Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

Calendar-based graphics for visualising people's daily schedules

Earo Wang
Monash University

Dianne Cook
Monash University

Rob J Hyndman
Monash University

Abstract

This paper describes a `frame_calendar` function that organises and displays temporal data, collected on sub-daily resolution, into a calendar layout. Calendars are broadly used in society to display temporal information, and events. The `frame_calendar` utilizes linear algebra on the date variable to create the layout. It utilizes the grammar of graphics to create the plots inside each cell, and thus synchronises neatly with **ggplot2** graphics. The motivating application is studying pedestrian behavior, based on counts which are captured at hourly interval by sensors scattered around the city. Facetting 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 visualisation, statistical graphics, time series, R package, grammar of graphics.

1. Introduction

This work was originally motivated by studying foot traffic in the city of Melbourne (City of Melbourne 2017). There have been 43 sensors installed that count pedestrians every hour across the downtown area up until the end of 2016 (see Figure 1). The dataset can shed light into understanding people's daily schedules, or assisting administration and business planning. We start off with the conventional time series plot to catch a glimpse of such data. A small multiples, shown in Figure 2, gives an overall picture of the foot traffic at 3 different sensors. Figure 3 provides more details of the temporal patterns by facetting on day of the week. The sensor data, like many temporal datasets on human behaviours, lends itself to a number of exploratory data visualisation challenges:

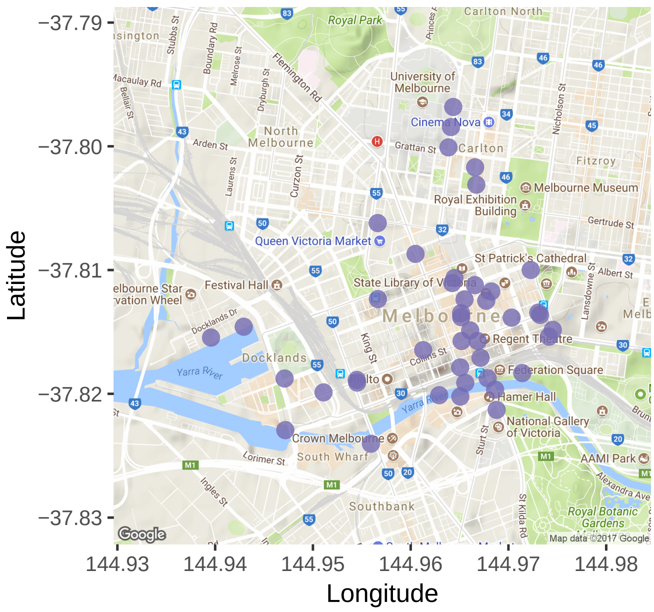

Figure 1: Map of the Melbourne city with purple dots indicating sensor locations.

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

A collection of data plots, organised in a familiar format, offer a useful and intuitive way to understand complex data. Wickham, Hofmann, Wickham, and Cook (2012) embedded a sequence of daily temperature line charts into a glyph map according to the spatial locations; and Hafen (2017) provided methods in the **geofacet** package to arrange data plots into a grid, while preserving the geographical layout. They both attempt to enable one to see the spatial context of each self-contained data plot.

Alternatively, calendar-based graphics turn out to be a useful tool in unfolding human-related activities over time. For example, Van Wijk and Van Selow (1999) developed a calendar view of the heatmap to represent the number of employees in the work place over a year, where colours indicate different clusters derived from the days. It contrasts weekdays and weekends, highlights public holidays, and presents other known seasons like school vacations; all of which have influence over the turn-outs in the office. The calendar-based heatmap was implemented in two R packages: **ggTimeSeries** (Kothari and Ather 2016) and **ggcal** (Jacobs 2017). However, these techniques are limited to colour-encoding graphics and are unable to use time scales smaller than day. Time of day, which serves as one of the most important aspects in explaining variations arisen from pedestrian sensor data, will be neglected through daily aggregation. Additionally if simply using coloured blocks rather than curves, it may become perceptually difficult to estimate the shape positions and changes, although using the curves comes with the cost of more display capacity (Cleveland and McGill 1984; Lam, Munzner, and Kincaid 2007).

