Calendar-based graphics for visualizing people's daily schedules

Earo Wang
Department of Econometrics and Business Statistics, Monash University
and
Dianne Cook
Department of Econometrics and Business Statistics, Monash University
and
Rob J Hyndman
Department of Econometrics and Business Statistics, Monash University

October 16, 2018

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**.

Keywords: data visualization, statistical graphics, time series, grammar of graphics, R

1 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 et al. 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.

However, the conventional displays of time series data obscure patterns relative to special events (such as public holidays and recurring cultural/sport events), which may appear more fascinating to viewers.

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 & 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,

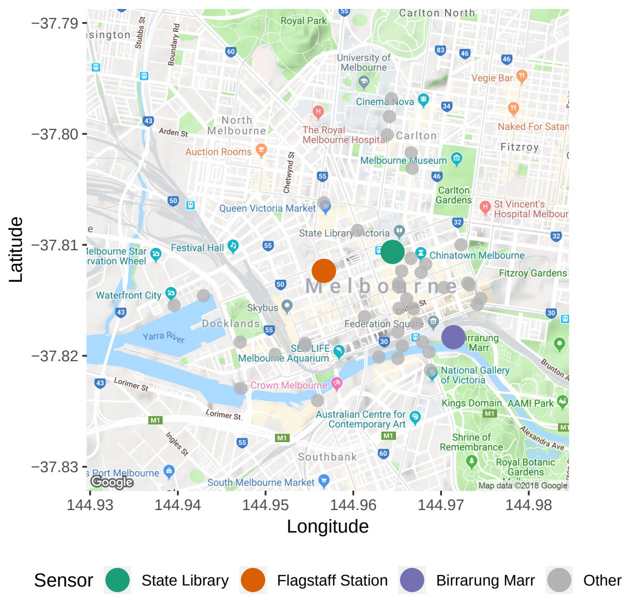

Figure 1: Map of the Melbourne city area with dots indicating sensor locations. 3 sensors have been highlighted to give a closer look at in the paper.

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 & 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 become perceptually difficult to estimate the shape positions and changes, although using curves comes with the cost of more display capacity (Cleveland & McGill 1984, Lam et al. 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.

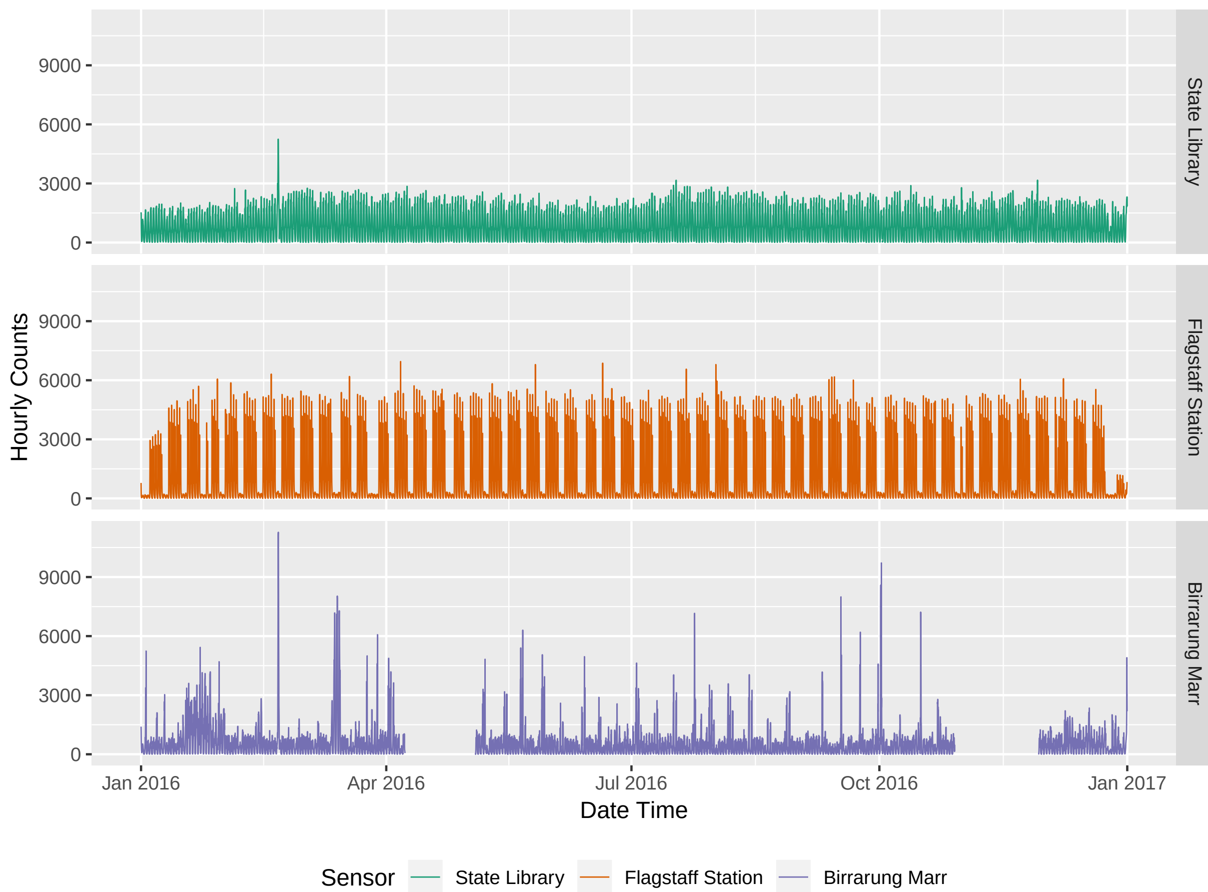

Figure 2: Time series plots showing the number of pedestrians in 2016 measured at 3 different sensors in the city of Melbourne. Colored by the sensors, small multiples of lines show that the foot traffic varies from one sensor to another in terms of both time and number. The weekly patterns look distinctive 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.

