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 25, 2019

Abstract

Calendars are broadly used in society to display temporal information and events. This paper describes a new calendar display for plotting data, that includes a layout algorithm with many options, and faceting functionality. The functions use modular arithmetic on the date variable to restructure the data into a calendar format. 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, is 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 functions for the calendar algorithm are available in the R package **sugrrants**.

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

1 Introduction

A new method for organizing and visualizing temporal data, collected at sub-daily intervals, into a calendar layout is developed. The format is created using modular arithmetic, giving a restructuring of the data that can then be integrated into a data pipeline. The core component of the pipeline is to visualize 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 variables in the data to graphical elements. The data restructuring approach is consistent with the tidy data principles available in the **tidyverse** suite of tools (Wickham 2017). 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 human activities, especially 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 are many sensors installed across the inner-city area, that count pedestrians every hour (Figure 1). Data from 43 sensors in 2016 is analyzed here. This data can shed light on people's daily rhythms, and assist the city administration and local businesses with event planning and operational management. Patterns relative to special events (such as public holidays and recurring cultural/sporting events) would be worth studying in comparison to regular days, but conventional displays of time series data may bury this detail.

A routine examination of the data would involve constructing a time series plot to examine the temporal patterns. The faceted plots in Figure 2 give an overall picture of the foot traffic at three different sensors in 2016. Further faceting by day of the week (Figure 3) provides a better view of the daily and sub-daily (hourly) pedestrian patterns. Flagstaff Station has a strong commuter pattern, with peaks in the morning and evening, and no pedestrians on the weekend. Around the State Library there are pedestrians walking around during the day, and an unusually large number on one Saturday night and Sunday morning. Birrarung Marr has a varied pedestrian pattern, with very different numbers of people on different days and times.

Faceting, initially called trellis displays (Becker et al. 1996), is an example of a small

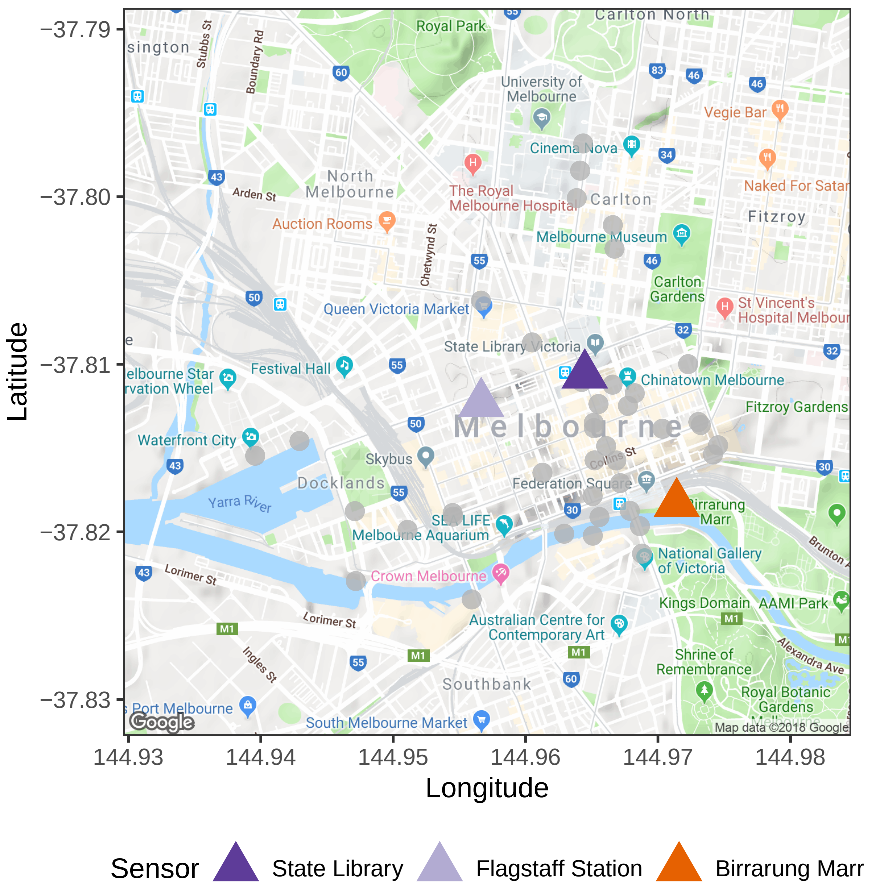

Figure 1: Google map of the Melbourne city area, grey dots indicate sensor locations. The three locations highlighted will be analyzed in the paper: the State Library is a public library; Flagstaff Station is a train station, closed on non-work days; Birrarung Marr is an outdoor park hosting many cultural and sports events.

multiple (Tufte 1983), where different subsets of the same data are displayed across one or more conditioning variables. It allows the comparison of subsets. Faceting can also be thought of as a simple ensemble graphic (Unwin & Valero-Mora 2018). It is a homogeneous collection of plots, whereas the ensemble graphics broadly organize related plots for a data set together into one display.

The work is inspired by Wickham et al. (2012), which uses modular arithmetic to display spatio-temporal data as glyphs on maps. It is also related to recent work by Hafen (2019) 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.