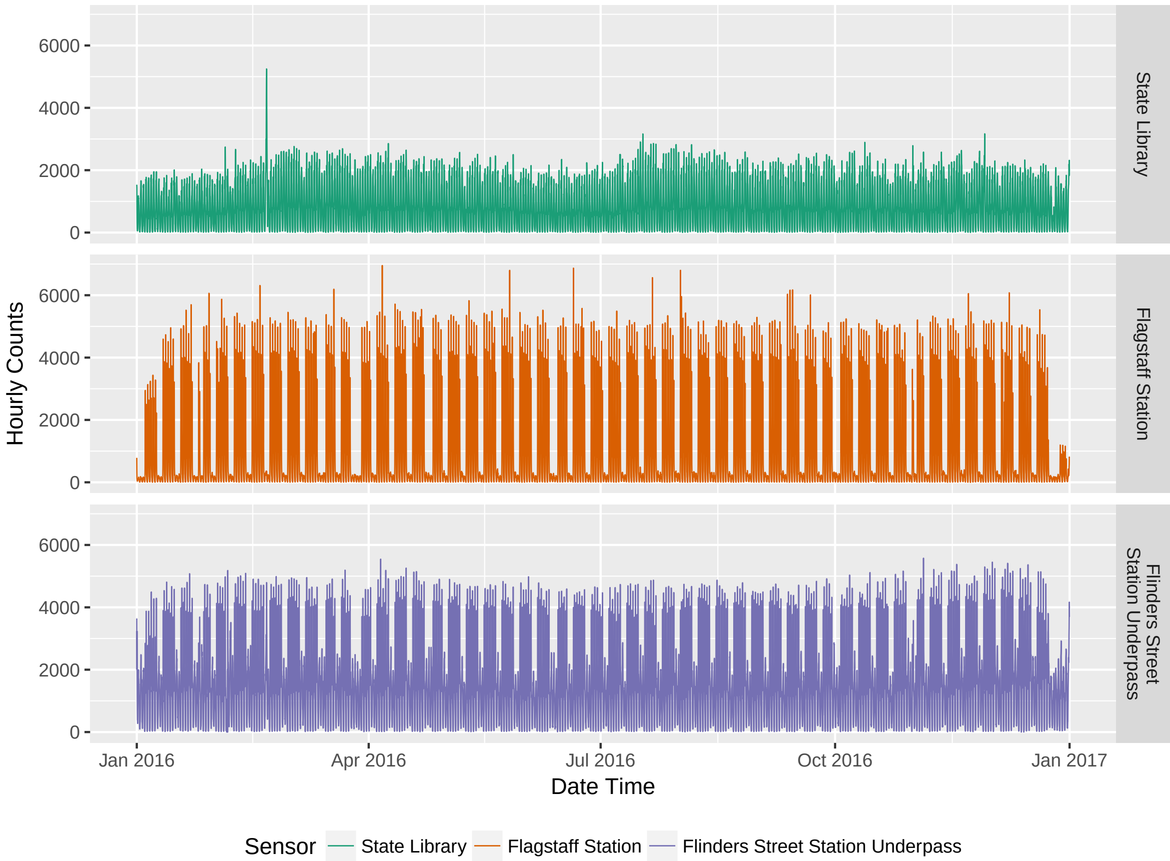

Figure 2: Time series plot about the number of pedestrians in 2016 measured at 3 different sensors in the city of Melbourne. Coloured by the sensors, small multiples of lines show that the foot traffic varies from one sensor to another in terms of both time and number. The weekly patterns look distincitve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

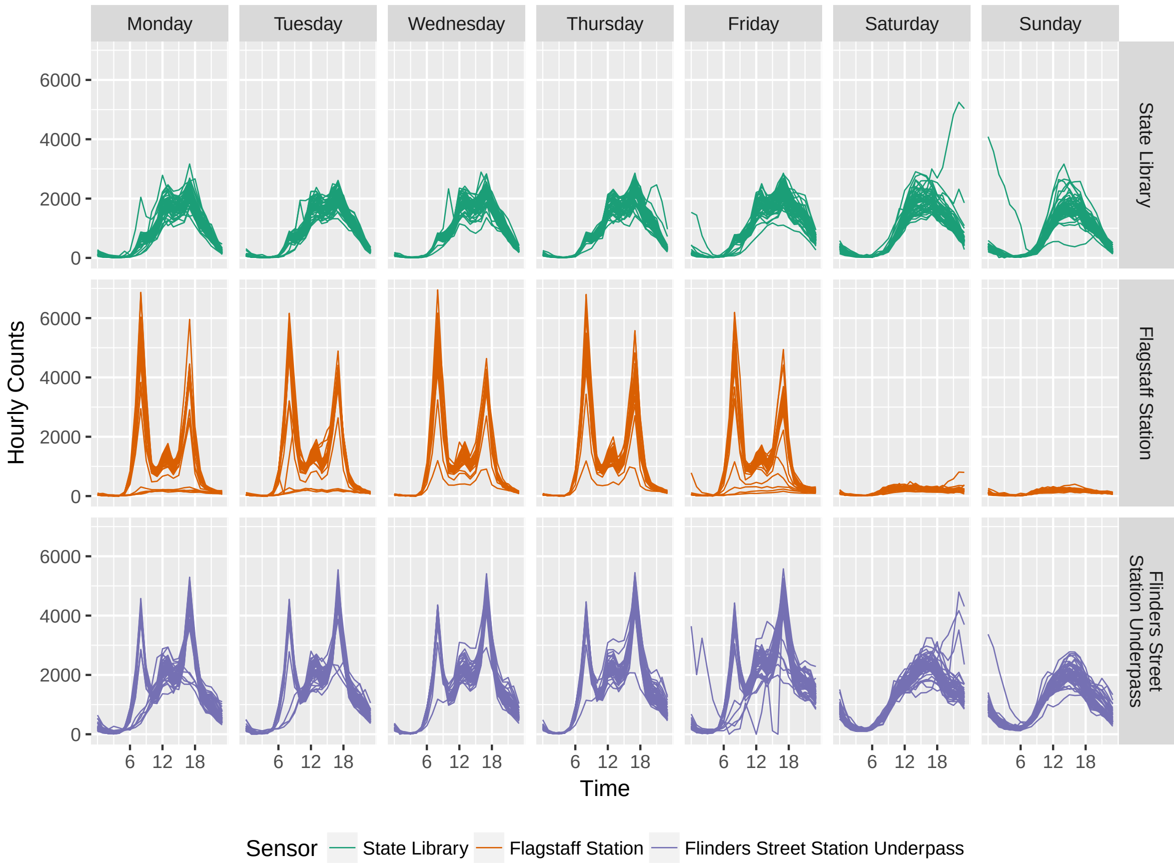

Figure 3: Hourly pedestrian counts facetted by sensors and days of the week using lines. It features at least two types of seasons–time of day and day of week–across all the sensors. The temporal patterns are subject to the sensor locations too.

