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

May 8, 2019

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 functions `frame_calendar()` and `facet_calendar()` use 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 integrated into a data pipeline. The core component of the pipeline is to visualise the resulting data using the grammar of graphics (Wilkinson 2005, Wickham 2009), as used in **ggplot2** (Wickham et al. 2019), where plots are defined as a functional mapping from data points to graphical primitives. 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 R package called **sugrrants** (Wang et al. 2019).

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 inner-city area until the end of 2016 (Figure 1). The data set 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 three 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. Faceting generates small multiples (Tufte 1983) or trellis displays (Becker et al. 1996), where different subsets of the same data are displayed based on conditioning variables, allowing for comparison across subsets. A series of graphics can be further grouped together to create multiple yet different data views, that forms ensemble graphics (Unwin & Valero-Mora 2018).

However, the conventional displays of time series data conceal patterns relative to special events (such as public holidays and recurring cultural/sport events), which may be worth noting 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 (2019)

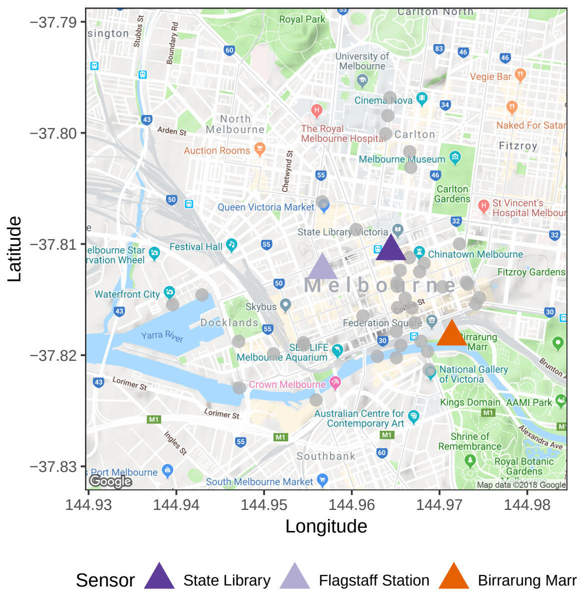


Figure 1: Map of the Melbourne city area with dots indicating sensor locations. These three highlighted sensors will be inspected in the paper: (1) the State Library—a public library, (2) Flagstaff Station—a train station, closed on non-work days, (3) Birrarung Marr—an outdoor park hosting many cultural and sports events.

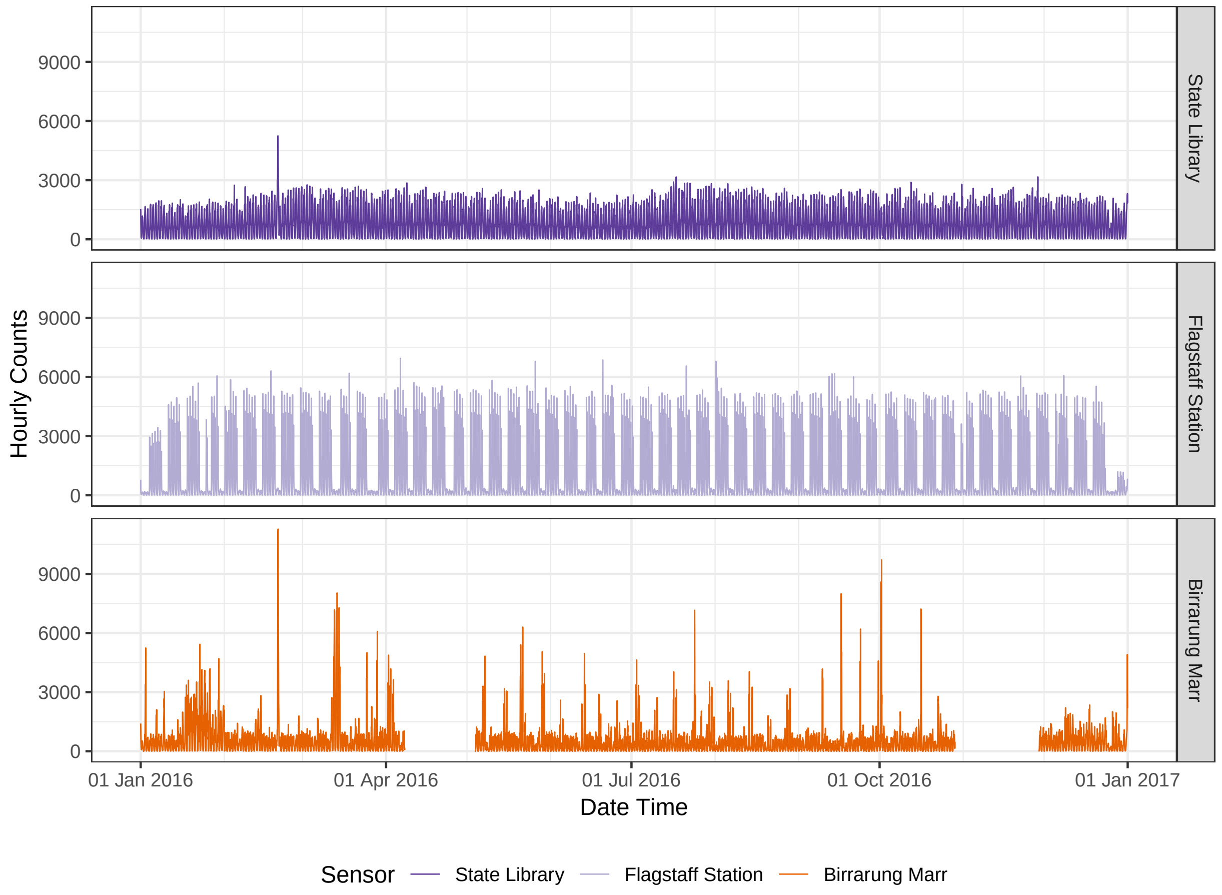


Figure 2: Time series plots showing the number of pedestrians in 2016 measured at three different sensors in the city of Melbourne. Colored by the sensors, small multiples of lines show that the foot traffic varies from one sensor to another in terms of both time and number. A spike occurred at the State Library, caused by the annual White Night event on 20th of February. A relatively persistent pattern repeats from one week to another at Flagstaff Station. Birrarung Marr looks rather noisy and spiky, with a couple of chunks of missing records.