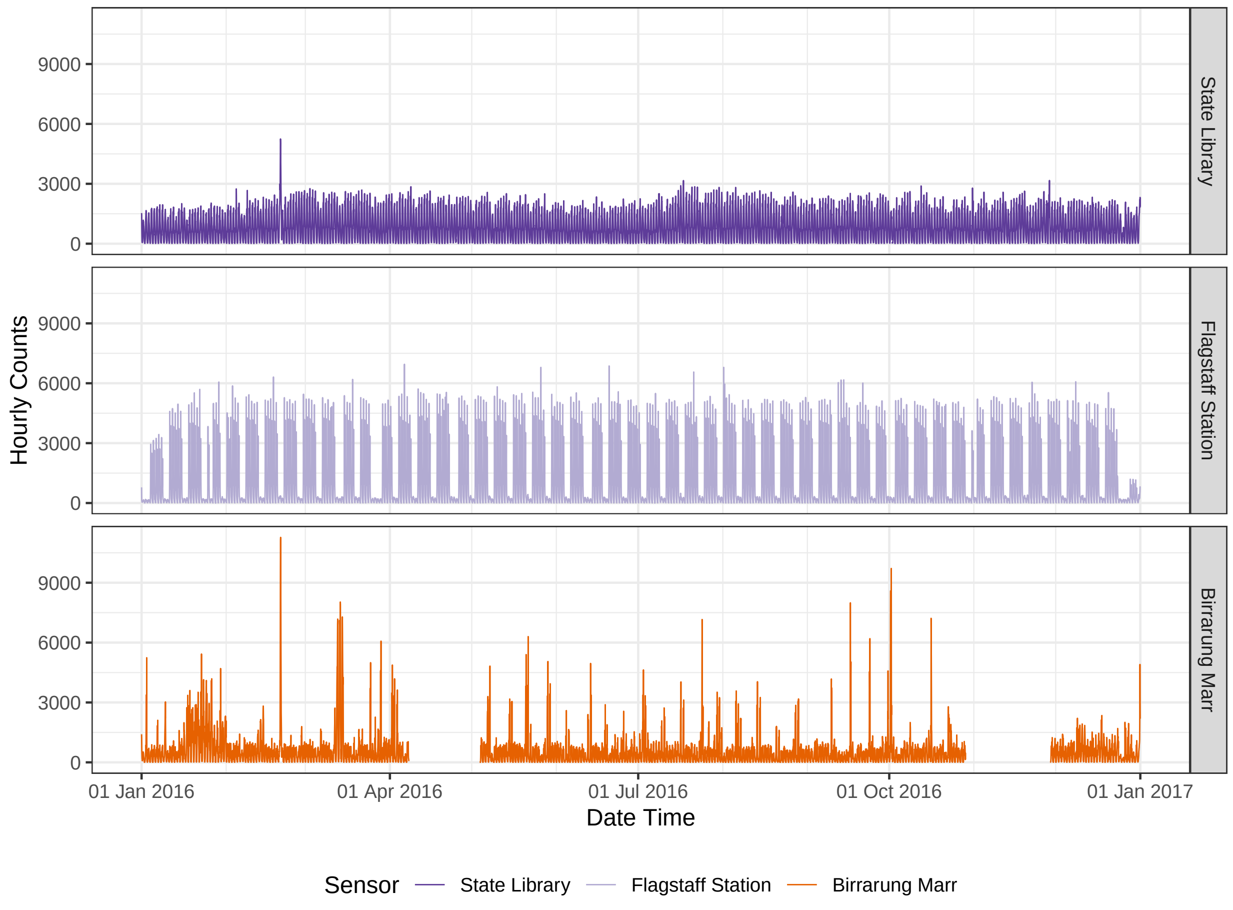

Figure 2: Time series plots showing 2016 pedestrian counts, measured by three different sensors in the city of Melbourne. Small multiples of lines show that the foot traffic varies at one location to another. The spike in counts at the State Library corresponds to the timing of the event “White Night”, where there were many people taking part in activities in the city throughout the night. A relatively persistent pattern repeats from one week to another at Flagstaff Station. Birrarung Marr looks rather noisy and spiky, with several runs of missing records.

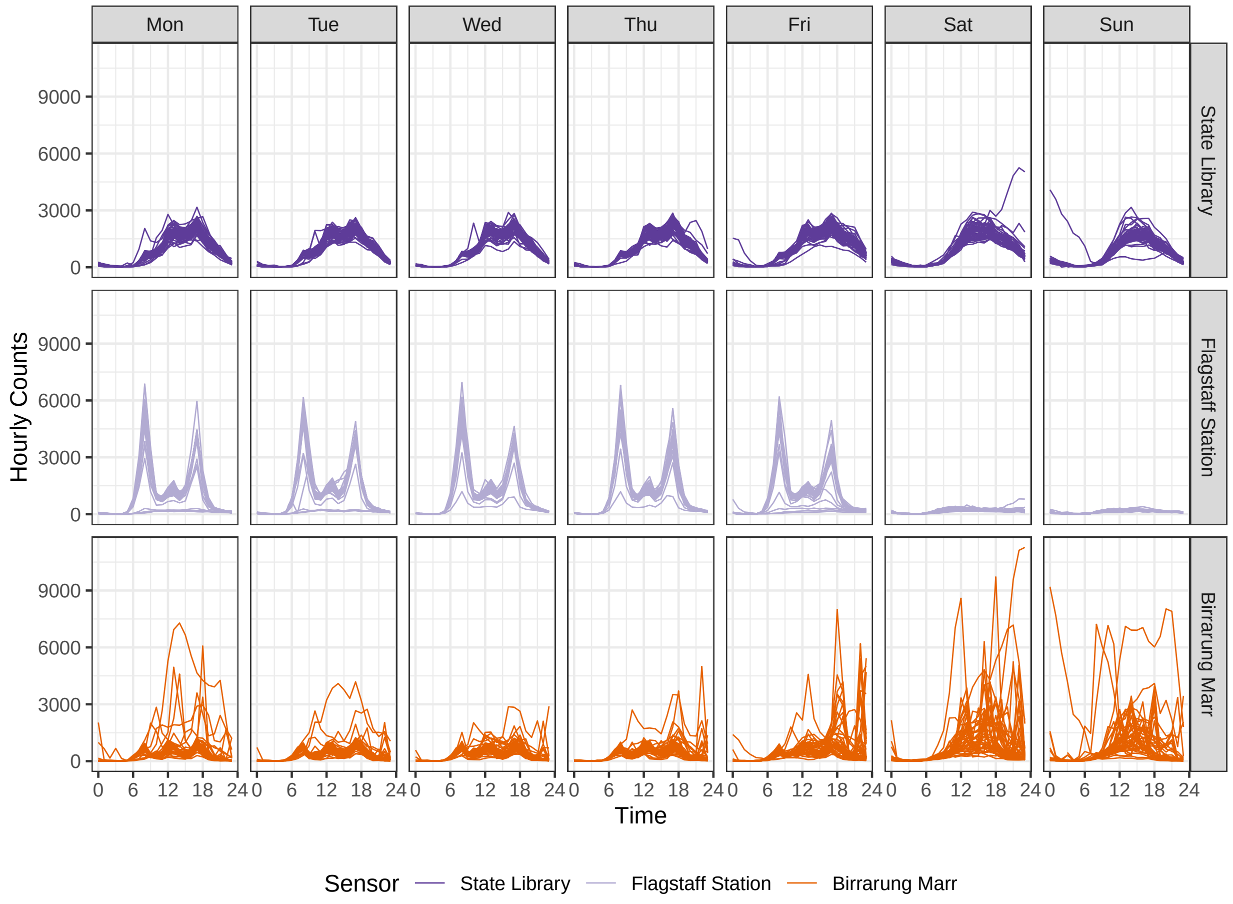

Figure 3: Hourly pedestrian counts for 2016, faceted by sensors, and days of the week. The focus is on time of day and day of week across the sensors. Daily commuter patterns at Flagstaff Station, the variability of the foot traffic at Birrarung Marr, and the consistent pedestrian behavior at the State Library, can be seen.

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 weekdays 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. Some variants of calendar-based heatmaps have been implemented in R packages: **TimeProjection** (Wong 2013), **ggTimeSeries** (Ather Energy Pvt Ltd 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. Color-encoding is also low on the hierarchy of optimal variable mapping (Cleveland & McGill 1984, Lam et al. 2007).

The proposed algorithm goes 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 lines and other types of glyphs into the graphical toolkit for the calendar layout; (3) to accentuate unusual patterns, such as those related to special events, for 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 details the construction of the calendar layout in depth. It describes the algorithms of data transformation (Section 2.1), the available options (Section 2.2), variations of its usage (Section 2.3), including the full faceting extension equipped with formal labels and axes (Section 2.3.5). An analysis of half-hourly household energy consumption, using the calendar display, is illustrated in a case study in Section 3. Section 4 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 modular arithmetic, 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 rows and columns refer to weeks of the month and days of the week respectively. 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 deterministic given the year of the calendar. Months are of different lengths, ranging from 28 to 31 days. Some months could extend over six weeks, but for these months the last few days are wrapped up to the top row of the block for compactness, and because it is convention. The fifth row could be blank for February if the month starts on Monday. 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 the row (week within the month), $1 \le j \le 7$, is the column (day of the week), with $(1, 1)$ being in the upper left corner.
- $g = k, \dots, (k + d - 1)$ indexes the day in the month, inside the 35 possible cells.

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

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

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

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 be the scaled count.

Then the final coordinates for making the calendar 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, which is easier to lay out and why the subtraction is performed. Within each cell, the starting position is the bottom left.