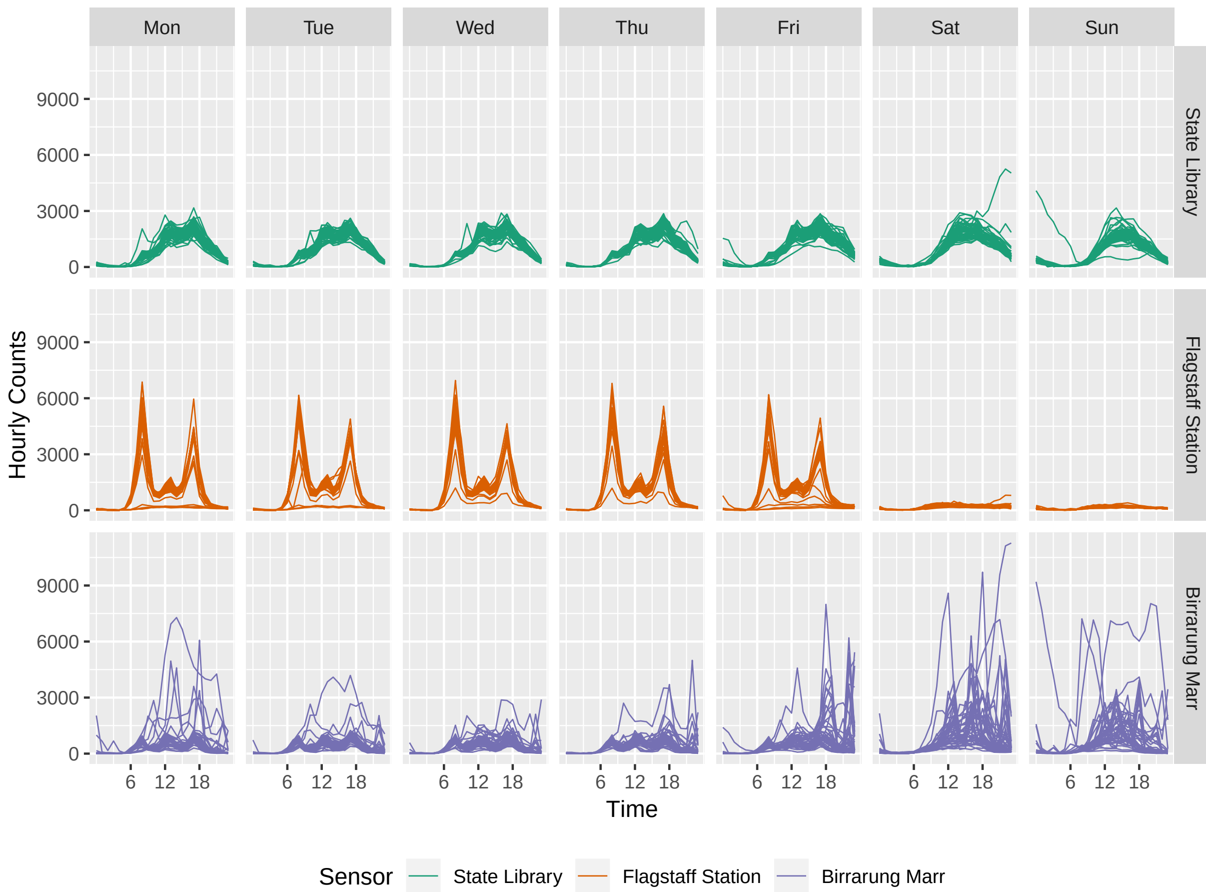

Figure 3: Hourly pedestrian counts for 2016 faceted by sensors and days of the week using lines. It features at least two types of seasons—time of day and day of week—across all the sensors. The White Night effect on increasing pedestrian counts can be seen on Saturday in February at the State Library.

The remainder of the paper is organized as follows. Section 2 demonstrates the construction of the calendar layout in depth. Section 3 lists and describes the options that come with the `frame_calendar` function. Section 4 presents some variations of its usage. Graphical analyses of sub-daily people's activities are illustrated with a case study in Section 5. Section 6 discusses the advantages and disadvantages of the method.

2 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:

- $k = 1, \dots, 7$ 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 $1 \le i \le 5$ is week within the month, $1 \le j \le 7$, is day of the week.
- $g = k, \dots, (k + d)$ indexes the day in the month, inside the 35 possible cells.

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

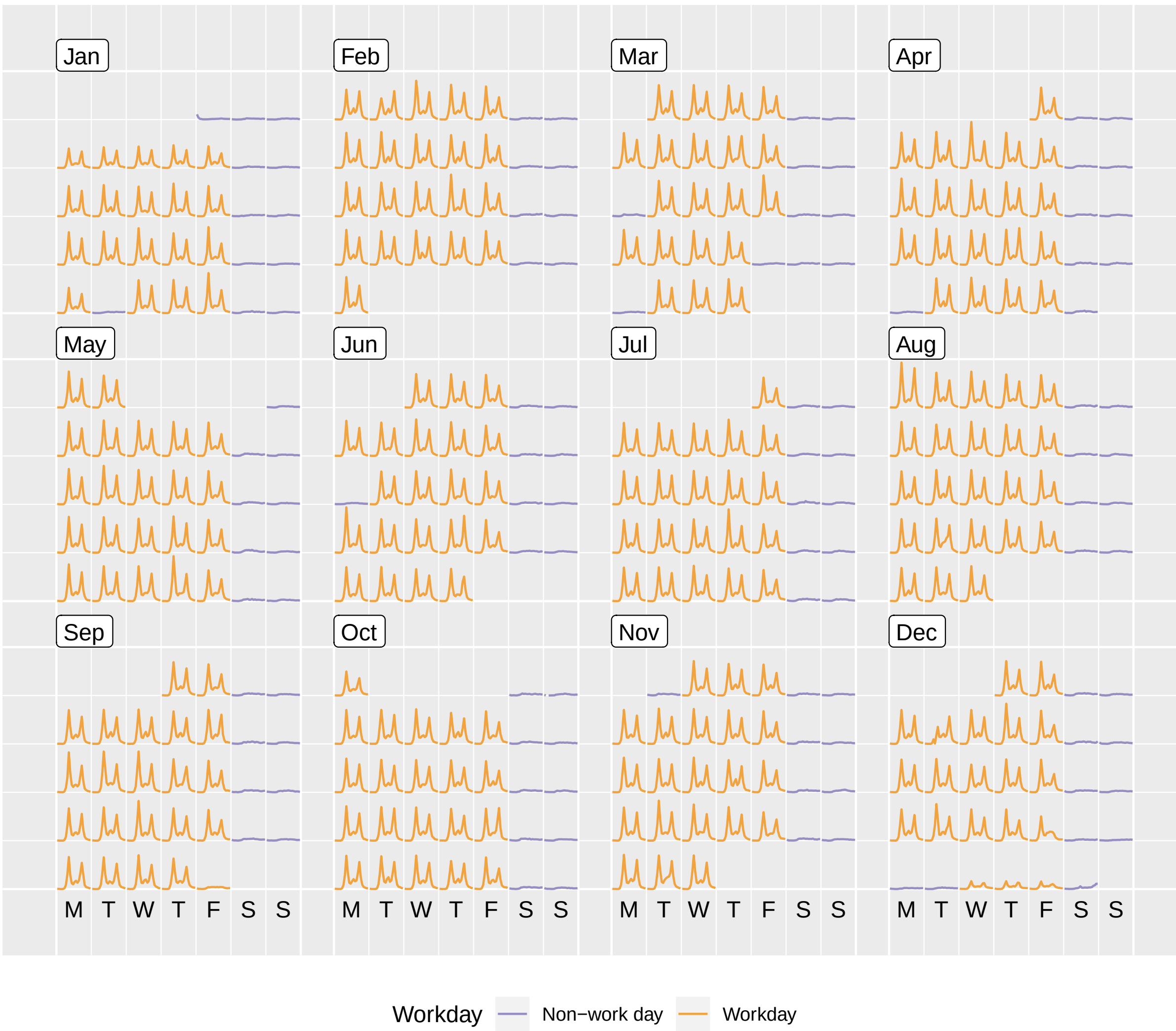

Figure 4: The calendar-based display of hourly foot traffic at Flagstaff Station using line glyphs. The arrangement of the data into a 3×4 monthly grid represents all the traffic in 2016. The disparities between week day and weekend along with public holiday are immediately apparent.

				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.