We propose a new algorithm via linear algebra tools to go beyond the calendar-based heatmap. The approach is developed with these conditions in mind: (1) to make time of day present in addition to the existing 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 an overlaying plot consisting of multiple time series. The proposed algorithm has been implemented in the `frame_calendar` function in the **sugrrants** package (Wang, Cook, and Hyndman 2017) using R (R Core Team 2017).

The remainder of the paper is organised 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. Section 5 discusses the advantages and disadvantages of the method.

2. Construction

Figure 4 shows the line glyphs framed in the monthly calendar over the year of 2016. This is achieved by the `frame_calendar` function computing the new coordinates according to the input data variables; in turn the rearranged data values are plotted using the **ggplot2** package (Wickham and Chang 2016), which is an implementation of the grammar of graphics (Wilkinson 2006; Wickham 2010).

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, organised 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. 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 length days, ranging from 28-31, 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 $1 \leq i \leq 5$ is week within the month, $1 \leq j \leq 7$, 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

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

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, label this b. Figure 6 illustrates this layout.

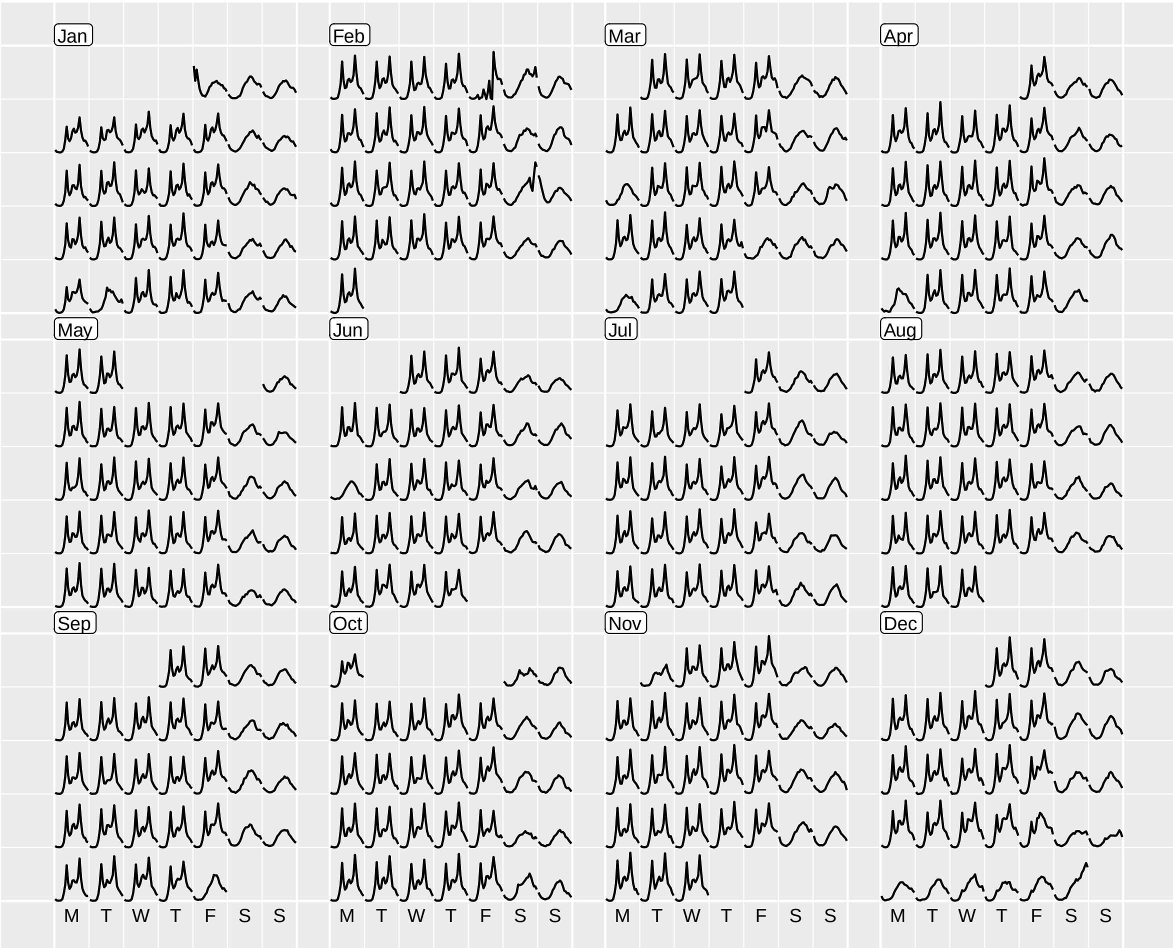

Figure 4: The calendar-based display of hourly foot traffic at Flinders Street Station using line glyphs. The arrangement of the data into a 3 by 4 monthly grid represents all the traffic in 2016. The disparities between weekday and weekend along with public holiday are immediately apparent.