The R package, **lubridate** (Grolemund & Wickham 2011), is used to extract the components of time, such as days of the week and the number of days in a month, that create the layout. These time variables are converted to integers for the modular arithmetic. Note that for any date-time information is associated with time zone. If your data is collected over multiple time zones, you will need to convert them to the same time zone before conducting any temporal analysis.

Figure 4 shows the line graphs 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). Thus, 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 weekday 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 weekday texts are uniformly positioned on the bottom of the whole canvas, that is $\min(y)$,

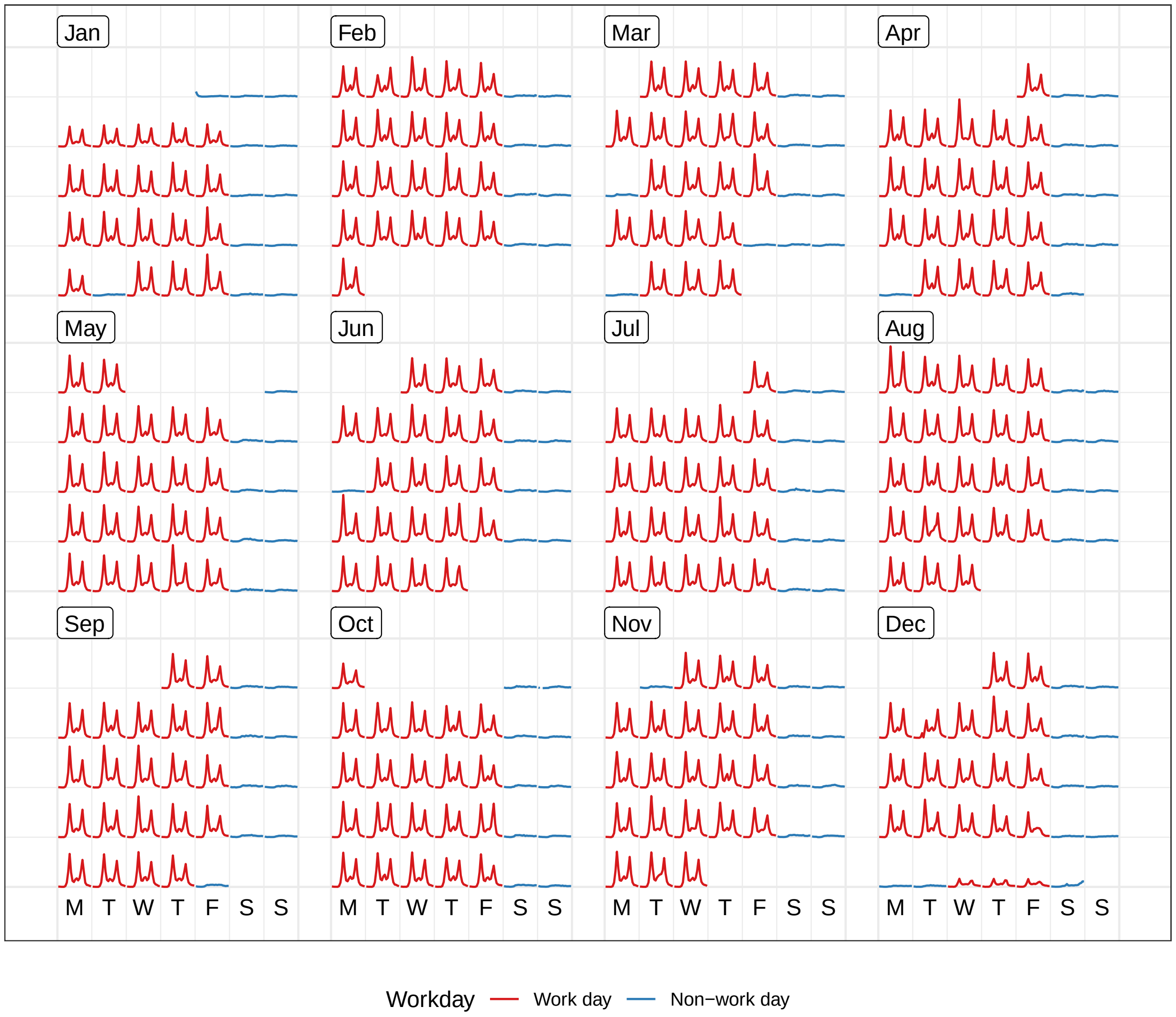

Figure 4: The calendar plot of hourly foot traffic at Flagstaff Station ranging from 0 to 6952, using line graphs. The disparities between weekday and weekend along with public holiday, are immediately apparent. The arrangement of the data into a 3×4 monthly grid represents all the traffic in 2016. Note 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 in May and October.

with the central position of a cell $x/2$ for each j. Formal axes and labels are discussed later in Section 2.3.5.