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

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:

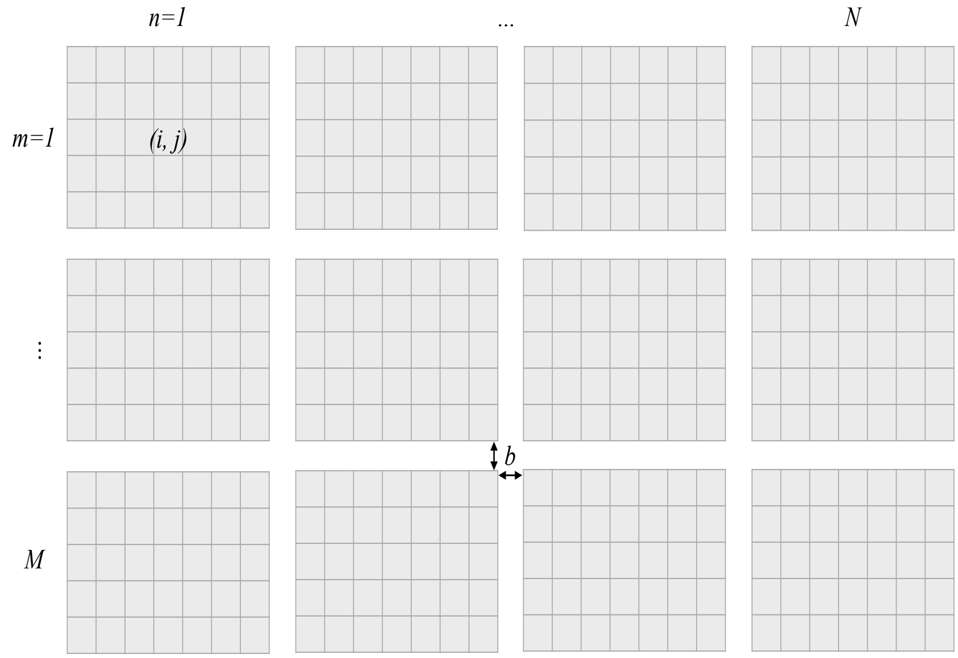

Figure 6: Illustration of the indexing layout for months of one year, where M and N indicate number of rows and columns, b is a space parameter separating cells.

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

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 $\min(y)$.

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.

3 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.

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

3.2 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.

3.3 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.

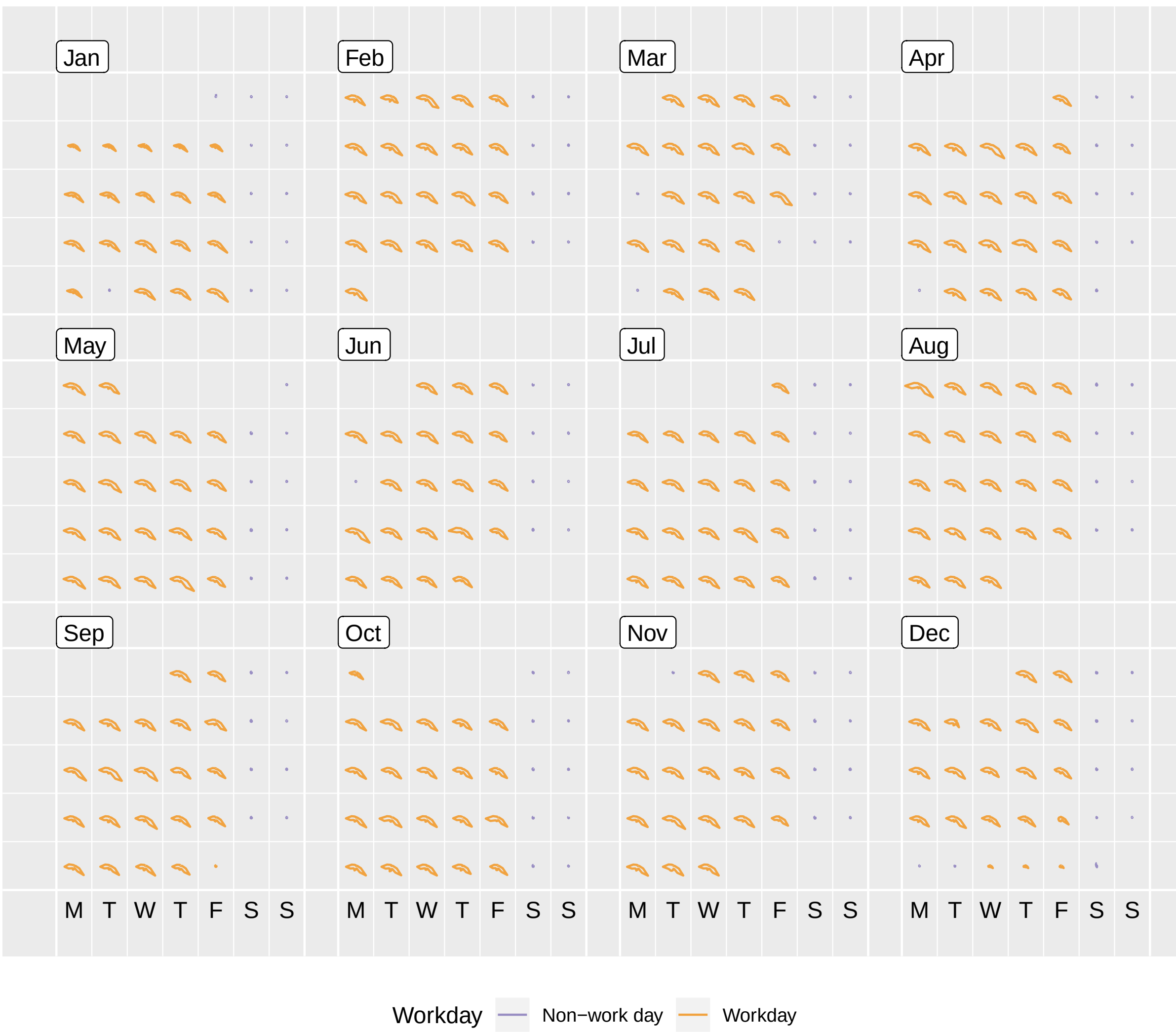

Figure 7: Figure 4 in circular layout, which is referred to as star plots. The daily periodicity on work days are clearly visible.