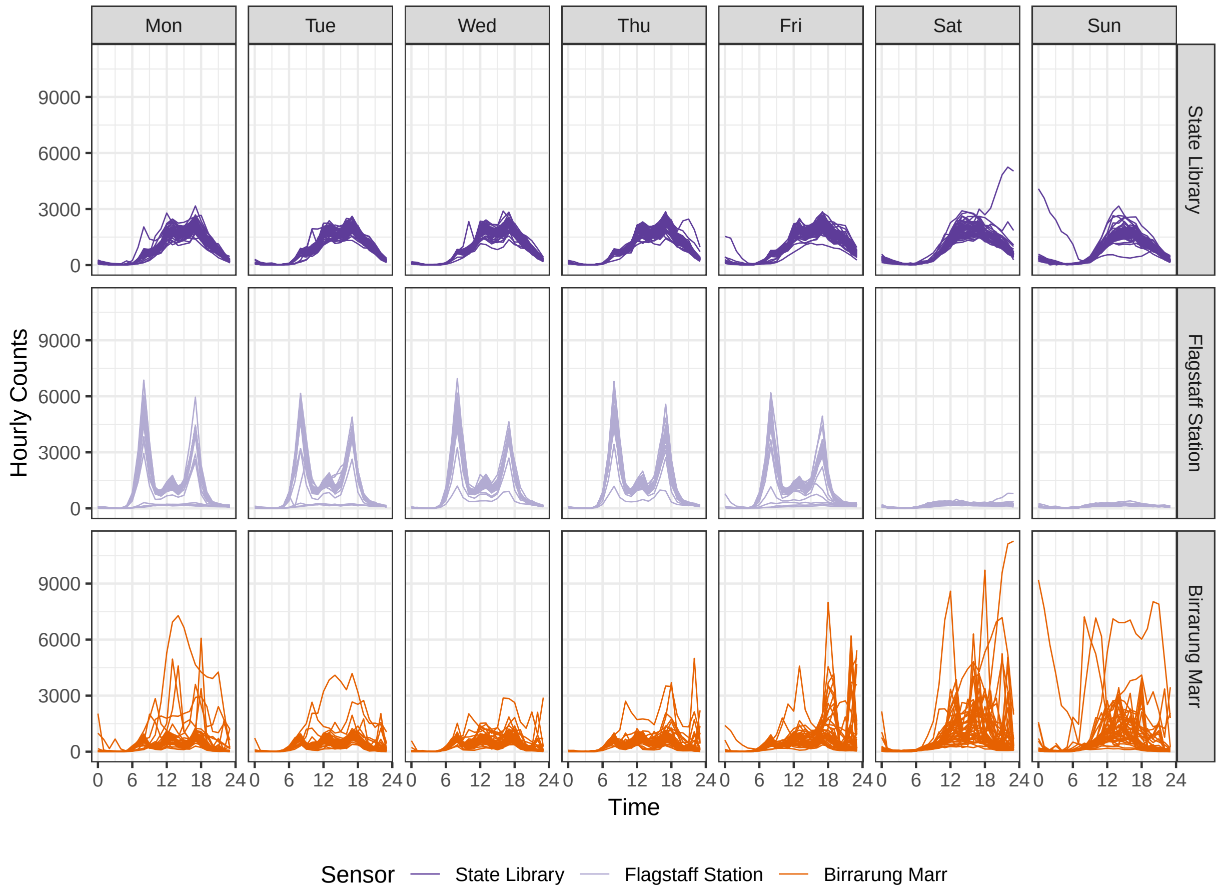


Figure 3: Hourly pedestrian counts for 2016 faceted by sensors and days of the week using lines. It primarily features two types of seasons—time of day and day of week—across all the sensors. Apparently other factors have influence over the number of pedestrians, which cannot be captured by the faceted plots, such as the overnight White Night traffic on Saturday at the State Library and a variety of events at Birrarung Marr.

which provides methods in the **geofacet** R 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 are some existing works 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, 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. Alongside Jones (2016), Wong (2013), Kothari & Ather (2016), and Jacobs (2017) implemented some variants of calendar-based heatmaps as in R packages: **TimeProjection**, **ggTimeSeries**, and **ggcal** respectively. 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 patterns related to special events more easily pop-up to viewers. The proposed algorithm has been implemented in the `frame_calendar()` and `facet_calendar()` functions in the **sugrrants** package using R.

The remainder of the paper is organized as follows. Section 2 demonstrates the construction of the calendar layout in depth. Section 2.1 describes the algorithms of data transformation. Section 2.2 lists and describes the options that come with the `frame_calendar()` function. Section 2.3 presents some variations of its usage. The calendar layout is a type of faceting, and Section 3 extends this into a fully-fledged faceting method as in `facet_calendar()` with formal labels and axes. Graphical analyses of sub-daily people's activities are illustrated with a case study in Section 4. Section 5 discusses the limitations of calendar displays and possible new directions.

2 Creating a calendar display

2.1 Data transformation

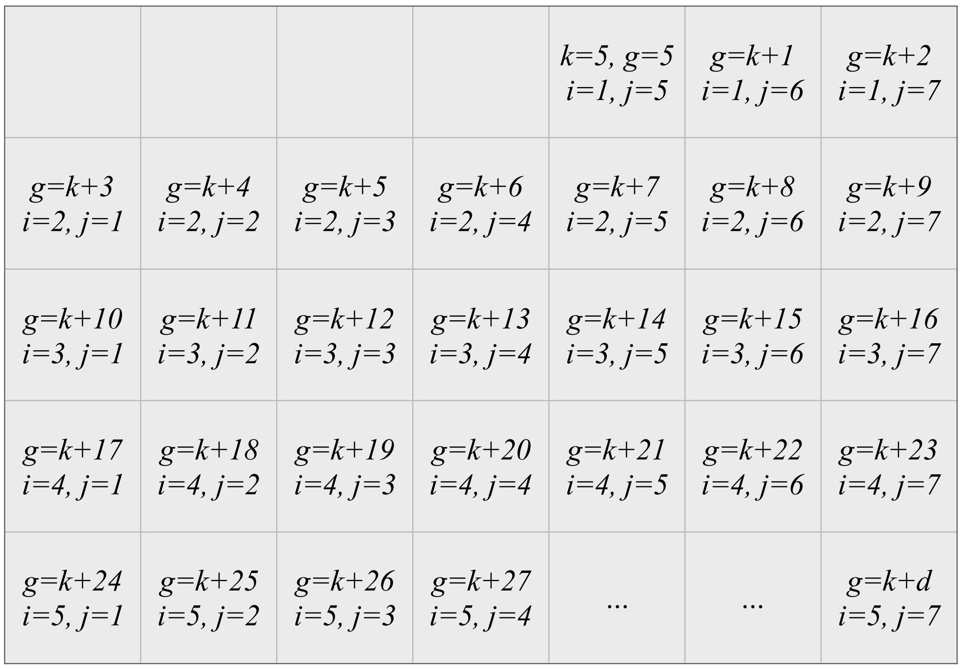
The algorithm of transforming data for constructing a calendar plot uses linear algebra, similar to that used in the glyph map displays for spatio-temporal data (Wickham et al. 2012). To make a year long calendar requires cells for days, embedded in blocks corresponding to months, organized into a grid layout for a year. Each month conforms to a layout of 5 rows and 7 columns, 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 some months could extend over six weeks but for these months we intentionally wrap the last few days up to the top row of the block for the purpose of compact displays. 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

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

Figure 4 illustrates this (i, j) layout for a month where $k = 5$.



				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 4: 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.

To create the layout for a full year, (m, n) denotes the position of the month arranged in the plot, where $1 \le m \le M$ and $1 \le n \le N$; b denotes the small amount of white space between each month for visual separation. Figure 5 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

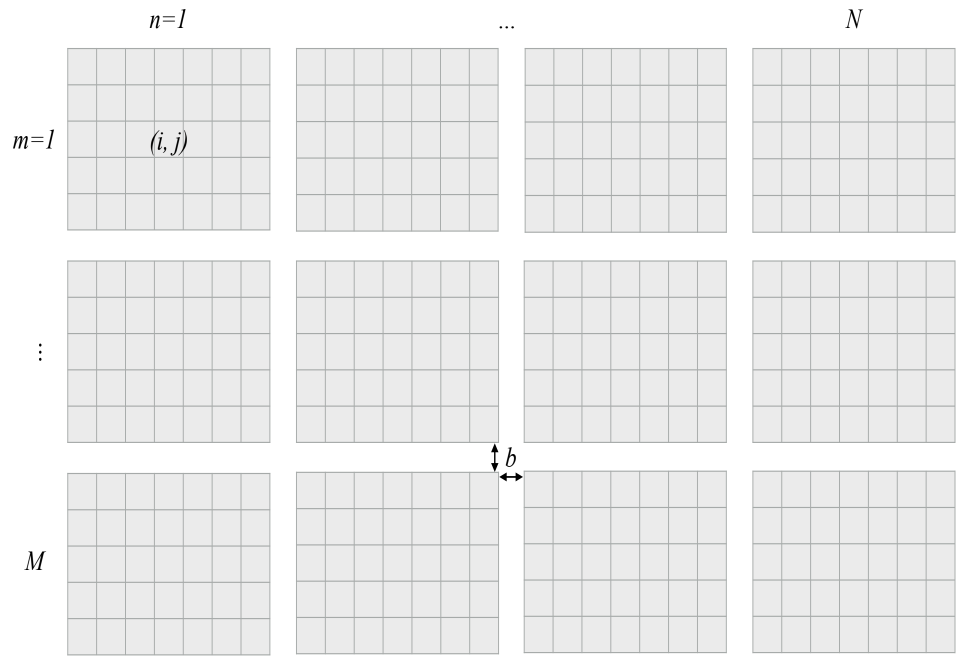


Figure 5: Illustration of the indexing layout for months of one year, where M and N indicate number of rows and columns, b is a space parameter separating cells.

Figure 4) which is why subtraction is performed. Within each cell, the starting position is the bottom left.

Figure 6 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** R package (Wickham et al. 2019) functions. All of the grammar of graphics can be applied.