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()`:

```r
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 in Figure 4, the `x` is the time of the day, and `y` is the count. The `date` argument specifies the date variable in the data, facilitating the range of dates plotted in 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 weekday, 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). This produces star glyphs (Chambers et al. 2017), where time series lines are transformed in polar coordinates, embedded in the monthly calendar layout. It is most useful in exhibiting cyclical patterns in the data.

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 on 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 5 shows the same data as Figure 4, scaled locally using `scale = "free"`. The daily trends are magnified.

The `free_wday` scales each weekday together. It can be useful to compare trends across weekdays, allowing relative patterns for weekends versus weekdays 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 weekday and month would be different across countries. Other language specifications than English, for text labeling, are available.

2.3 Varieties of calendar display

2.3.1 Information overlay

Plots can be layered. A comparison of sensors can be done by overlaying them in the same calendar pane. Figure 6 overlays the pedestrian counts for three locations on the same calendar. Differences between the pedestrian patterns at these locations can be more directly compared. For example, the magnitude of the difference in pedestrians at Flagstaff Station at peak hours of commuter can be seen. The big peak in pedestrian counts for special events at Birrarung Marr is clear. Birrarung Marr has a very distinct temporal pattern relative to the other two locations. The nighttime events, such as White Night (third Saturday in February), only affects the foot traffic at the State Library and Birrarung Marr.

2.3.2 Faceting by covariates

To avoid overlapping, when differences between groups are large enough to be seen separately, the calendar layout can be faceted into a series of subplots for the different sensors. Figure 7 shows calendar plots that are faceted by sensors. This arrangement allows comparison of the overall structure between sensors, while emphasizing individual sensor variation. In particular, it can be immediately learned that Birrarung Marr was busy and packed over many weekends, but events took place on Friday evenings only in September. The Australian Open, a major international tennis tournament, attracted constant foot traffic in