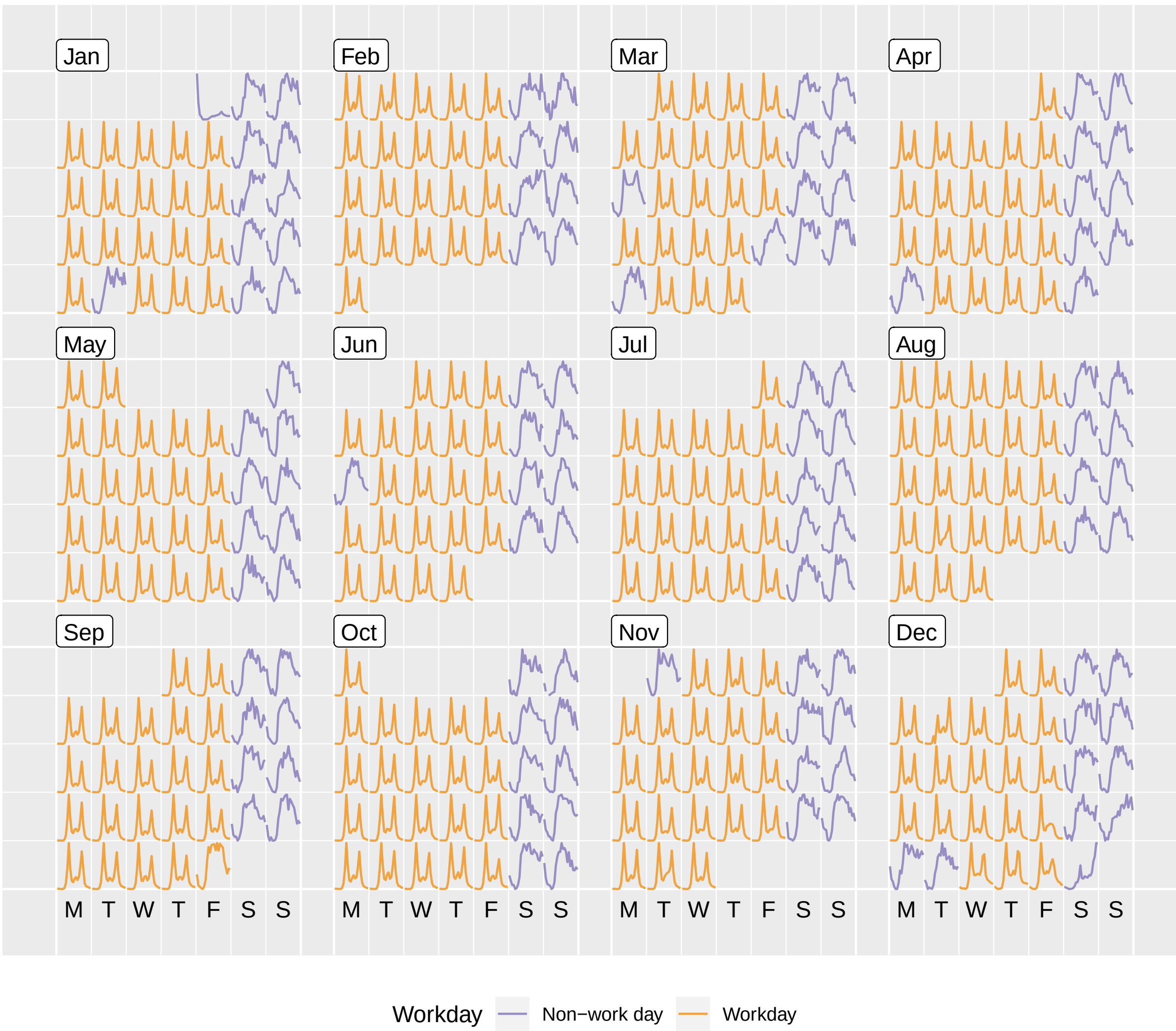

Figure 8: Line glyphs on the calendar format showing hourly foot traffic at Flagstaff Station, scaled over all the days. The individual shape on a single day becomes more distinctive, however it is impossible to compare the size of peaks between days.

3.4 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.

3.5 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.

4 Variations

4.1 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 at these sensors can be seen. Flagstaff Station exhibits strong commuters patterns, with fewer pedestrian counts during the weekends and public holidays. This suggests that Flagstaff Station has limited functionality on non-work days. From Figure 9 it can be seen that the State Library has a similar temporal trend to Flagstaff 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 Flagstaff 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.

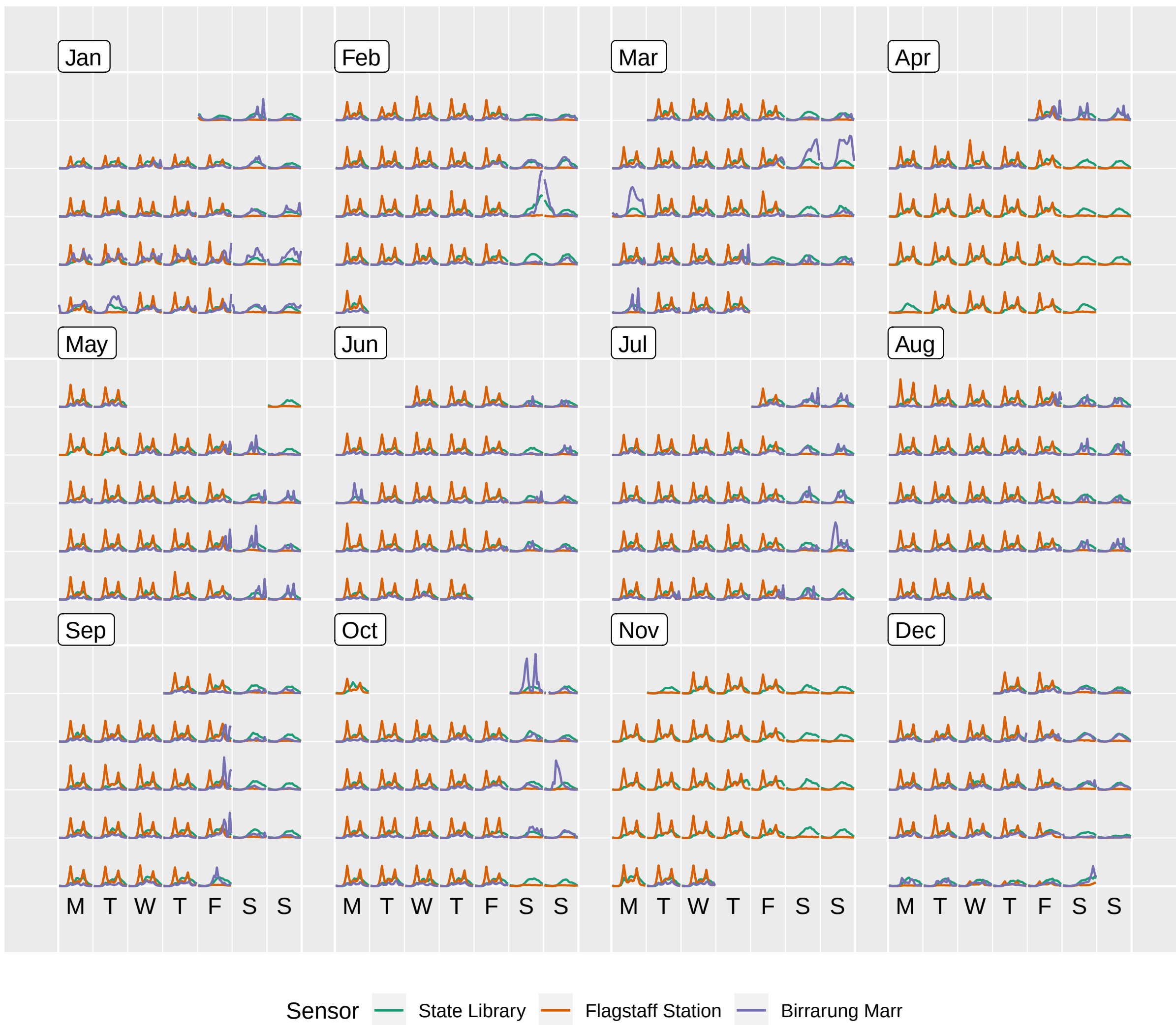

Figure 9: Overlaying line graphs of the 3 sensors in the monthly calendar. Flagstaff station is not as busy as the other two on non-work days.