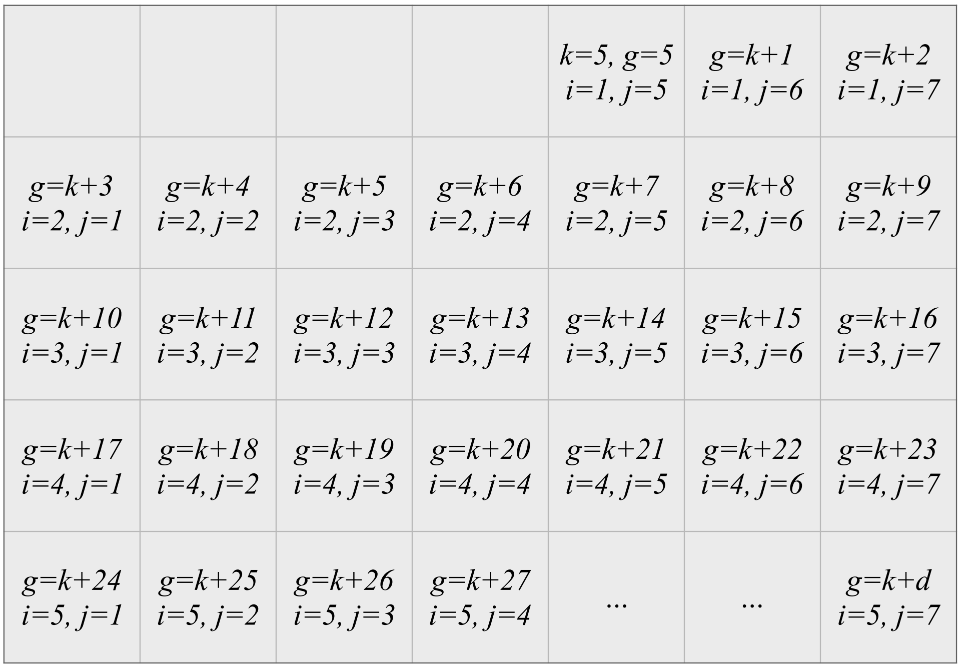

Figure 5: Illustration of the indexing layout for cells in a month.

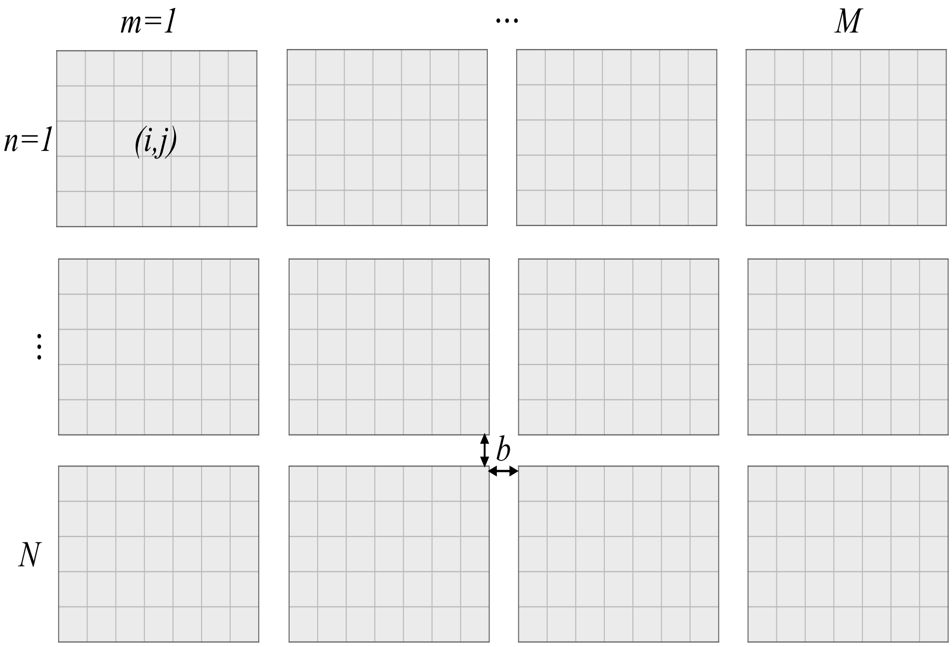

Figure 6: Illustration of the indexing layout for months of one year.

Each cell forms a canvas on which to draw the data. Consider the canvas to have limits $[0, 1]$ 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:

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

Note that for the vertical direction, the top left is the starting point of the grid which is why subtraction is performed. The resulting negative values place the cells down the vertical direction. Within each cell, the starting position is the bottom left.

Reference lines dividing each cell and block as well as labels indicating weekday and month are also provided in order to make calendar-based graphics more accessible and informative.

