*.

Wickham, Hadley, Heike Hofmann, Charlotte Wickham, and Dianne Cook. 2012. "Glyph-Maps for Visually Exploring Temporal Patterns in Climate Data and Models." *Environmetrics* 23 (5): 382–93.

Wilkinson, Leland. 2005. *The Grammar of Graphics (Statistics and Computing)*. Secaucus, NJ: Springer-Verlag New York, Inc.

Earo Wang
Monash University
Department of Econometrics and Business Statistics,
Monash University, VIC 3800
Australia
earo.wang@monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics,
Monash University, VIC 3800
Australia
dicook@monash.edu

Rob J Hyndman
Monash University
Department of Econometrics and Business Statistics,
Monash University, VIC 3800
Australia
rob.hyndman@monash.edu