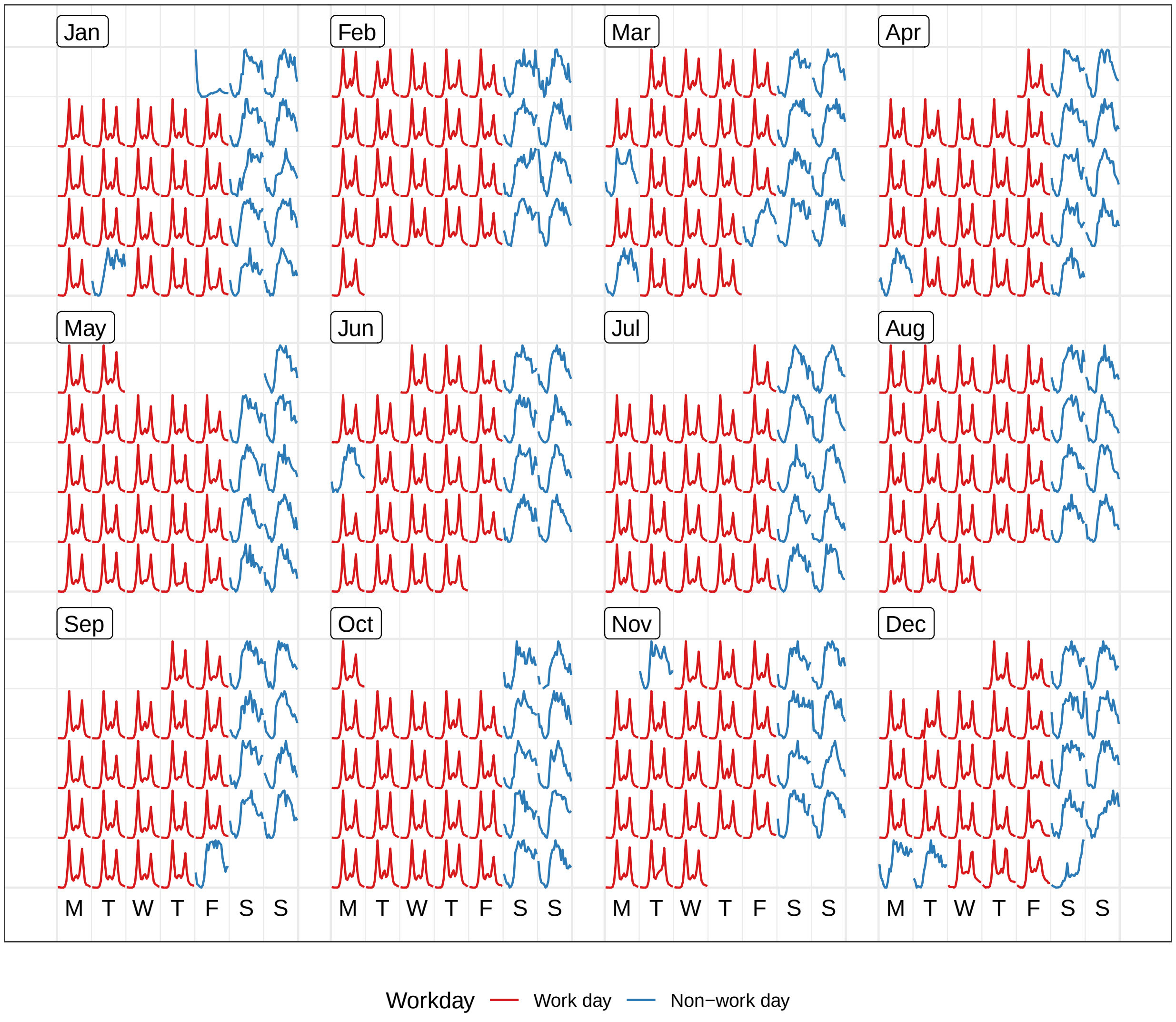

Figure 5: Line graphs on the calendar format showing hourly foot traffic at Flagstaff Station, scaled individually by day. The shape on a single day becomes more distinctive, as compared to Figure 4.

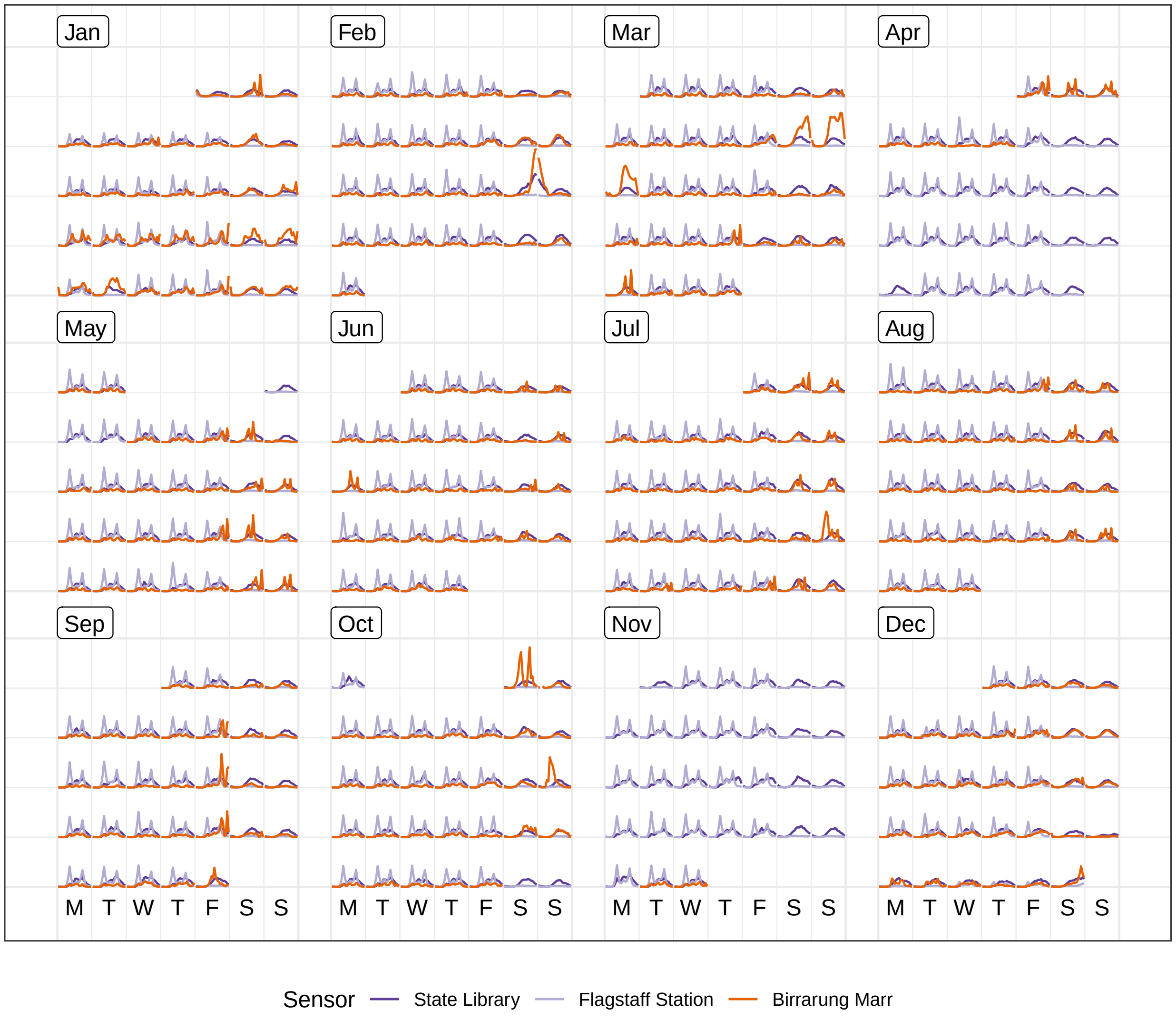

Figure 6: Overlaying line graphs of the three sensors in the monthly calendar, to enable a direct comparison of the counts at three locations. They have very different traffic patterns. Birrarung Marr tends to attract large numbers of pedestrians for special events typically held on weekends, contrasting to the bimodal massive peaks showing commuting traffic at Flagstaff Station.

the last two weeks of January. The calendar plot can be faceted by any categorical variable in the data.

2.3.3 Different types of plots

Many types of plot can be shown in a calendar pane, by taking advantage of the existing **ggplot2** plotting capabilities. An example is shown in Figure 8: the panes contain lag scatterplots for Flagstaff Station from Week 1 to 17 in 2016, constructed with the scaling for each day and aligning by days of the week, where the lagged hourly count is assigned to the `x` argument and the current hourly count to the `y` argument. It indicates strong autocorrelation on weekends, and weak autocorrelation on work days. The V-shape in the weekday graphs arises when the next hour sees a substantial increase or decrease in counts.

The algorithm can also produce more complicated plots, such as boxplots. Figure 9 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.

2.3.4 Interactivity

The previous calendar plots are static, made with **ggplot2**. The interactivity of calendar-based displays can be easily enabled, as long as the interactive graphics 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 a point in the calendar plot, 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

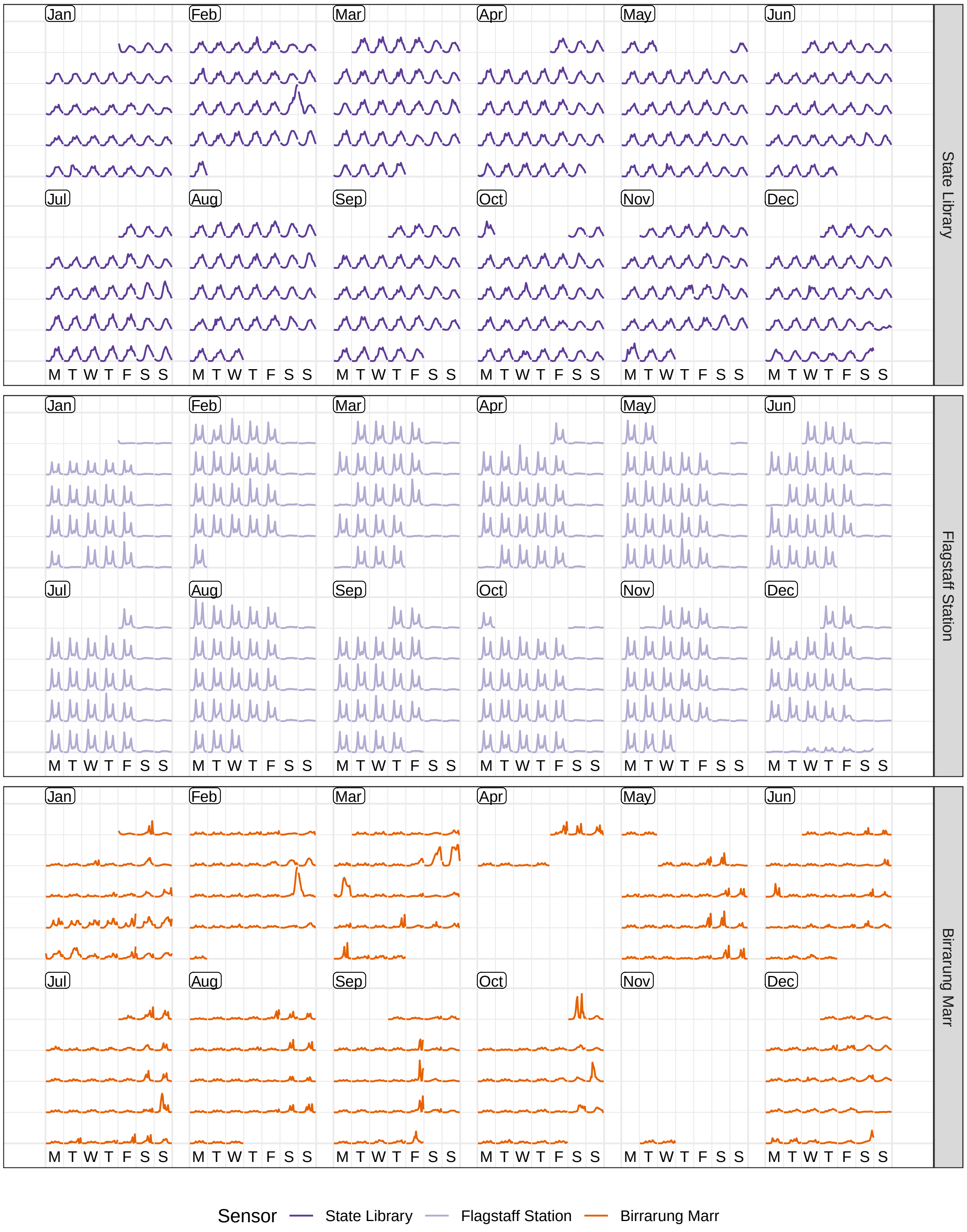

Figure 7: Line graphs, embedded in the 6 × 2 monthly calendar, colored and faceted by the 3 sensors. The variations of an individual sensor are emphasized.

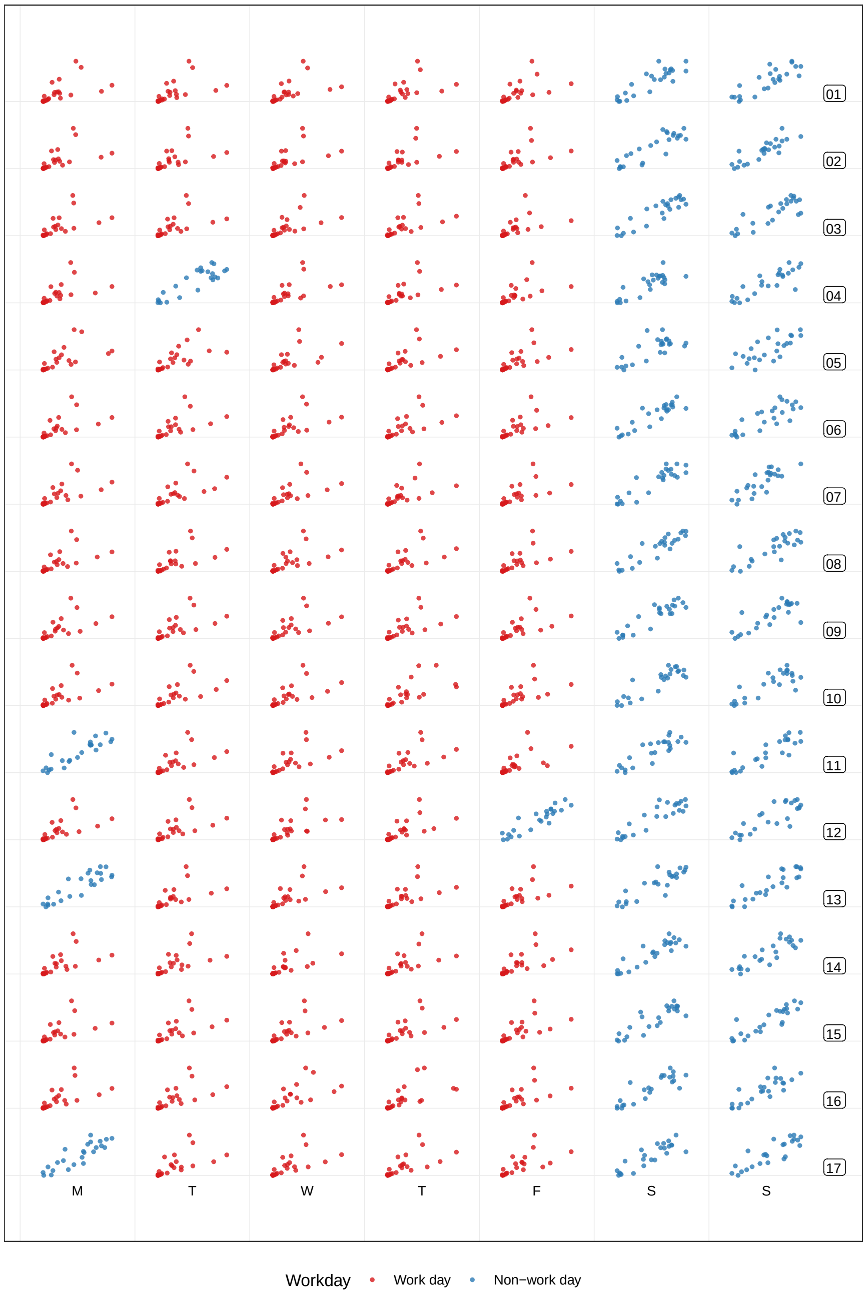

Figure 8: Examining lag 1 autocorrelation for Flagstaff Station from week 1 to 17, using lag scatterplots scaled by each day and aligned by days of the week. Each hour's count is plotted against the previous hour's count. The autocorrelation is stronger on non-work days (blue) than work days (red).