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 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 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 abbreviated month labels located on the top left are obtained through $(\min(x), \max(y))$ for every (m, n). The weekday texts with a single letter 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

There are several options provided for the `frame_calendar` function to initialize and adjust the display of a calander plot:

```
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
)
```

Assuming that tidy data (Wickham 2014) is the underlying data structure, the parameter `x` takes a variable that will be mapped to the x axis and the parameter `y` mapped to the y axis for plot construction. In Figure 4, for example, the `x` is the variable specifying the time of the day, and the `y` is the variable representing the hourly counts. The `date` argument is given by the date variable that determines the correct order of the calendar layout.

The algorithm can be extended to display data from a single month up to a few years. The number of rows and columns in the layout can be specified with the arguments `nrow` and `ncol`. If the one would like to visualise data spanning over three years, for example, `nrow =`

12 and `ncol = 3` would be an appropriate choice when assessing the differences of a given month across the years.

In Section 2 we only illustrated that grids are laid out horizontally. The vertical direction can be enabled by swapping i and j in equation 1, which occurs when the argument `dir` is set to `"v"`. This benefits users who are accustomed to calendars that are organised vertically, which is common in some countries. Next we shall describe some of the arguments that allow for alternate displays.

3.1. Layouts

The algorithm described in Section 2 is for the most common calendar layout, the "monthly" calendar, but it can be simplified to accommodate the other two types of calendar formats. One is comprised of days of a week in columns and weeks of a year in rows, and the other is days of a month in columns and months of a year in rows, which we refer to as the "weekly" and "daily" calendar, respectively. The layout to be used is controlled by the `calendar` argument. The weekly calendar puts more emphasis on days of a week over days of a year, whereas the daily calender does the opposite. The monthly calender can be considered as the advanced variant of both weekly and daily. Temporal patterns motivate which variant should be employed. The weekly calendar is appropriate if most of the variation can be characterised 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. When both effects are present, the monthly calendar would be a better choice.

3.2. Polar transformation

When `polar = TRUE`, the polar transformation is carried out on the data. The computation is similar to the one described in Wickham *et al.* (2012). Figure 7 is considered as the spiral display of Figure 4.

3.3. Scales

Section 2 discusses the implementation applied to a fixed scale, which uses the range over all data values. The `scale` argument controls the scaling of the display. The fixed scale (`fixed`) is the default, meaning that the data values in all positions are scaled. The remaining options 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.

Grouping the cells based on the same time period gives rise to the different scales. For example, the minimums and maximums obtained from each cell produce a local scale. The overall variation gives way to the individual shape. Figure 8 is an example of plotting line charts scaled locally. The daily variation is more distinctive compared to Figure 4.

Similarly, the same j indexing day of the week is grouped to compute the scales in days of a week; in other words, the same value within a given j corresponds to the same position across each j^{th} cell. To construct the scales for days of the month, g representing day of the month is used. This makes it easier to compare shape of a given day across each month block. The scaling option combined with the calendar layouts offers a number of varieties to construct the plot.