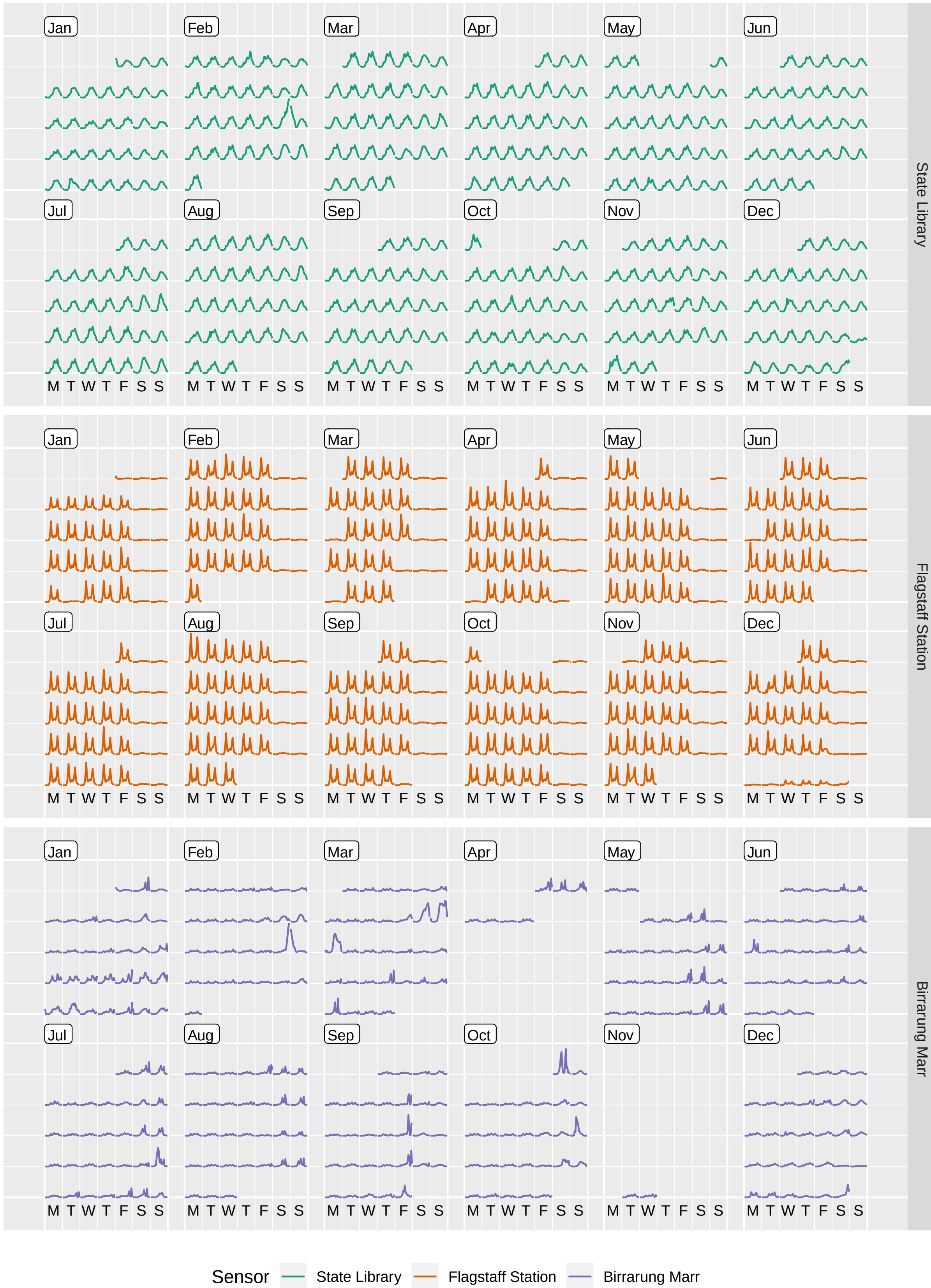

Figure 10: Line charts, embedded in the 6 × 2 monthly calendar, colored and faceted by the 3 sensors. The variations of an individual sensor are emphasised, and the shapes can be compared across the cells and sensors.

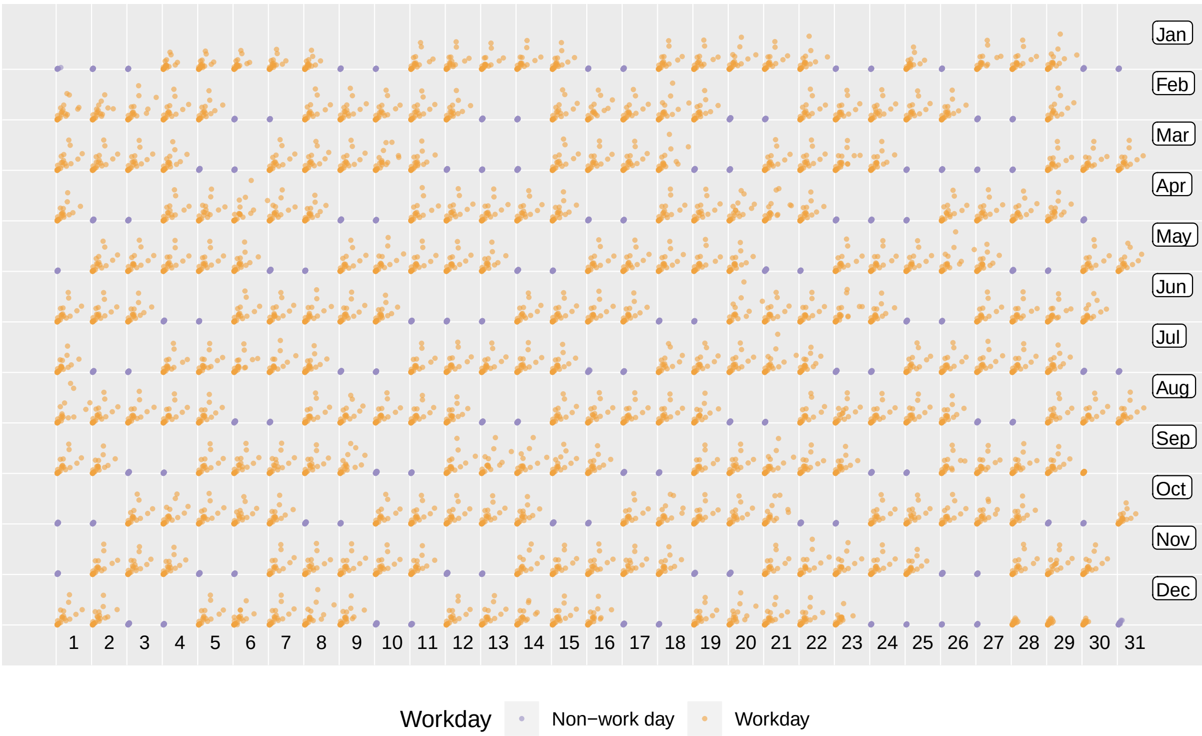

Figure 11: Lag scatterplot in the daily calendar layout. Each hour's count is plotted against previous hour's count at Flagstaff Station to demonstrate the autocorrelation at lag 1. The correlation between them is more consistent on non-work days than work days.

4.2 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 Flagstaff 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.

5 Case study

The case study uses individual-level energy usage from Australia. The dataset contains half-hourly electricity consumption data of 4 households over different time periods, with 64,128 observations. The goal is to sketch typical profiles as well as unusual patterns in order to better understand each household's daily schedules.

Firstly, a letter value plot (Hofmann et al. 2017) shown as Figure 14 helps grasp an overview of the variability between households across days of week. Household 2 has seen a great deal of variability in daily energy consumption, depicting relatively large variations on Thursdays. By contrast, household 4 consumes much less energy with constant variations. The dark blue line (representing median) highlights more energy consumed by households 1, 2, and 4 over the weekends relative to weekdays.

Then a snapshot is taken at a different time resolution to picture individual's energy usage against time of day, across workdays and non-work days. Figure 15 gives a more detailed depiction of each household's life style. Household 1 appears an early bird, starting the day before 6 and going back home around 18 on workdays. There is an uncommon pattern occurring to household 2 only, with a cluster of points concentrated above 1 kWh across all time of day. It perhaps alerts household 2 to potential electricity leakage. Household 3 shows persistent trimodel behaviors across workdays and non-workdays. Nevertheless, the non-work life of household 4 distinguishes from the workdays when most energy is consumed after 18.

Figure 14 and 15, part of traditional graphical toolkits, are useful for summarizing overall deviations across days and households. Is there any untold story from these plots? Figure 16 lays out the data points at the finest time resolution in a 3×4 monthly grid, for household 1 and 3. Household 1 was on vacation over three weeks of June, and household 2 was also away for holidays from September to early October and late June to mid July in the past year. This calendar plot evidently fills the missing piece of the picture.