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 6, 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

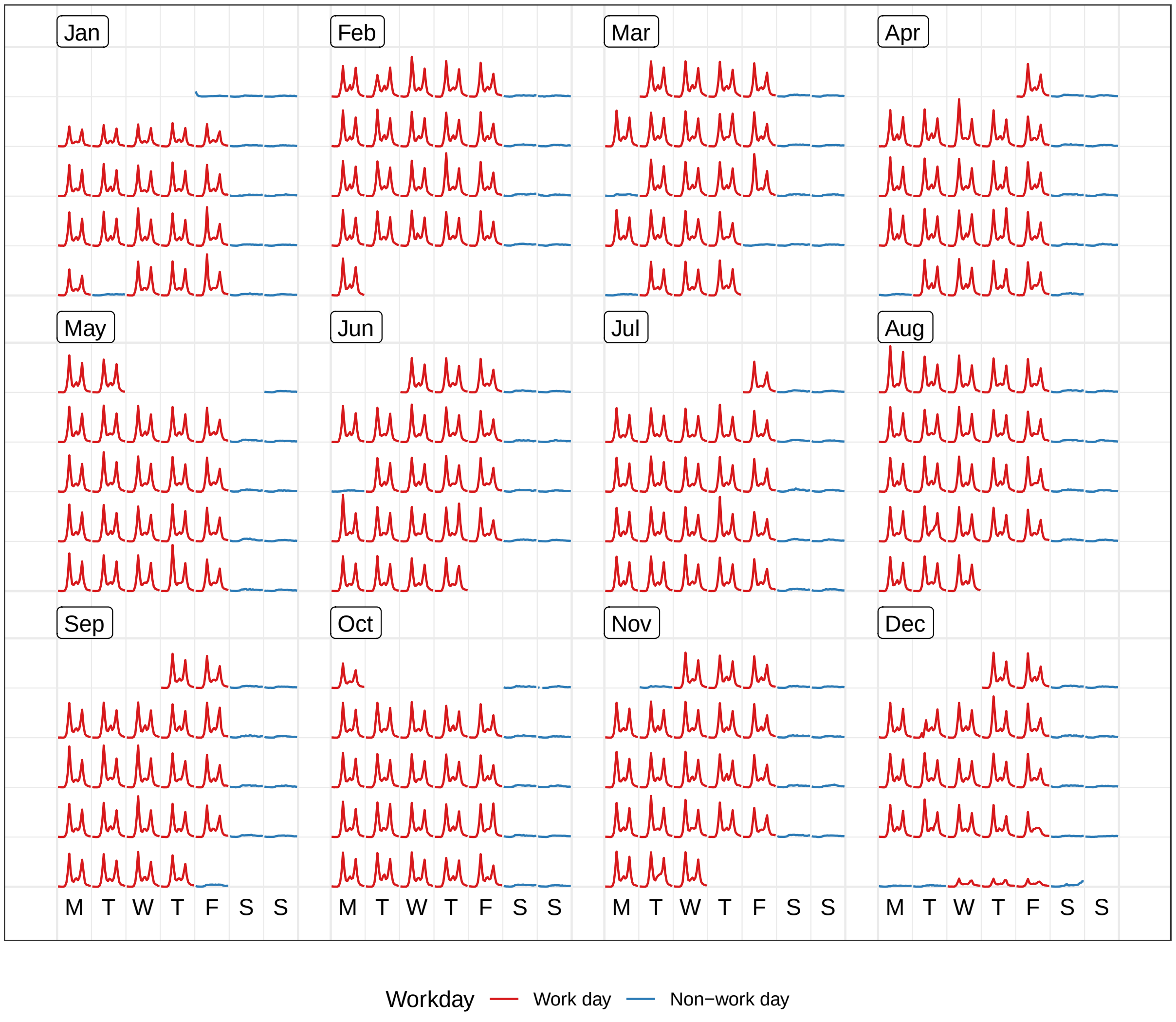


Figure 6: The calendar-based display of hourly foot traffic at Flagstaff Station using line glyphs. The disparities between week day and weekend along with public holiday are immediately apparent. The arrangement of the data into a 3×4 monthly grid represents all the traffic in 2016. Note that the algorithm wraps the last few days in the sixth week to the top row of each month block for a compact layout, which occurs to May and October.

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.

2.2 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",
  week_start = 1, 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 6). 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.

2.2.1 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.

2.2.2 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 Equation (1). This can be useful for creating calendar layouts for countries where vertical layout is the convention.

2.2.3 Start of the week

The start of the week for a monthly calendar is adjustable. The default is Monday (`1`), which is chosen from the data perspective. The week, however, can begin with Sunday (`7`) as commonly used in the US and Canada, or other week day subject to different countries and cultures.

2.2.4 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). The resulting plot is star glyphs (Chambers et al. 2017), where time series lines are transformed in polar coordinates, embedded in the monthly calendar layout.

2.2.5 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 7 shows the same data as Figure 6 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.

2.2.6 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 language specifications other than English for text labelling.

2.3 Variations

2.3.1 Overlaying and faceting subsets

Plots can be layered. The comparison of sensors can be done by overlaying the values for each (Figure 8). 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 8 it can be seen that Birrarung Marr has a distinct temporal pattern from the other two all year round. The nighttime events, such as White Night, have barely affected the operation of Flagstaff Station but heavily affected the incoming and outgoing traffic to the State Library and Birrarung Marr.

To avoid the overlapping problem, the calendar layout can be embedded into a series of subplots for the different sensors. Figure 9 presents the idea of faceting calendar plots by sensors. This allows comparing the overall structure between sensors, while emphasizing individual sensor variation. In particular, it can be immediately learned that Birrarung Marr was busy and packed, for example during the Australian Open, a major international tennis tournament, in the last two weeks of January. This is concealed in the conventional graphics.

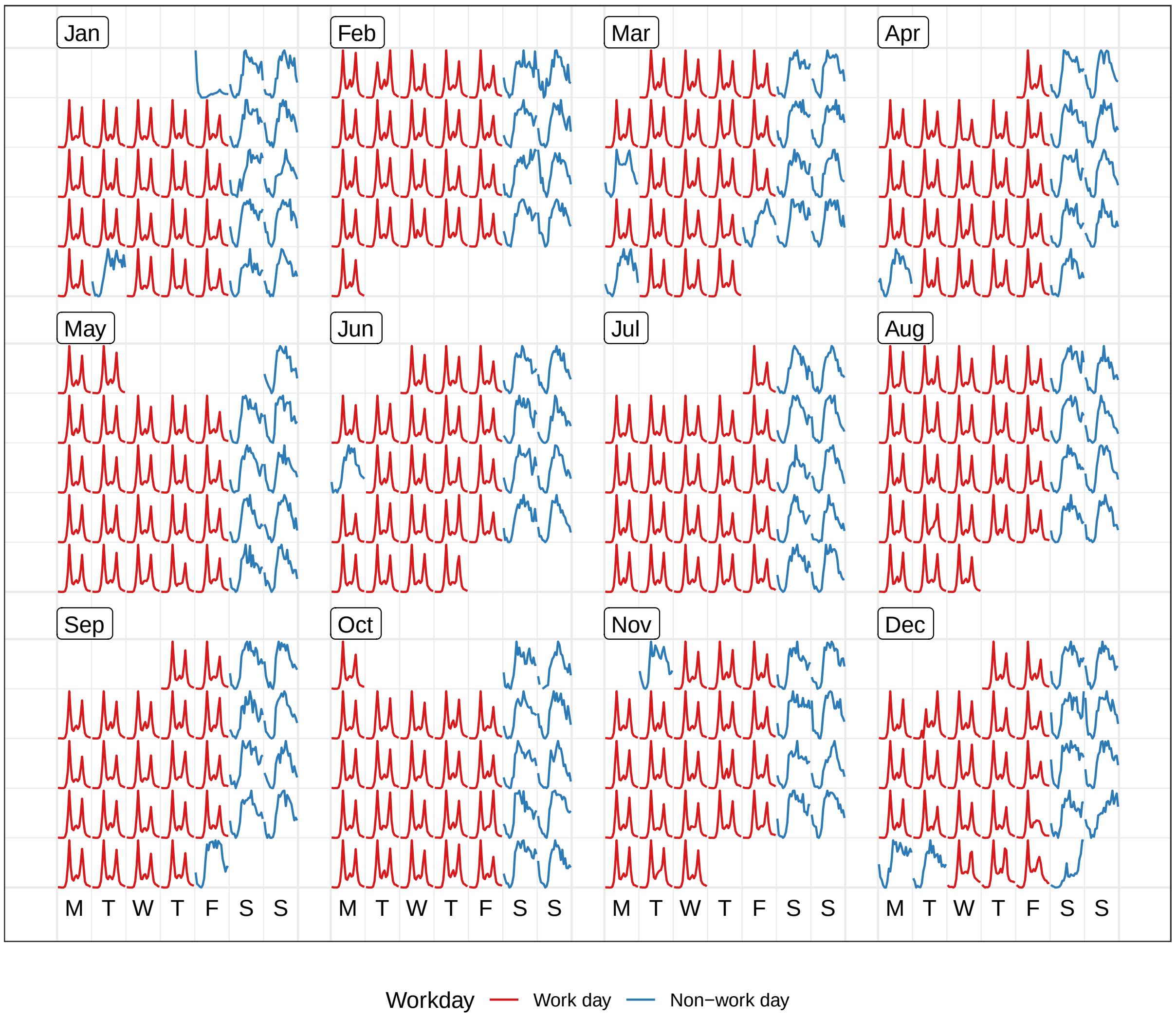


Figure 7: Line glyphs on the calendar format showing hourly foot traffic at Flagstaff Station, scaled over all the days. The individual shape on a single day becomes more distinctive, however it is impossible to compare the size of peaks between days.