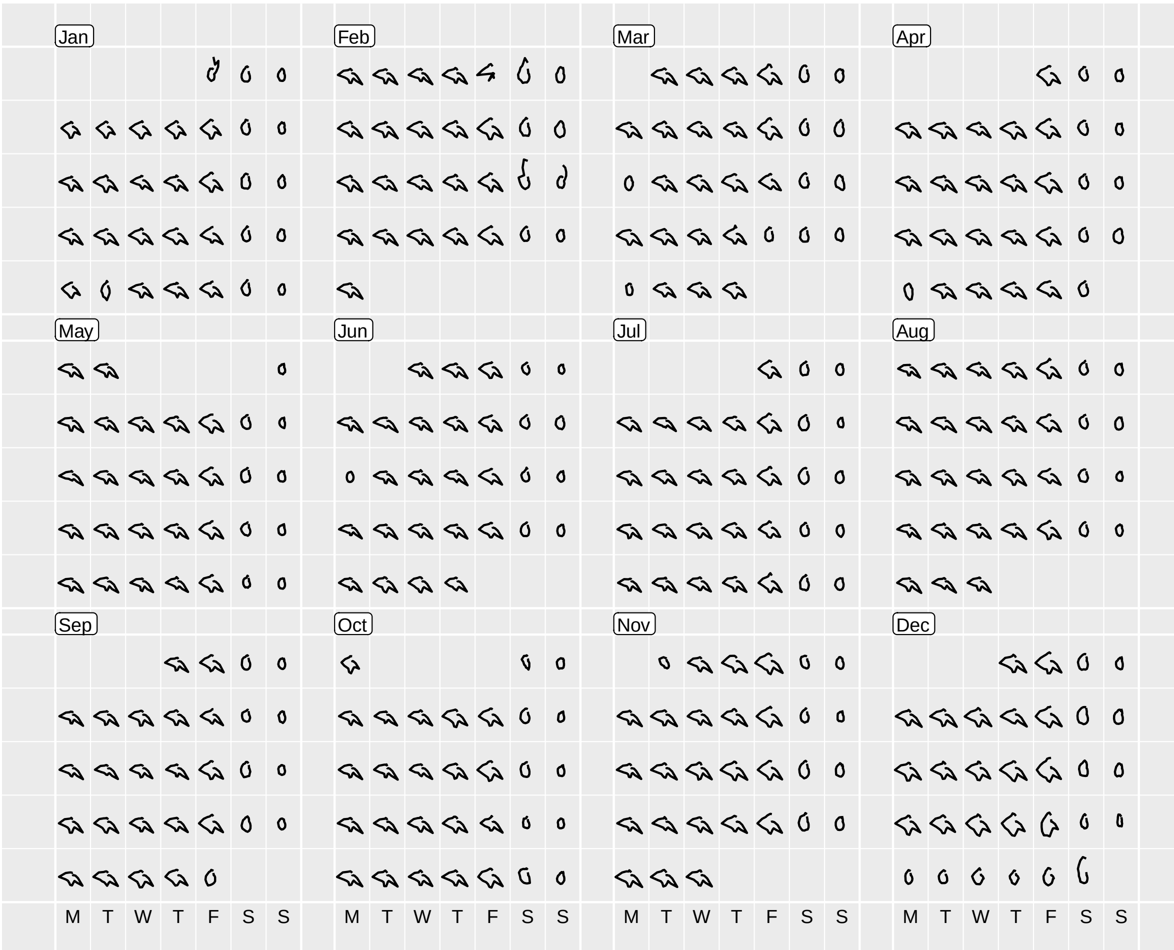

Figure 7: Star plots of hourly pedestrian counts at Flinders Street Station, which are line charts placed in polar coordinates. The periodic behaviours on workdays are clearly visible.

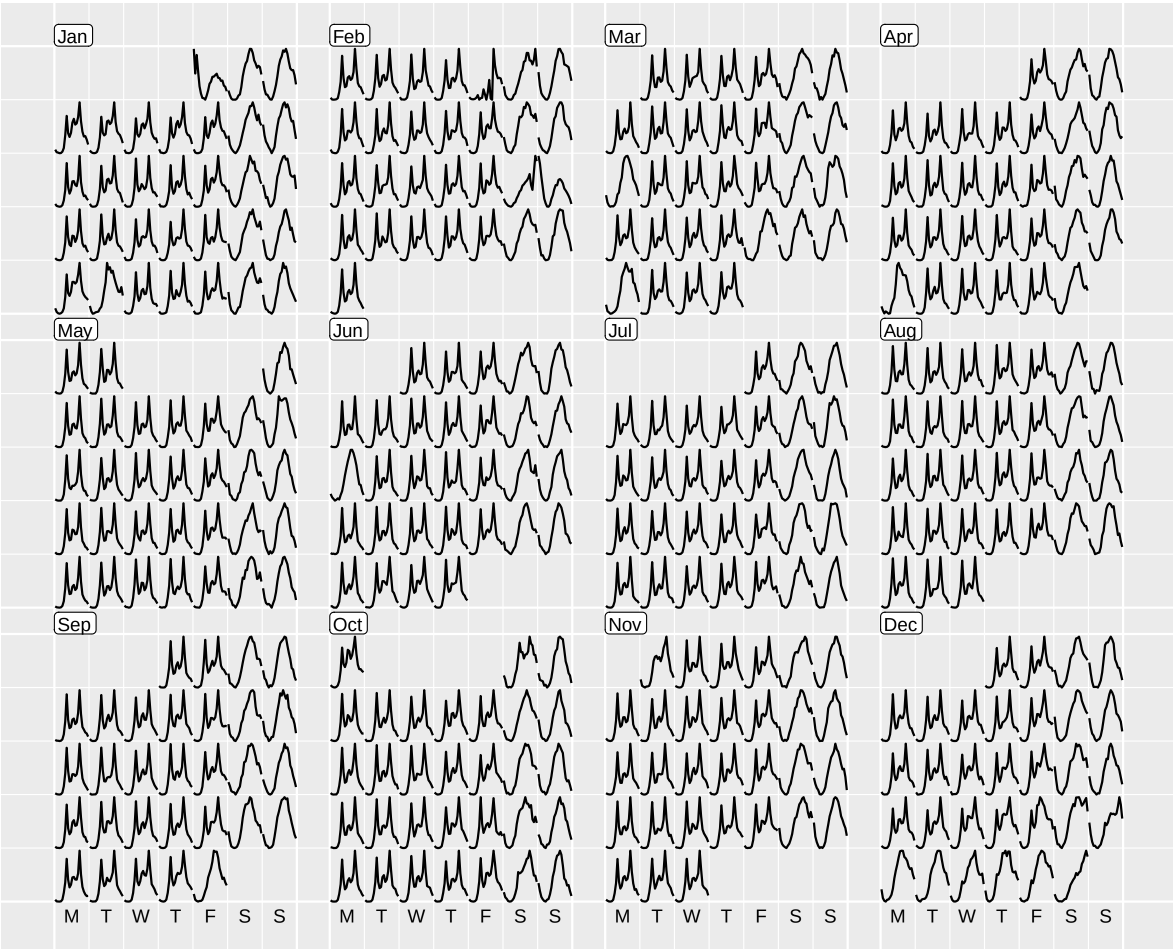

Figure 8: Line glyphs on the calendar format showing hourly foot traffic at Flinders Street Station, scaled over all the days. The individual shape on a single day becomes more distinctive, however it is more difficult to compare the size of peaks between days.