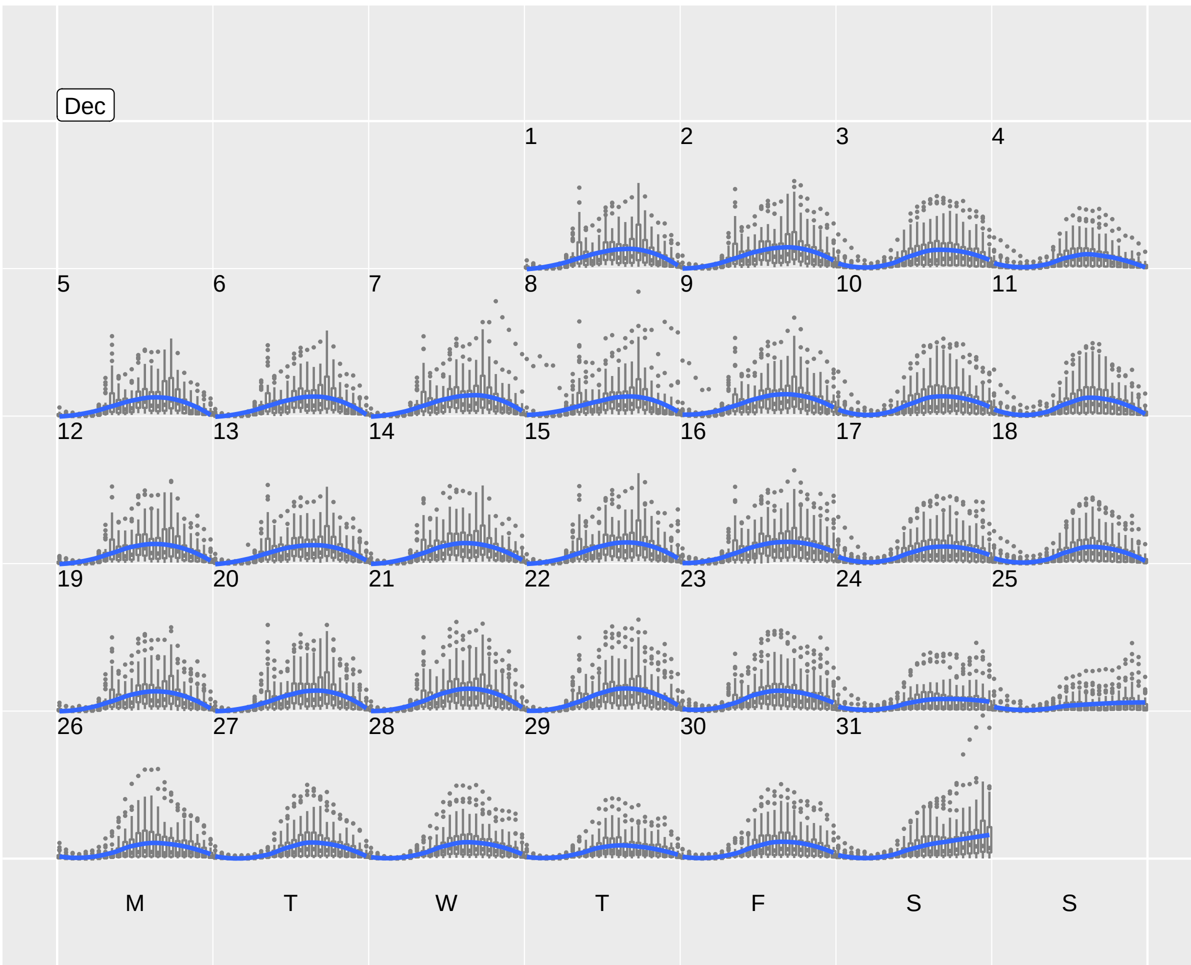

Figure 12: Side-by-side boxplots of hourly counts for all the 43 sensors in December 2016, with the loess smooth line superimposed on each day. It shows the hourly distribution in the city as a whole. There is one sensor attracting a larger number of people on New Year's Eve than the rest.

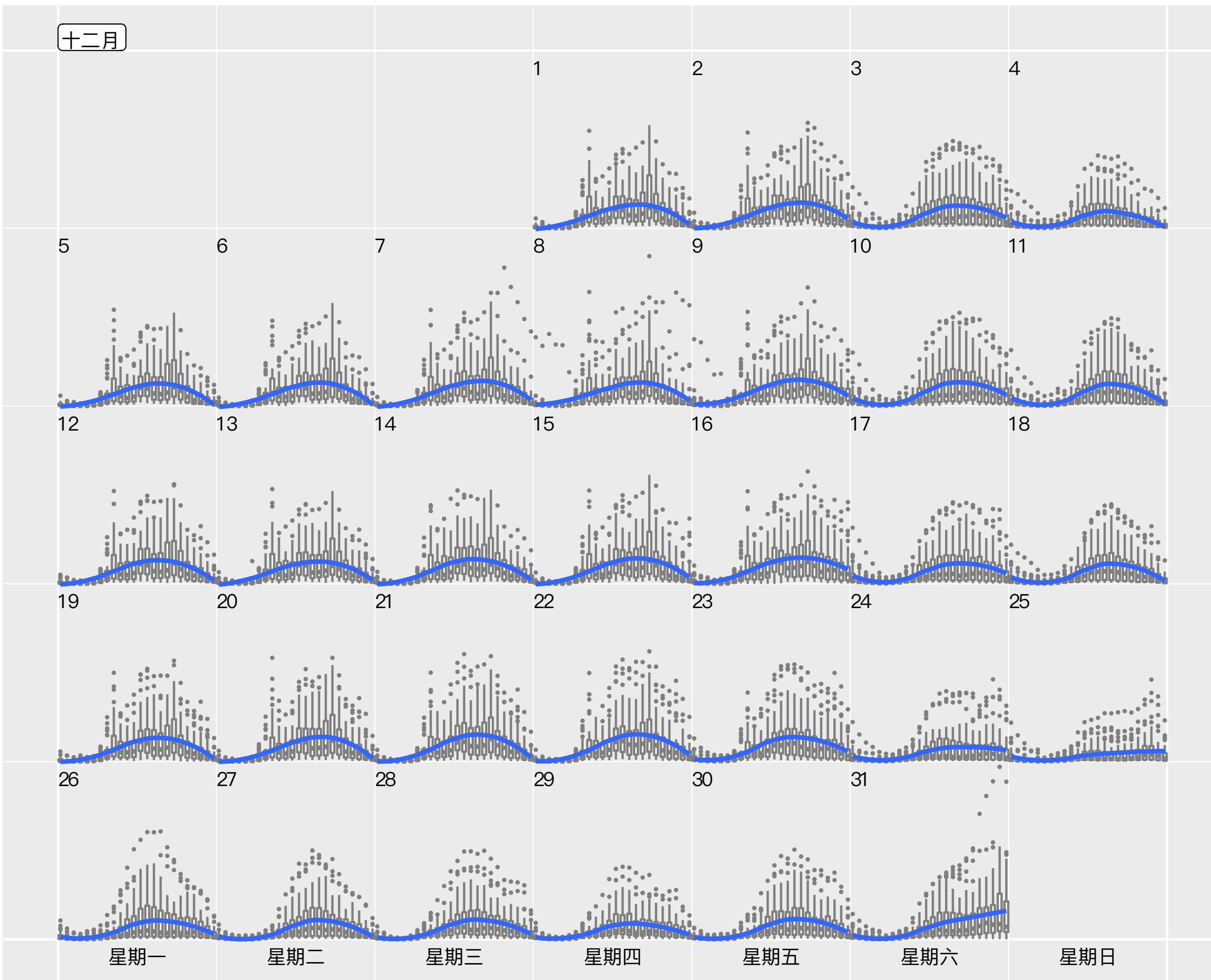

Figure 13: The same plot as Figure 12, but with the month and week day labels in Chinese. It demonstrates the natural support for languages other than English.