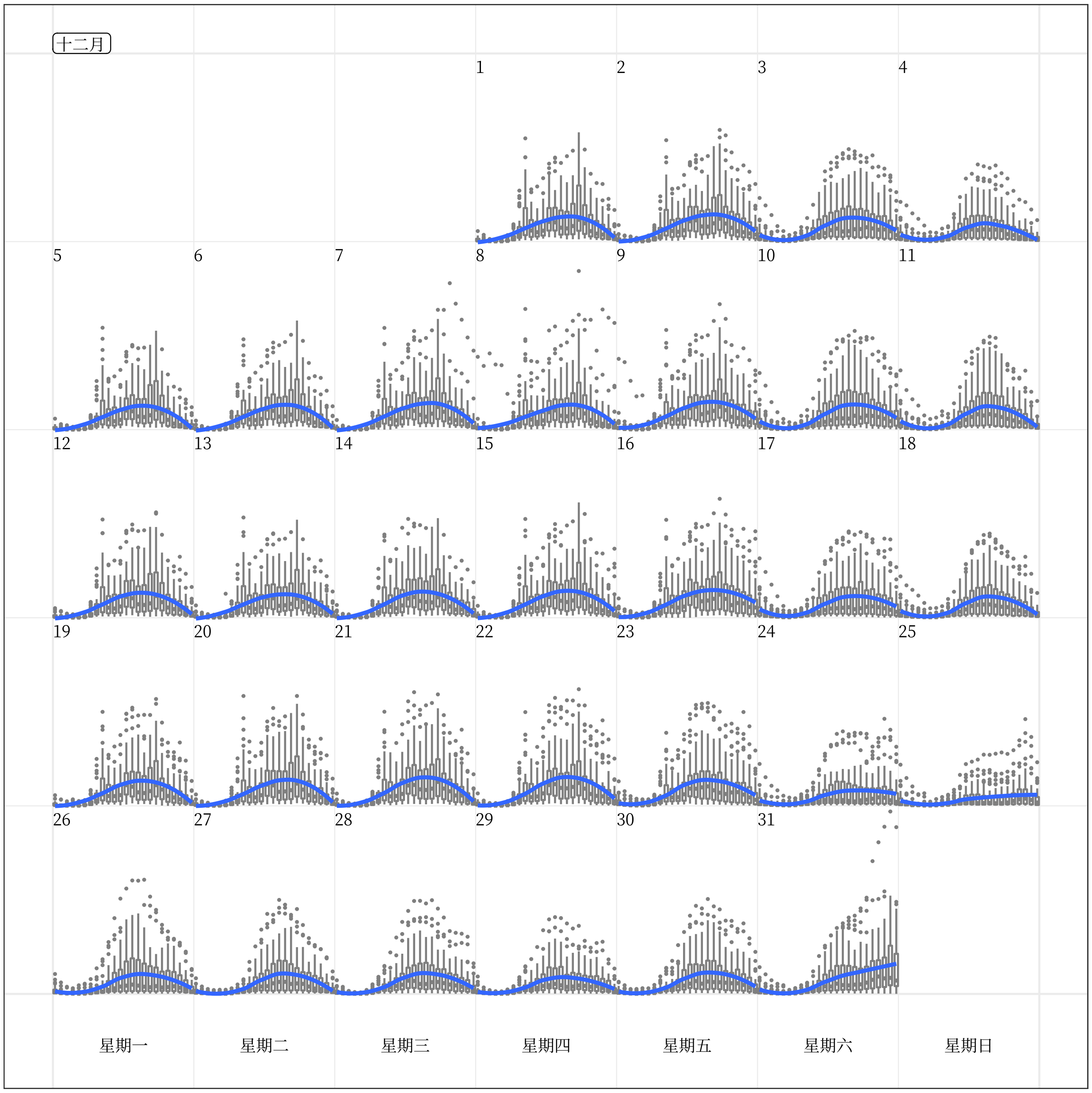

Figure 9: Side-by-side boxplots of hourly counts for all the 43 sensors in December 2016, with the loess smooth line superimposed on each day. It shows the hourly distribution in the city as a whole. The increased variability is notable on the last day of December as New Year's Eve approaches. The month and weekday are labeled in Chinese, which demonstrates the support for languages other than English.

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. The linking between weather data and calendar displays is achieved using the common dates.

2.3.5 Faceted calendar

The `frame_calendar()` function described in Section 2.2 is a data restructuring function, neatly integrating into a data pipeline but it requires two steps: data transformation and then plot. There is also little freedom to tailor axes and labels, because specialist code needs to be applied.

The `facet_calendar()` integrates the algorithm into the **ggplot2** graphical system so that the calendar layout is automatic, and the full functionality of axes, labels, and customization is accessible. A faceting method lays out panels in a grid. The user needs to supply the variable containing dates, in order for the faceting calendar function to prepare the arrangement of panels, as defined by Equation (1). The remainder of the plot construction for each panel is handled entirely by **ggplot2** internals.

Formal axes and labels unavailable in calendar plots generated by the `frame_calendar()` are possible (Figure 10). It is much easier for readers to infer the scaling (global or local) 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 confusion about missing data or days when missingness lives in the ends of month panels, which may occur when using `frame_calendar()`.

However, the `facet_calendar()` takes much more run time compared with `frame_calendar()`. The faceted calendar also uses more plot real estate for panel headings and axes. The reader can compare the two approaches by examining the compact Figure 6, relative to Figure 10. The space consumed by the former shows a full year, and the latter shows four months, only a third of the data. For fast rendering and economy of space, `frame_calendar()` is recommended.

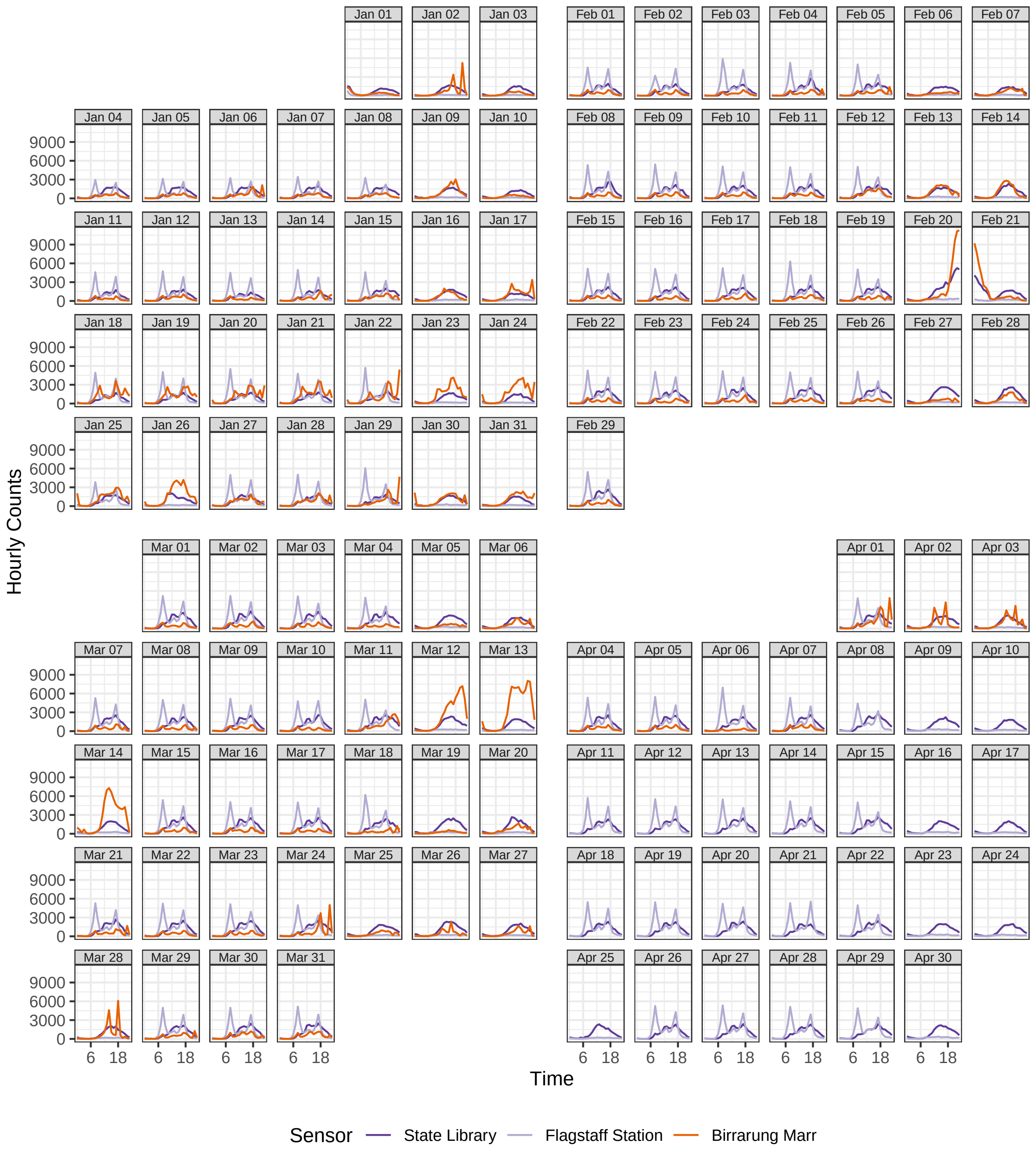

Figure 10: A faceted calendar showing a fraction of the data shown in Figure 6. The faceted calendar takes more plot real estate than the calendar plot, but it provides native **ggplot2** support for labels and axes.