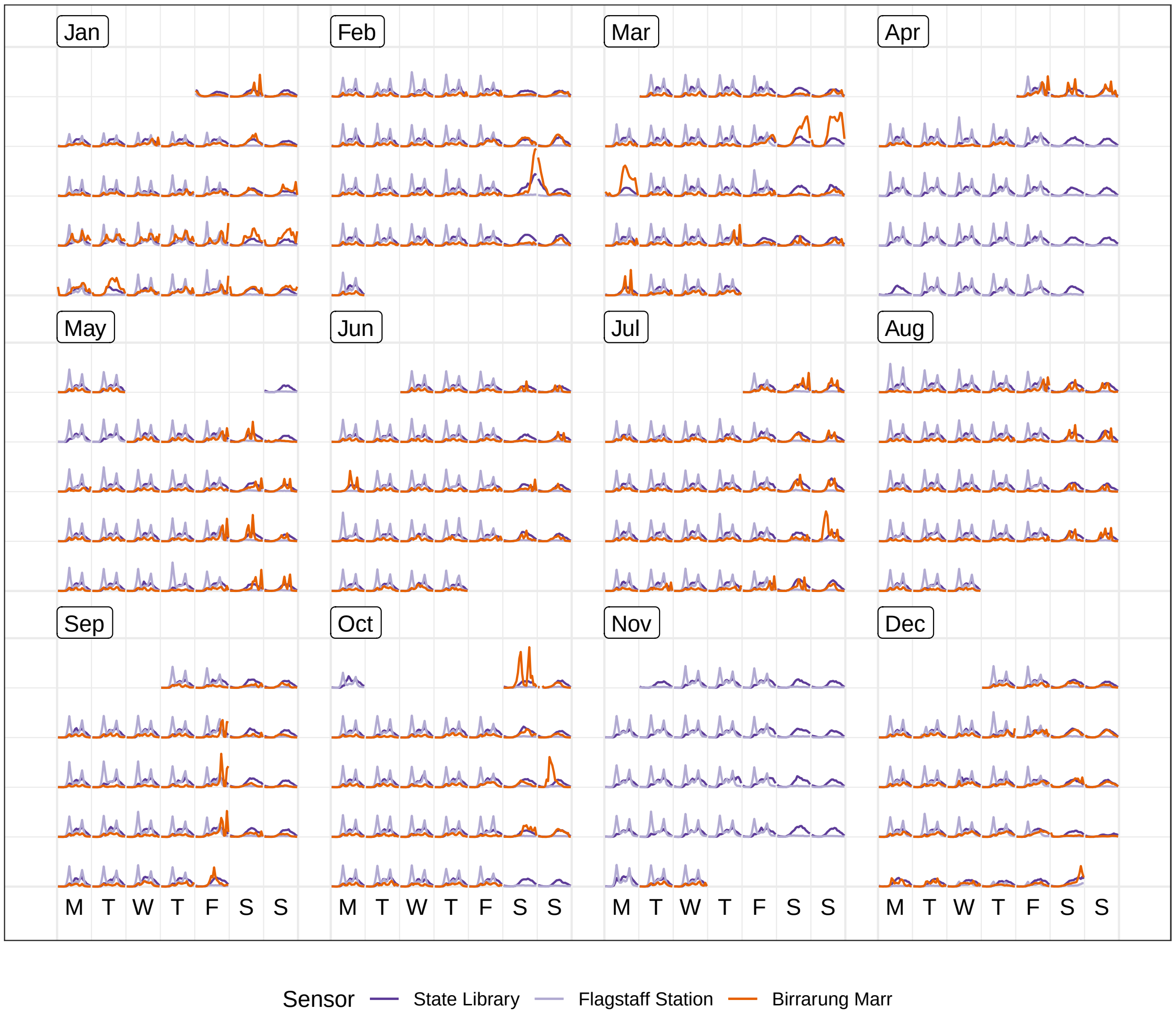


Figure 8: Overlaying line graphs of the three sensors in the monthly calendar. Three sensors demonstrate very different traffic patterns. Birrarung Marr tends to attract many pedestrians for special events held on weekends, contrasting to the bimodal commuting traffic at Flagstaff Station.

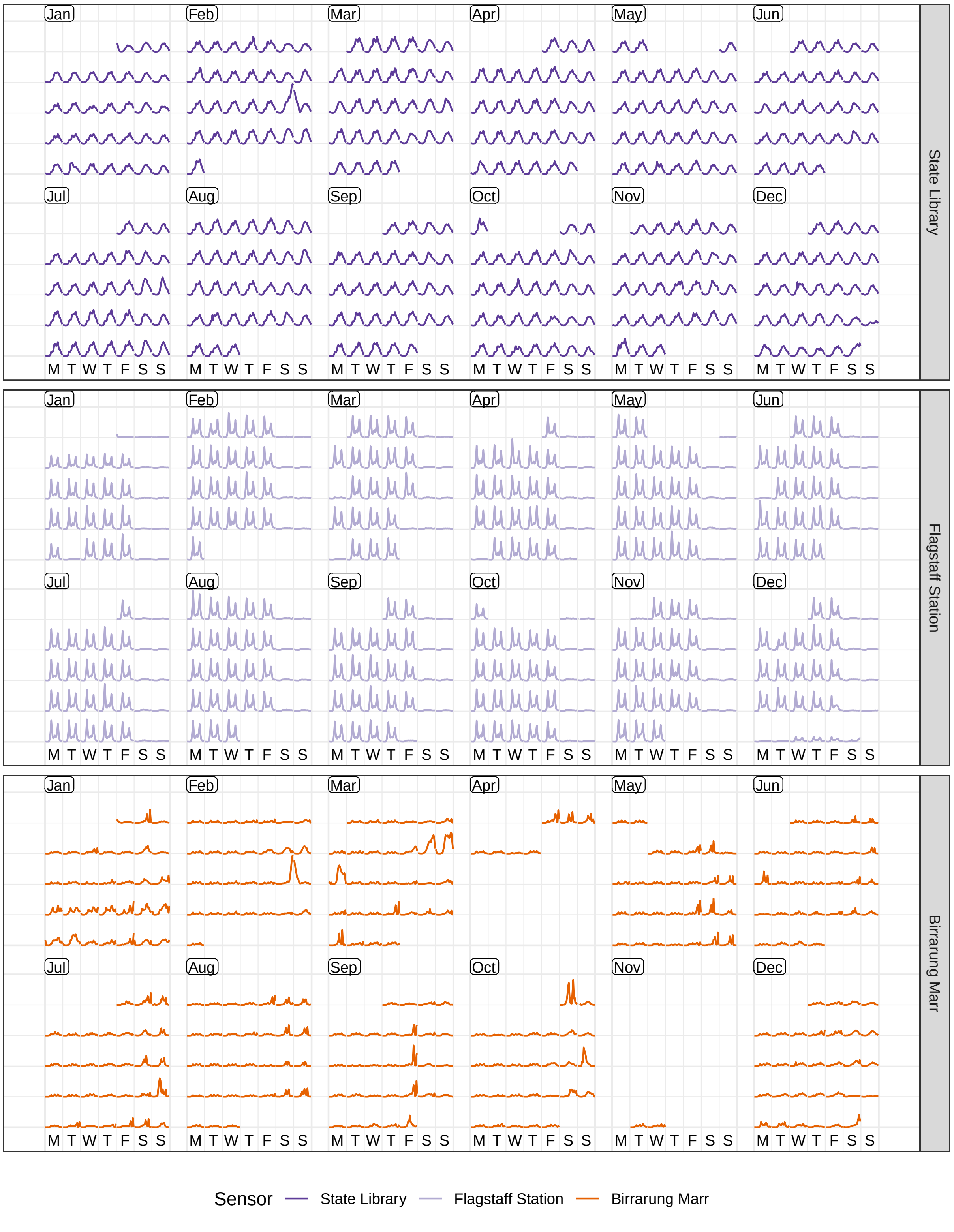


Figure 9: Line charts, embedded in the 6×2 monthly calendar, colored and faceted by the 3 sensors. The variations of an individual sensor are emphasised, and the shapes can be compared across the cells and sensors.