4. Variations

4.1. Overlaying and faceting subsets

The comparison of one sensor to another adds additional insights to the dataset, which is commonly done with an overlaying plot such as Figure 9. For instance, the dominant patterns that occurred at Flinders Street and Flagstaff train stations are driven by the commuters on work days; however, Flinders Street station has higher pedesterian counts during the weekends and public holidays. This suggests that Flagstaff station limits its functionality on non-work days, but various activities take place around the Flinders Street Station other than commuting. Figure 9 also shows that State Library follows a similar temporal trend as Flinders Street Station on non-work days. The nighttime events, such as White Night and New Year's Eve has barely affected the operation of Flagstaff Station but heavily affected the incoming and outgoing traffic to Flinders Street Station and 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 facetting calendar plots. This allows comparing the overall structure between sensors, while emphasising indivdual sensor variation.

4.2. Different types of plots

The `frame_calendar` function does not constrain itself to mapping only the temporal variable to the `x` argument. Figure 11 shows a lag scatterplot at Flinders Street Station, where the current hourly count is assigned to the `x` argument and the lagged hourly count to the `y` argument. This figure is organised in the daily calendar layout. It provides a visual tool for identifying repeating patterns. Figure 11 indicates two separate paths in the work-day glyphs, suggesting that an hour of a work day with many pedestrians is likely to be followed in the next hour with either greater or fewer numbers of pedestrians. However, this pattern is not present on non-work days. There is a bimodality on work days while there is unimodality on the rest of the days, which is also supported by Figure 4.

The newly computed coordinates can work with more complicated plots, such as the boxplot. Figure 12 uses the loess smooth line superimposed on side-by-side boxplots. It shows the distribution of hourly counts across all 43 sensors during December. In general, bimodality features on work days whereas unimodality features on the rest of the days. 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. Discussion

The calendar-based visualisation provides data plots in the familiar (at least for the Western world) format of an everyday tool. 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, rather than a typical work day.

This sort of layout may be useful for studying consumer trends, or human behaviour, like the pedestrian patterns. It may not work so well for physical patterns like temperature, which are not typically affected by human activity.