2.4 Reasons to use calendar-based graphics

The purpose of the calendar display is to facilitate quick discoveries of unusual patterns in people's activities, which is consistent with why analysts should and do use data visualization. It complements the traditional graphical toolbox used to understand general trends, and better profiles vivid and detailed data stories about the way we live. Comparing the conventional displays (Figure 2 and 3) with the new display (Figure 7), it can be seen that the calendar display is more informatively compelling: when special events happened, and on what day of the week, and whether they were day or night events. For example, Figure 7 informs the reader that many events were held in Birrarung Marr on weekend days, while September's events took place on Friday evenings, which is difficult to discern from conventional displays.

3 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 11 shows energy consumption against time of day, separately by weekday and weekend. Household 1 is an early riser, starting their day before 6 and going back home around 18 on weekdays. They switch air conditioning on when they get home from work and keep it operating until midnight, evident 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 air conditioning may run continuously for some periods, consuming twice the energy as household 1. A third peak occurs around 15 for household 3 only, likely coinciding when the children arrive home from school. They also have a consistent energy pattern between weekdays and weekends. As for household 4, their home routine starts after 18 on

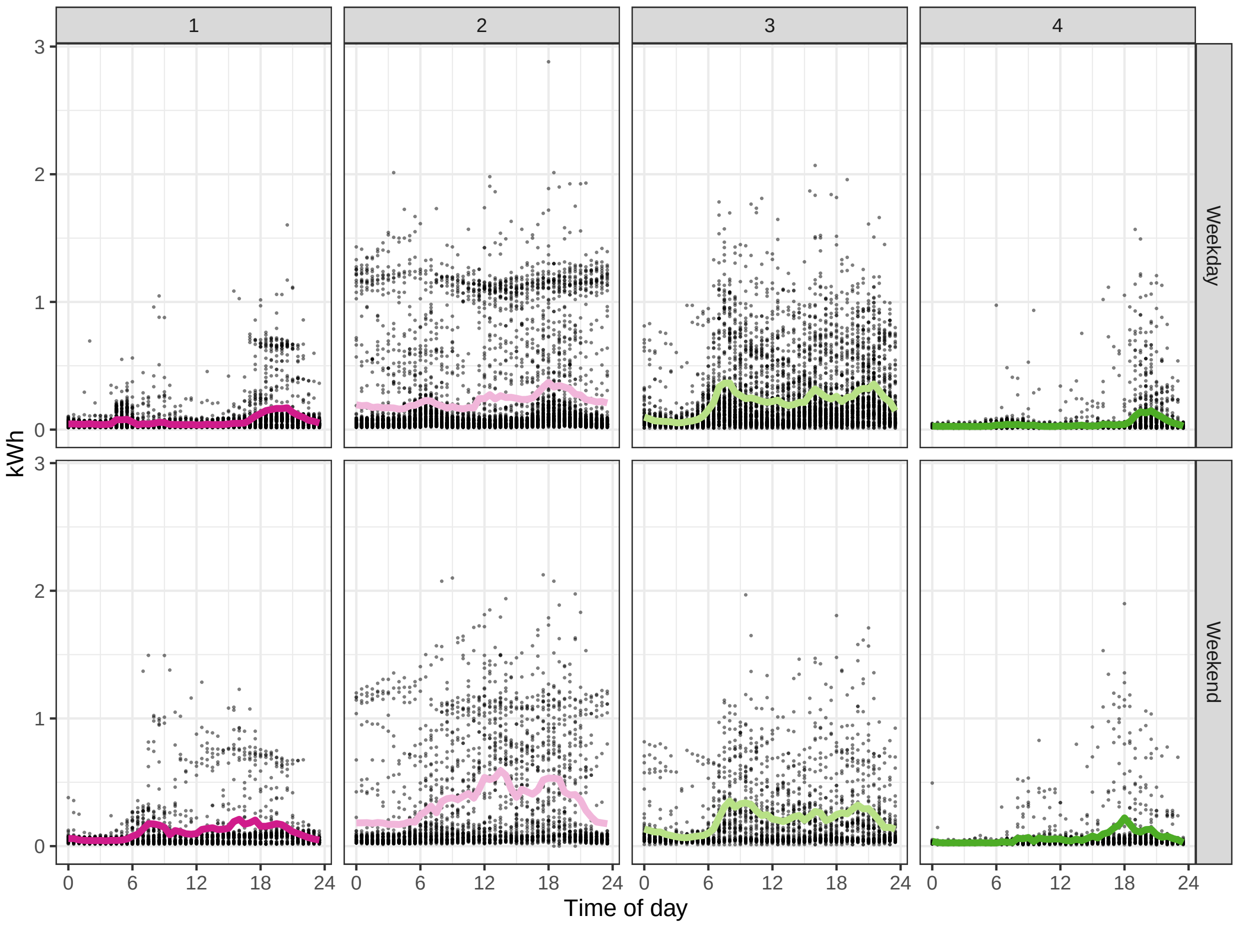

Figure 11: Examining sub-daily variability of energy usage by time of day, with hourly averages overlaid, faceted by household and type of day. On weekdays, household 1 wakes up early before 6, and household 2 around 6, followed by household 3 and 4. The use of air conditioning is notable in households 1 and 2, as seen by horizontal clusters of points.

weekdays. Figures 11, part of a traditional graphical toolkit, are useful for summarizing overall deviations across days and households.

Figure 12 displays the global scaling of each household's data in a calendar layout, unfolding their day-to-day life via electricity usage. Glancing over household 1, their overall energy use is relatively low. Their weekday energy use is distinguishable from their weekends, indicating a working household. The air conditioner appears to be used in the summer months (January and February) for a couple of hours in the evening and weekends. In contrast, household 2 keeps a cooling system functioning for much longer hours, which becomes more evident from late Wednesday through Thursday to early Friday in