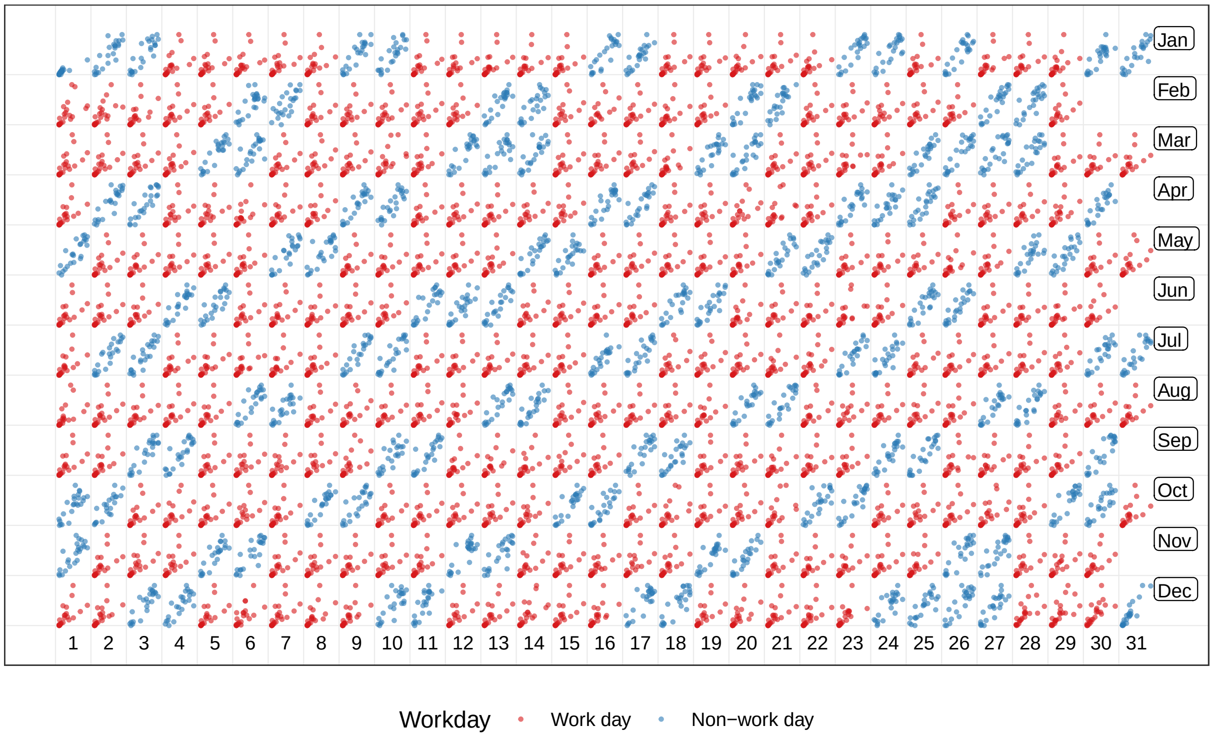


Figure 10: Lag scatterplot with local scaling in the daily calendar layout. Each hour's count is plotted against previous hour's count at Flagstaff Station to demonstrate the autocorrelation at lag 1. The correlation between them is more consistent on non-work days than work days.

2.3.2 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 10 shows a lag scatterplot with local scaling for each day 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 10 indicates two dominate 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, as is depicted as the V-shape in the graphs. This is consistent with the bimodality from Figure 6, since positive correlations are perceived when approaching to the peak hour yet negative correlations when moving away from the peak hour.

The algorithm can also produce more complicated plots, such as boxplots. Figure 11 uses a loess smooth line (Cleveland 1979) 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 (December 24), go shopping on the Boxing day (December 26), and stay out for the fireworks on New Year's Eve. The text in the plot is labeled in Chinese, showcasing the support for other languages.

2.3.3 Interactivity

As a data restructuring tool, the interactivity of calendar-based displays can be easily enabled, as long as the interactive graphic system remains true to the spirit of the grammar of graphics, for example **plotly** (Sievert 2018) in R. As a standalone display, an interactive tooltip can be added to show labels when mousing over it in the calendar layout, for example the hourly count with the time of day. It is difficult to sense the values from the static display, but the tooltip makes it possible. Options in the `frame_calendar()` function can be ported to a form of selection button or text input in a graphical user interface like R shiny (Chang et al. 2019). The display will update on the fly accordingly via clicking or text input, as desired.

Linking calendar displays to other types of charts is valuable to visually explore the relationships between variables. An example can be found in the **wanderer4melb** shiny application (Wang 2019). The calendar most naturally serves as a tool for date selection: by selecting and brushing the glyphs in the calendar, it subsequently highlights the elements of corresponding dates in other time-based plots. Conversely, selecting on weather data plots, linked to the calendar can help to assess if very hot/cold days and heavy rain affect the number of people walking in downtown Melbourne. The linking between weather data and calendar display is achieved using the common dates.

2.4 Reasons to use calendar-based graphics

The calendar-based graphics provides a contextual canvas to plot glyphs into a familiar calendar format. The purpose of this calendar display is to facilitate quick discoveries of