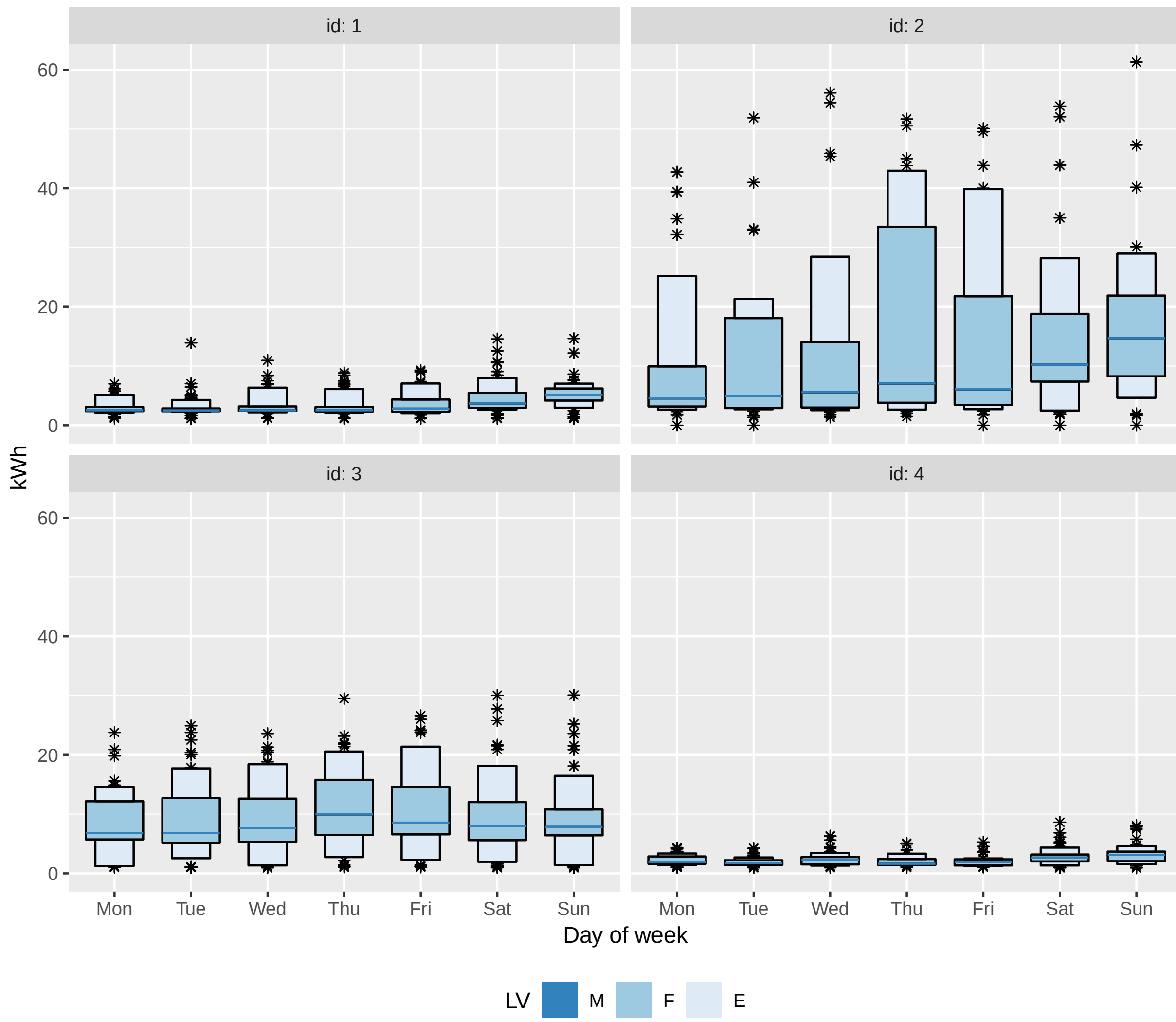

Figure 14: Letter value plot of daily energy usage against day of week by households. The enery consumed by household 2 exhibits much larger variability, compared to the rest households. Households 1, 2, and 4 use mose energy over the weekends.

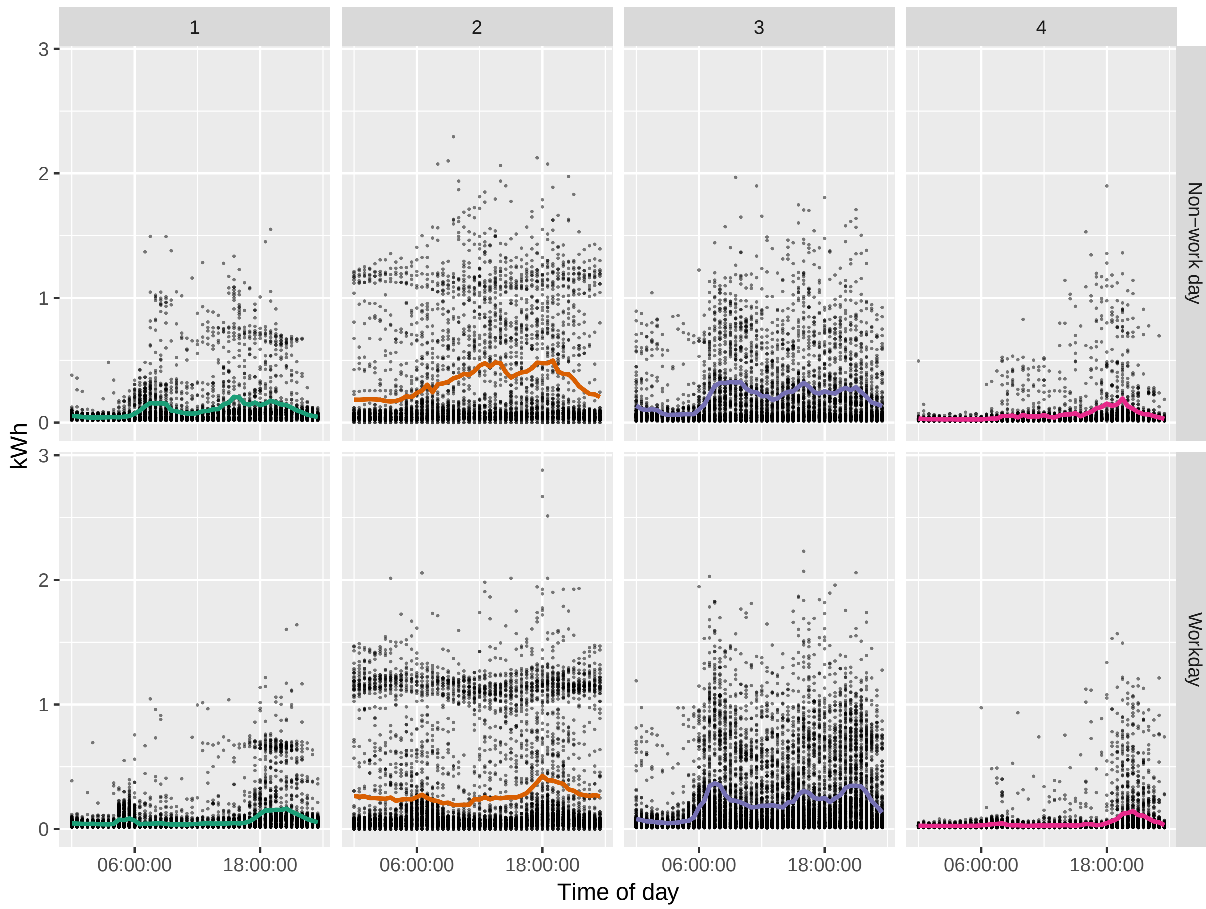

Figure 15: Scatterplot between time of day and half-hourly energy usage, constrasting workdays and non-work days for each household. A distinctive pattern exclusive to household 2 is popped out: a fair amount of points are clustered above 1 kWh over all time of day.