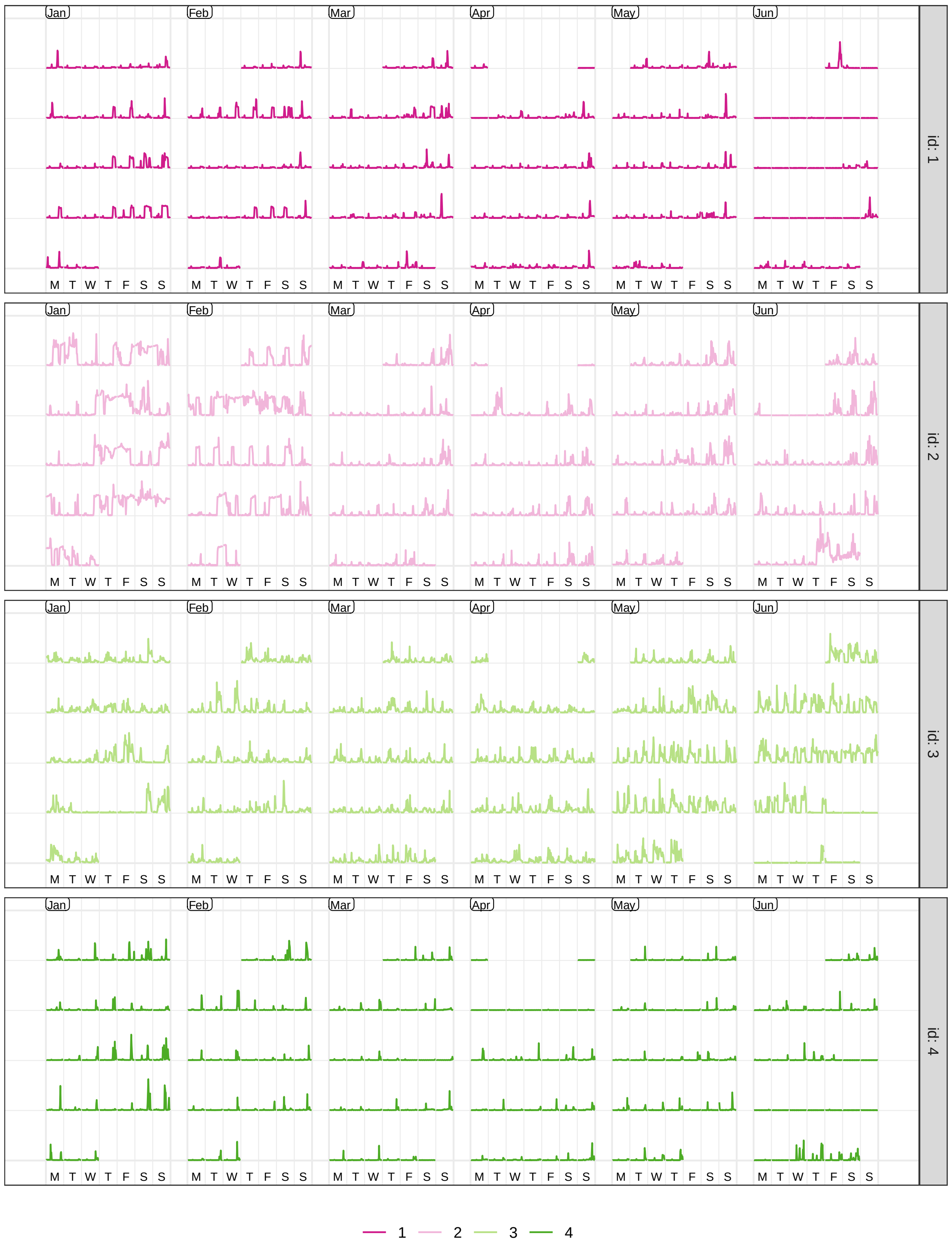

Figure 12: Calendar displays faceted by each household using global scales. Long flat low energy usage indicates vacation time, and high energy consumption by household 2 is visible in January and February. Note that April 30 is wrapped to the start of the month.

mid-January. These observations help to explain the stripes and clusters of household 2 in Figure 11. It is difficult to give a succinct description of household 3 since everyday energy pattern is variable, but May and June see more structure than the previous months. Individual data can be idiosyncratic, hence aggregated plots like Figure 11 are essential for assembling pieces to form a picture. However, the calendar plots tell the stories 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 took some days off in the second week of June. Further, household 3 takes one short trip in January and another one starting in the fourth week of June. Household 4 is away over two or three weeks in early April and late June. They all tend to take breaks during June probably due to the fact that the University winter break starts in June.

4 Discussion

The calendar-based visualization provides data plots in the familiar format of an everyday tool. Patterns relating to 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 processes such as weather. The layout does not replace traditional displays, but serves to complement them 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 in 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 to anonymous reviewers for helpful comments that have led to many improvements of the paper. The **sugrrants** R package is available from CRAN `https://CRAN.R-project.org/package=sugrrants` and the development version is available on 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

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

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.

Grolemund, G. & Wickham, H. (2011), 'Dates and times made easy with lubridate', *Journal of Statistical Software, Articles* **40**(3), 1–25.
URL: *https://www.jstatsoft.org/v040/i03*

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*

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*