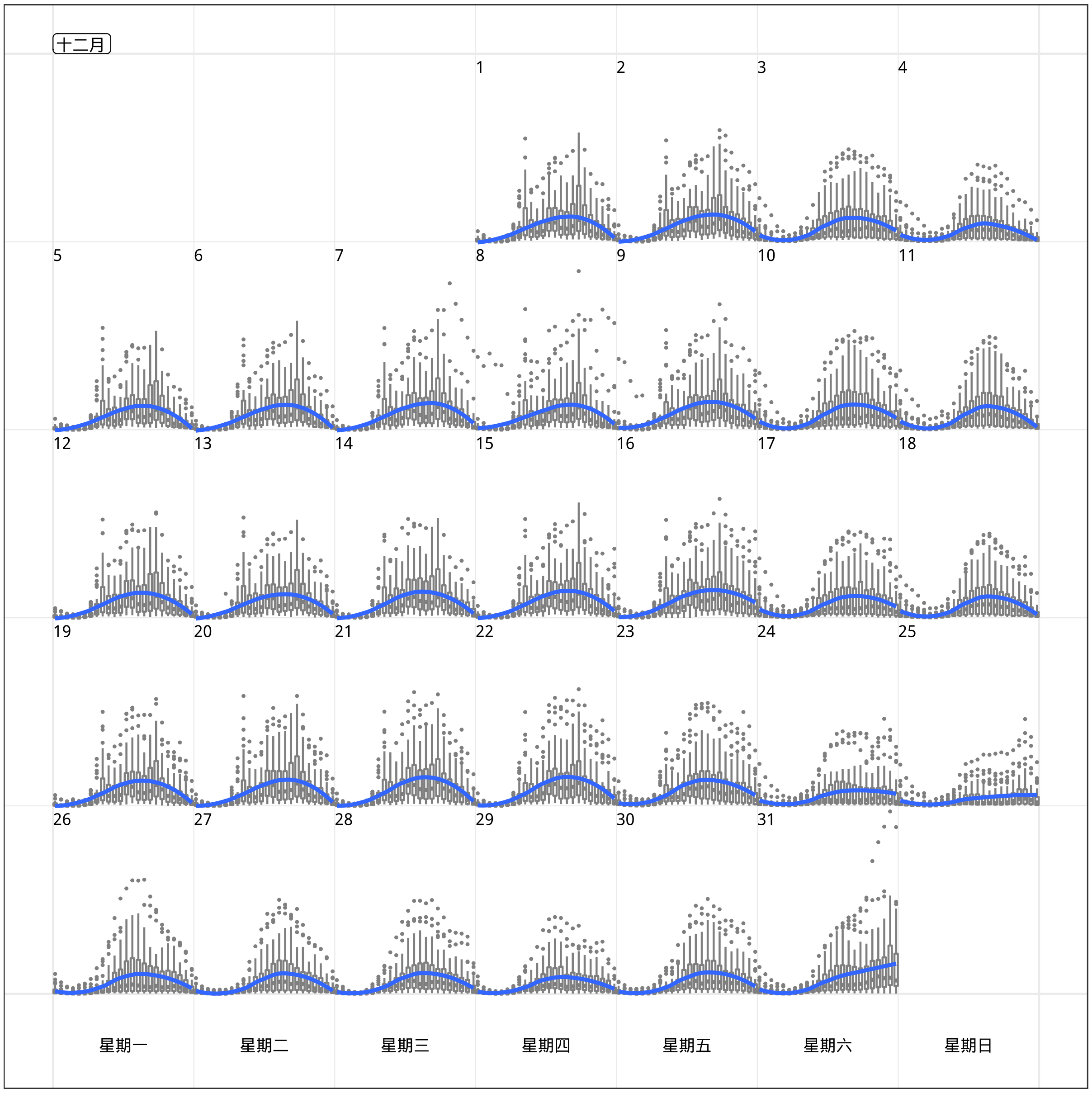


Figure 11: Side-by-side boxplots of hourly counts for all the 43 sensors in December 2016, with the loess smooth line superimposed on each day. It shows the hourly distribution in the city as a whole. The increased variability is noted on the last day of December as approaching towards New Year's Eve. The month and week day are labeled in Chinese, which demonstrates the natural support for languages other than English.

unusual patterns in people's activities, which is critical in data visualisation. It not only complements the traditional graphical toolbox for understanding general trends, but it profiles vivid and detailed data stories about the way we live. Comparing to Figure 2 and 3, Figure 9 is more informatively compelling: whether special events happened on which day and whether they are day or night events. Figure 9 informs that many events were held in Birrarung Marr during weekends, while September's events took place on Fridays, which is difficult to discern in the other two plots.

3 A fully-fledged faceting calendar method

The `frame_calendar()` function described in Section 2 is a data restructuring function rather than a plotting tool. Thus, a full pipeline inevitably involves two separate steps: data transforming and data plotting. We provide an alternative method `facet_calendar()` that extends the **ggplot2** graphical system so that to be able to make a calendar plot with one shot. A faceting method lays out panels in a grid. The faceting calendar method simply requires a variable containing dates for preparing a sequence of panels defined by Equation (1), and the coordinates and everything else have been taken care by **ggplot2** internals.

Formal axes and labels previously absent from plots generated by the `frame_calendar()` are now made available, noted in Figure 12. It is much easier for readers to infer the scaling employed for the plot. Non-existing panels mean non-existing days in the month, and blank panels indicate missing data on the day. This avoids the confusion about missing data or days when missingness lives in the ends of month panels, which is brought by the `frame_calendar()`.

However, the `facet_calendar()` takes much more run time comparing to data rearranging, since so many panels need rendering as well as other visual elements. The faceting calendar plot also uses lots of real estate for making room for strips and axises. Figure 12 also takes much more graphic space than the compact display in Figure 8, but the amount of the data is as one third as the latter. For fast rendering and economic of space, we still recommend the `frame_calendar()` approach.

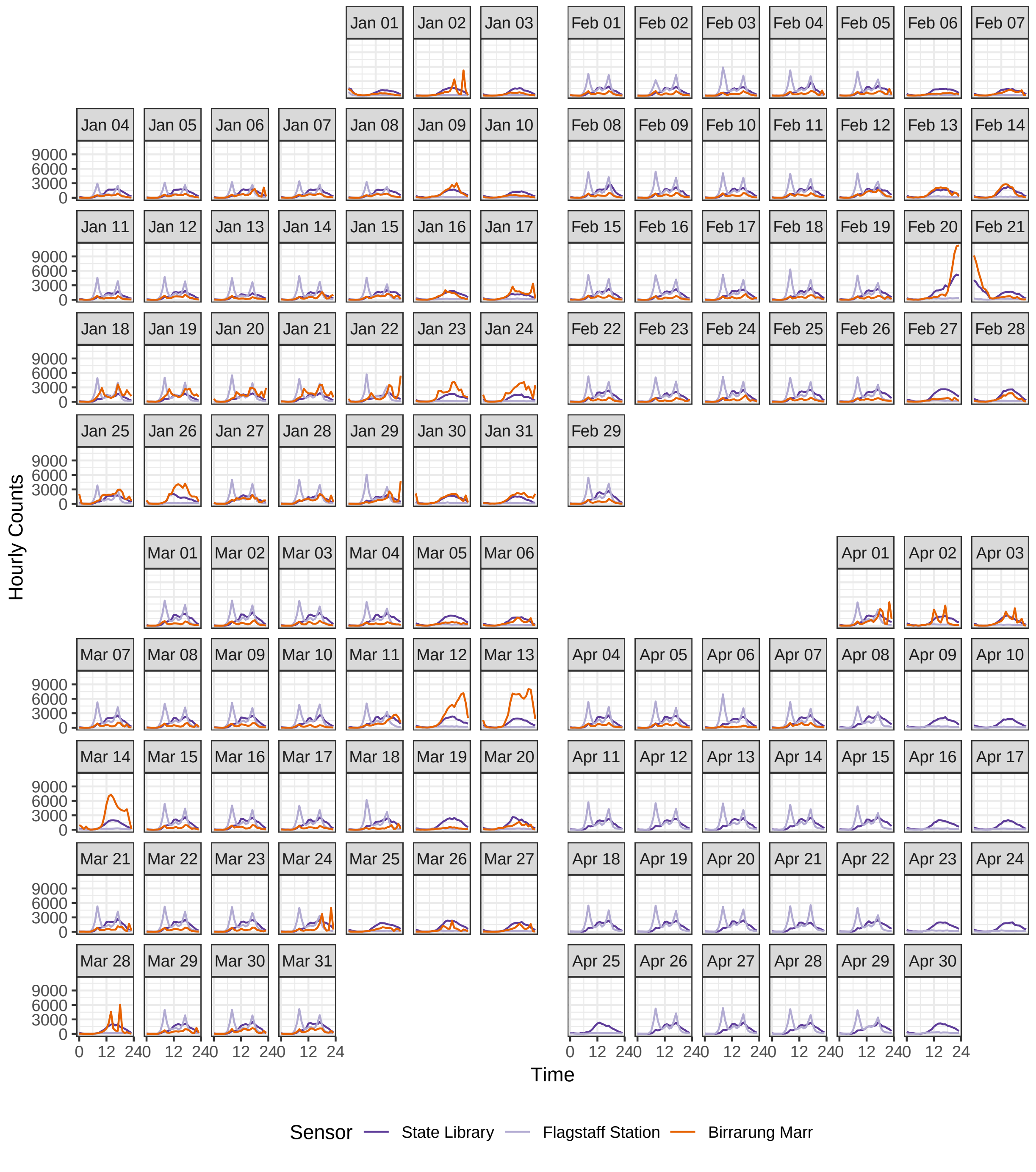


Figure 12: A fraction of Figure 8 is reconstructed with `facet_calendar()` for native **ggplot2** support on labels and axes.

4 Case study

The use of the calendar display is illustrated on smart meter energy usage from four households in Melbourne, Australia. Individuals can download their own data from the energy supplier, and the data used in this section is sourced from four colleagues of the authors. The calendar display is useful to help people understand their energy use. The data contains half-hourly electricity consumption in the first half of 2018. The analysis begins by looking at the distribution over days of week, then time of day split by work days and non-work days, followed by the calendar display to inspect the daily schedules.

Figure 13 shows the energy use across days of week using boxplots. Inspecting the medians across households tells us that household 3, a family size of one couple and two kids, uses more energy over the week days, than other households. The relatively larger boxes for household 2 indicates greater variability in daily energy consumption with noticeable variations on Thursdays, and much higher usage over the weekends. The other two households (1 and 4) tend to consume more energy with more variation on the weekends relative to the week days, reflecting of work and leisure patterns.

Figure 14 shows energy consumption against time of day, separately by week day and weekend. Household 1 is an early bird, starting their day before 6 and going back home around 18 on week days. They switch air conditioning on when they get home from work and keep it operating until mid-night, learned from the small horizontal cluster of points around 0.8 kWh. On the other hand, the stripes above 1 kWh for household 2 indicates that perhaps air conditioning runs continuously for some periods, consuming the twice the energy as household 1. A third peak occurs around 15 for household 3 only, likely when the kids are home from school. They also have a consistent energy pattern between week days and weekends. As for household 4, their home routine starts after 18 on week days. Figures 13 and 14, part of a traditional graphical toolkit, are useful for summarizing overall deviations across days and households.