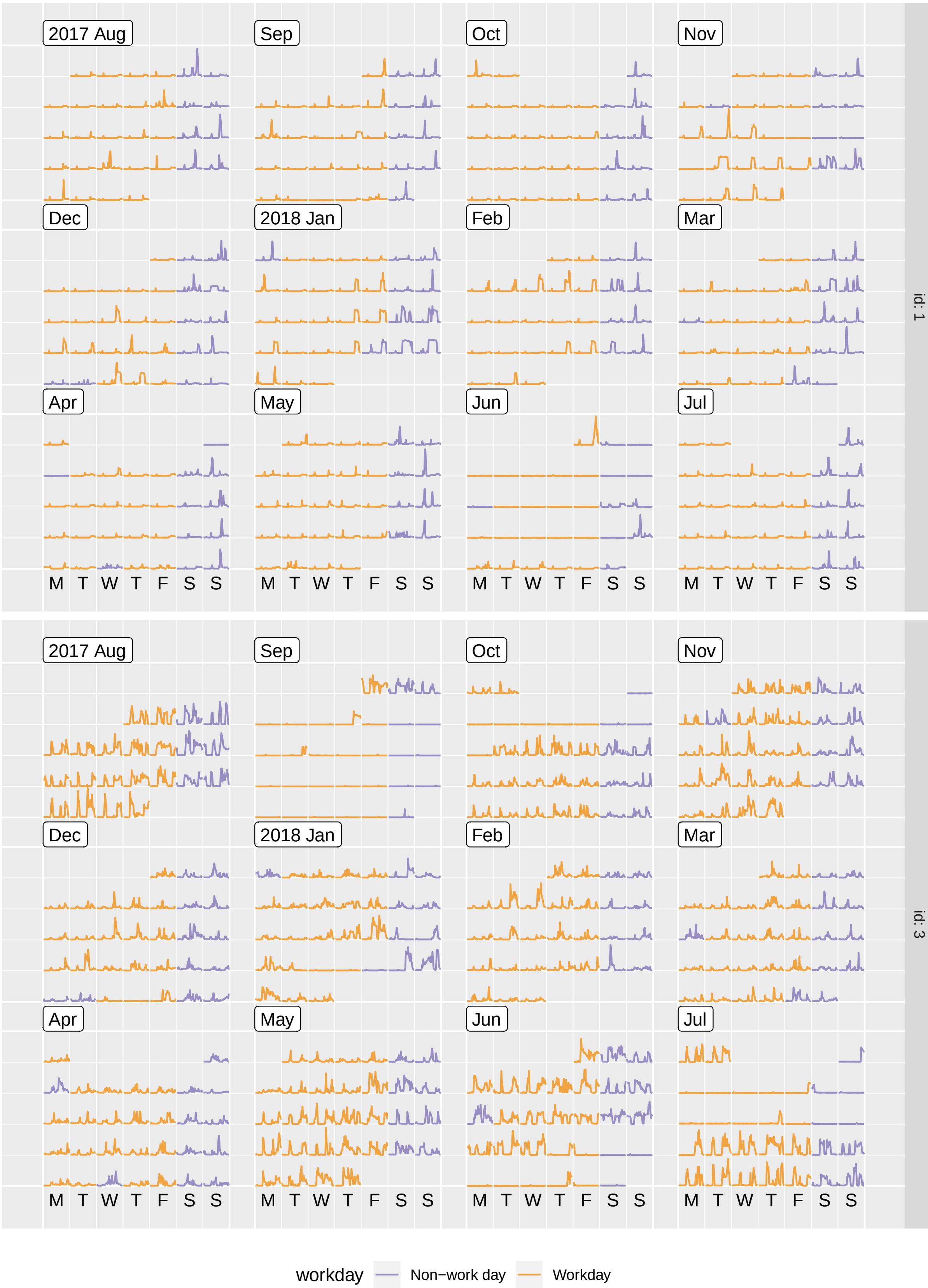

Figure 16: Arranging sub-daily energy usage into monthly grids reveals the vacation time periods for households 1 and 3, which are hidden in Figure 14 and 15.

6 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.

Acknowledgements

We would like to thank Stuart Lee and Heike Hofmann for their feedback about this work. The most recent version of the `frame_calendar` function is included in the **sugrrants** package, which can be accessed via the CRAN website `https://CRAN.R-project.org/package=sugrrants` or Github `https://github.com/earowang/sugrrants`. All materials required to reproduce this article and a history of the changes can be found at the project's Github repository `https://github.com/earowang/paper-calendar-vis`.

References

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

Cleveland, W. S. & McGill, R. (1984), 'Graphical perception: Theory, experimentation, and application to the development of graphical methods', *Journal of the American Statistical Association* **79**(387), 531–554.

Hafen, R. (2018), *geofacet: 'ggplot2' Faceting Utilities for Geographical Data.* R package version 0.1.9.
URL: *https://CRAN.R-project.org/package=geofacet*

Hofmann, H., Wickham, H. & Kafadar, K. (2017), 'Letter-value plots: Boxplots for large data', *Journal of Computational and Graphical Statistics* **26**(3), 469–477.

Jacobs, J. (2017), ***ggcal**: Calendar Plot Using **ggplot2**.* R package version 0.1.0.
URL: *https://github.com/jayjacobs/ggcal*

Kothari, A. & Ather (2016), ***ggTimeSeries**: Nicer Time Series Visualisations with ggplot syntax.* R package version 0.1.
URL: *https://github.com/Ather-Energy/ggTimeSeries*

Lam, H., Munzner, T. & Kincaid, R. (2007), 'Overview use in multiple visual information resolution interfaces', *IEEE Transactions on Visualization and Computer Graphics* **13**(6), 1278–1285.

Van Wijk, J. J. & Van Selow, E. R. (1999), Cluster and calendar based visualization of time series data, *in* 'Information Visualization, 1999. INFOVIS 1999 Proceedings. IEEE Symposium on', IEEE, pp. 4–9.

Wang, E., Cook, D. & Hyndman, R. (2018), *sugrrants: Supporting Graphs for Analysing Time Series.* R package version 0.1.6.
URL: *https://pkg.earo.me/sugrrants*

Wickham, H. (2009), ***ggplot2**: Elegant Graphics for Data Analysis*, Springer-Verlag New York, New York, NY.

Wickham, H. (2014), 'Tidy data', *Journal of Statistical Software* **59**(10), 1–23.

Wickham, H. (2017), *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.2.1.
URL: *https://CRAN.R-project.org/package=tidyverse*

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C. & Woo, K. (2018), *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. http://ggplot2.tidyverse.org, https://github.com/tidyverse/ggplot2.

Wickham, H., Hofmann, H., Wickham, C. & Cook, D. (2012), 'Glyph-maps for visually exploring temporal patterns in climate data and models', *Environmetrics* **23**(5), 382–393.

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