Figure 15 displays the global scaling of each household's data in calendar layout, unfolding their day-to-day life via electricity usage. Glancing over household 1, we can see that their overall energy use is relatively low. Their week day energy use is distinguishable from their weekends, indicating a working household. The air conditioner appears to be used

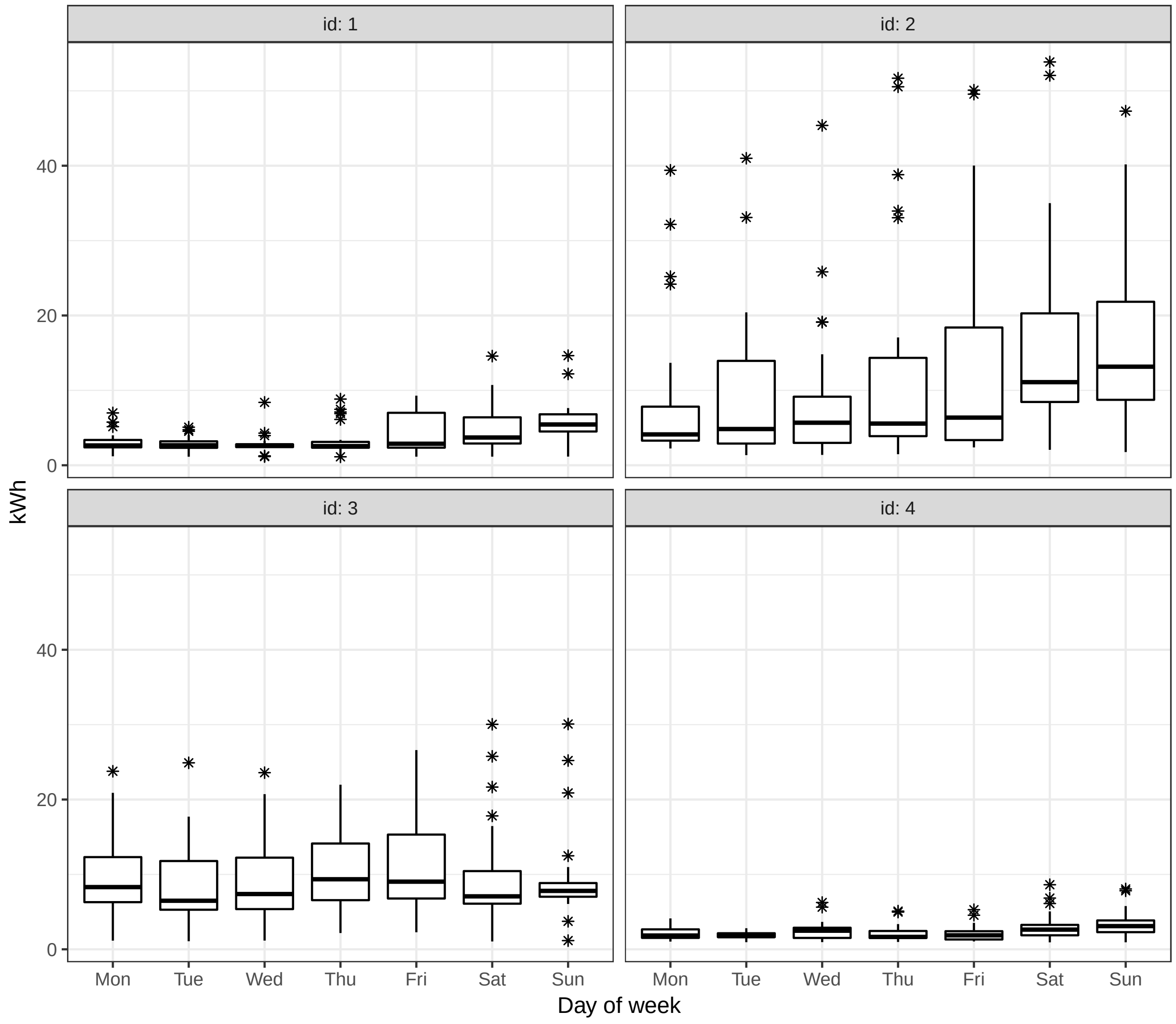


Figure 13: Boxplots of daily energy usage against day of week for four households. Suggested by the medians, household 3 uses more enery than the others on the week days, due to a large family size. By constrast, household 2 sees considerally large variability.

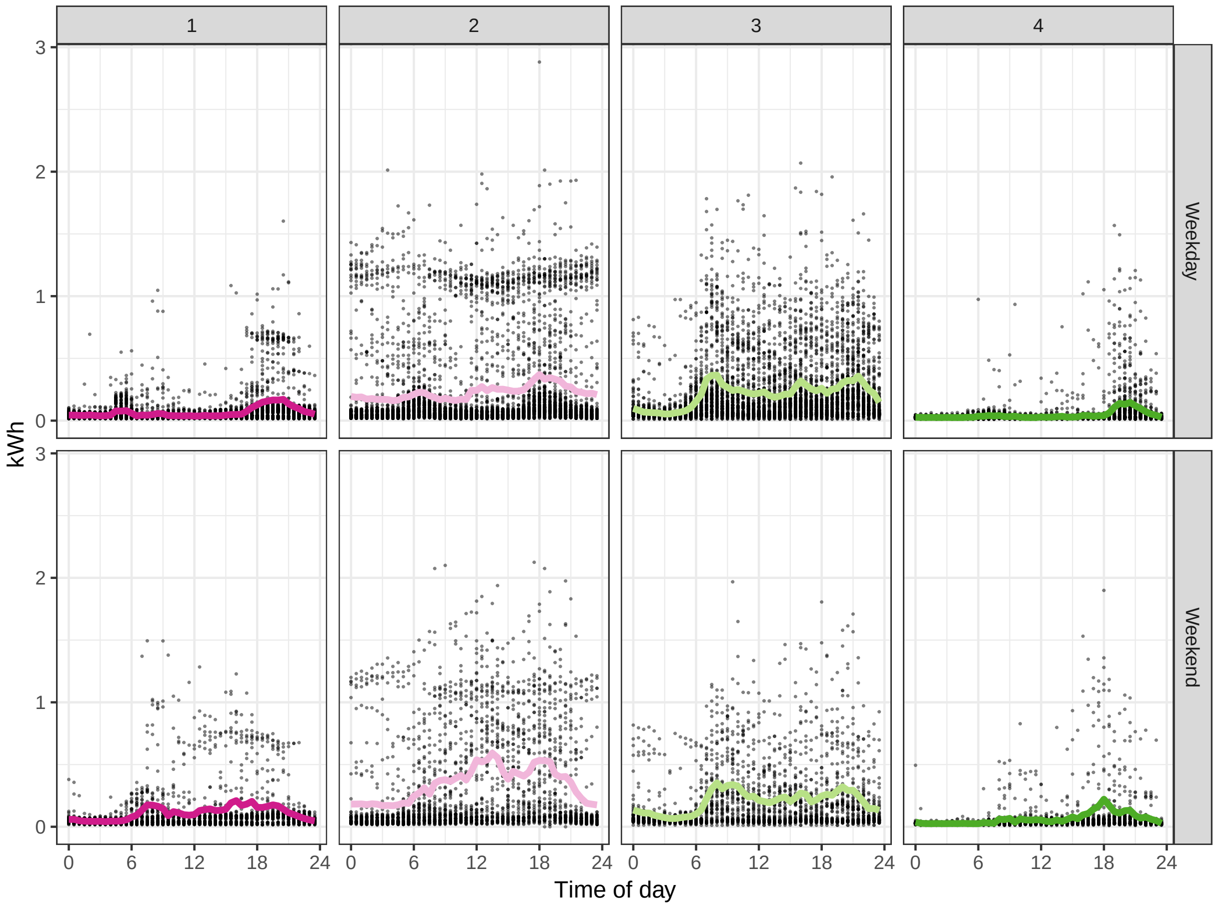


Figure 14: Scatterplot between half-hourly energy usage and time of day with the averages overlaid, constrasting week days and weekends for each household. All have different daily routines between week days and weekends, except for household 3. On week days, household 1 wakes up early before 6, and household 2 around 6, followed by household 3 and 4. The use of air conditioning is noted in household 1 and 2.

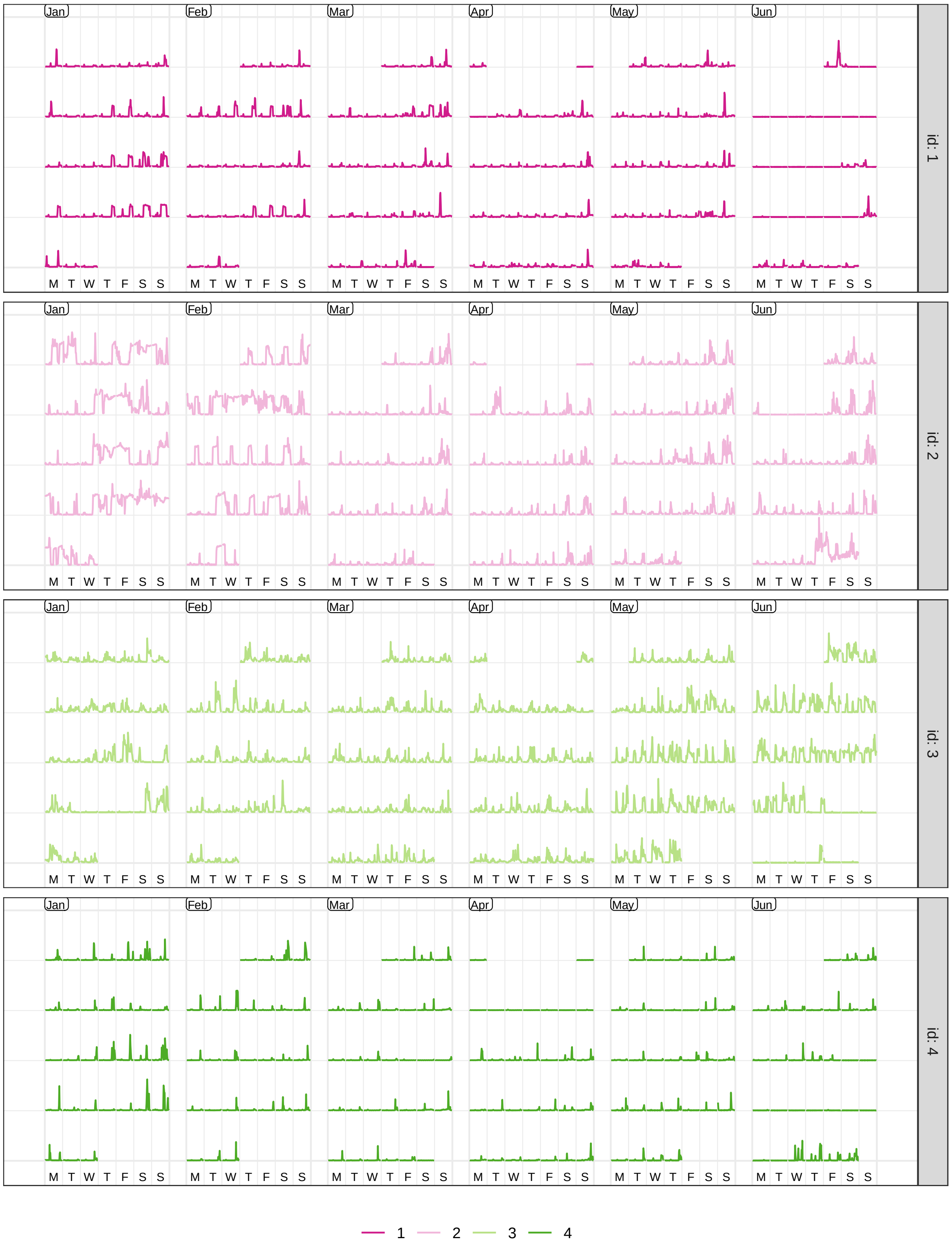


Figure 15: Calendar displays faceted by each household based on common scales. We can precisely tell each indidual's vacation days and the overnight use for household 2 in mid January.

in the summer months (January and February) for a couple of hours in the evening and weekends. In contrast, household 2 keeps cooling system functioning for much longer hours, which becomes more evident from late Wednesday through Thursday to early Friday in the mid January. These observations help to explain the stripes and clusters of household 2 in Figure 14. It is difficult to give a succinct description of household 3 since everyday energy pattern is variable, but May and June see more structures than the previous months. Individual data can be idiosyncratic, hence aggregated plots like Figure 13 and 14 are essential for assembling pieces to pictures. However, the calendar plots speak the stories about the exact vacation time that are untold by previous plots, for example their vacation time. Household 1 is on vacation over three weeks of mid June, and household 2 also planned to take some days off in the second week of June. Further household 3 takes one short trip in January and the other one starting in the fourth week of June. Household 4 is away over two or three weeks early April and late June. They all tend to take breaks during June due to the fact that the University winter break starts in June.

5 Discussion

The calendar-based visualization provides data plots in the familiar format of an everyday tool. Patterns on special events and public holidays for the region are more visible to the viewer.

The calendar layout will be useful for studying consumer trends and human behavior. It will be less useful for physical patterns like climate, 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 methodology creates the western calendar layout, because most countries have adopted this format. The main difference between countries is the use of different languages for labeling, which is supported by the software. Formats beyond the western calendar, or six-weeks and tetris-like layouts could be achieved by slightly tweaking the modular arithmetic approach. These features will be added as new options for the future.

Acknowledgements

We would like to thank Stuart Lee and Heike Hofmann for their feedback on earlier versions of this work. We thank Thomas Lin Pedersen for pointing out some critical **ggplot2** internals, which makes the `facet_calendar()` implementation possible. We are very grateful for anonymous reviewers for helpful comments that have led to many improvements of the paper. The most recent versions of the `frame_calendar()` and `facet_calendar()` functions are included in the **sugrrants** R 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

Becker, R. A., Cleveland, W. S. & Shyu, M.-J. (1996), 'The Visual Design and Control of Trellis Display', *Journal of Computational and Graphical Statistics* **5**(2), 123–155.
URL: *http://www.tandfonline.com/doi/abs/10.1080/10618600.1996.10474701*

Chambers, J. M., Cleveland, W. S., Kleiner, B. & Tukey, P. A. (2017), *Graphical Methods for Data Analysis*, Chapman and Hall/CRC, New York, NY.

Chang, W., Cheng, J., Allaire, J., Xie, Y. & McPherson, J. (2019), *shiny: Web Application Framework for R*. R package version 1.3.2.
URL: *https://CRAN.R-project.org/package=shiny*

City of Melbourne (2017), *Pedestrian Volume in Melbourne*. Last accessed 2017-07-09.
URL: *http://www.pedestrian.melbourne.vic.gov.au*

Cleveland, W. S. (1979), 'Robust locally weighted regression and smoothing scatterplots', *Journal of the American Statistical Association* **74**(368), 829–836.

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. (2019), *geofacet: 'ggplot2' Faceting Utilities for Geographical Data.* R package version 0.1.10.
URL: *https://CRAN.R-project.org/package=geofacet*

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

Jones, H. (2016), *Calendar Heatmap.* Online post.
URL: *https://rpubs.com/haj3/calheatmap*

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.

Sievert, C. (2018), *plotly for R.* Last accessed 2018-10-20.
URL: *http://plotly-r.com*

Tufte, E. R. (1983), *The Visual Display of Quantitative Information*, Graphics press Cheshire, CT.

Unwin, A. & Valero-Mora, P. (2018), 'Ensemble Graphics', *Journal of Computational and Graphical Statistics* **27**(1), 157–165.
URL: *https://www.tandfonline.com/doi/full/10.1080/10618600.2017.1383264*

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. (2019), *wanderer4melb: Shiny App for Wandering Around the Downtown Melbourne 2016*. R package version 0.2.0.
URL: *https://github.com/earowang/wanderer4melb*

Wang, E., Cook, D. & Hyndman, R. (2019), *sugrrants: Supporting Graphs for Analysing Time Series*. R package version 0.2.4.
URL: *https://CRAN.R-project.org/package=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. (2019), *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.1.1.
URL: *https://CRAN.R-project.org/package=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.

Wong, J. (2013), *TimeProjection: Time Projections*. R package version 0.2.0.
URL: *https://CRAN.R-project.org/package=TimeProjection*