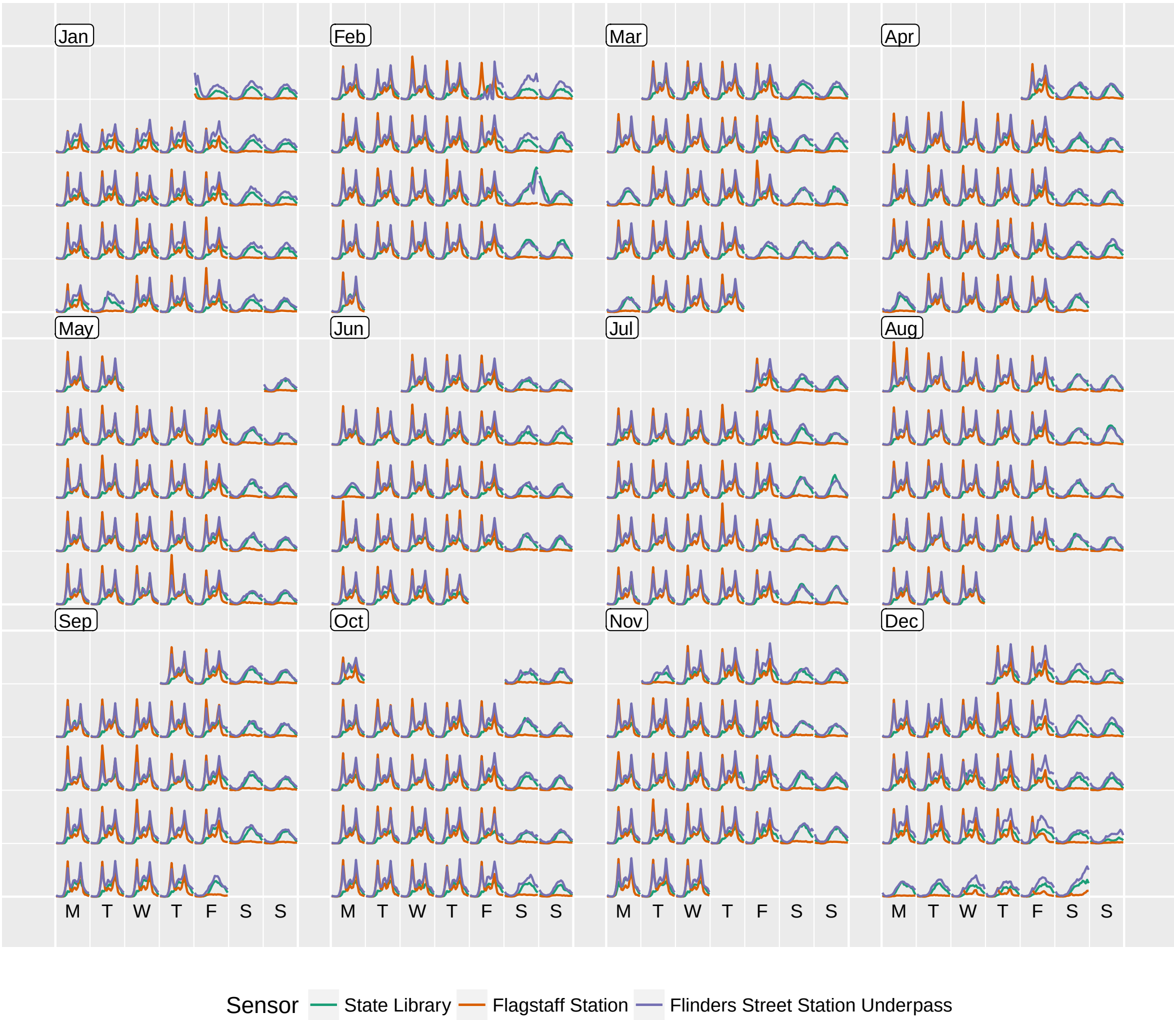

Figure 9: Overlaying line graphs of the 3 sensors in the monthly calendar. Flagstaff station is not as busy as the other two on non-work days.

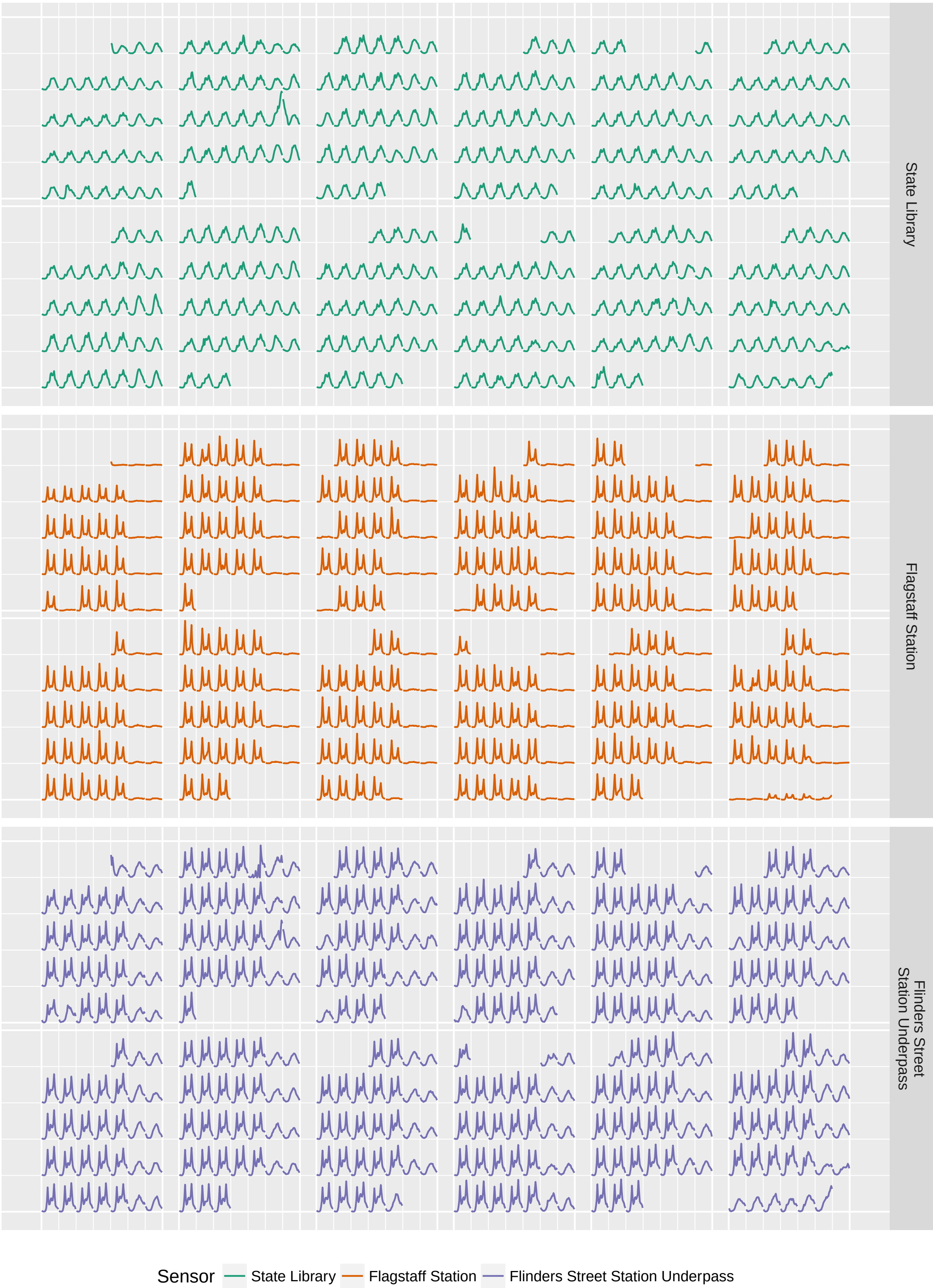

Figure 10: Line charts, embedded in the 6 by 2 monthly calendar, coloured and faceted by the 3 sensors. The variations of an individual sensor are emphasised, and the shapes can be compared across the cells and sensors.

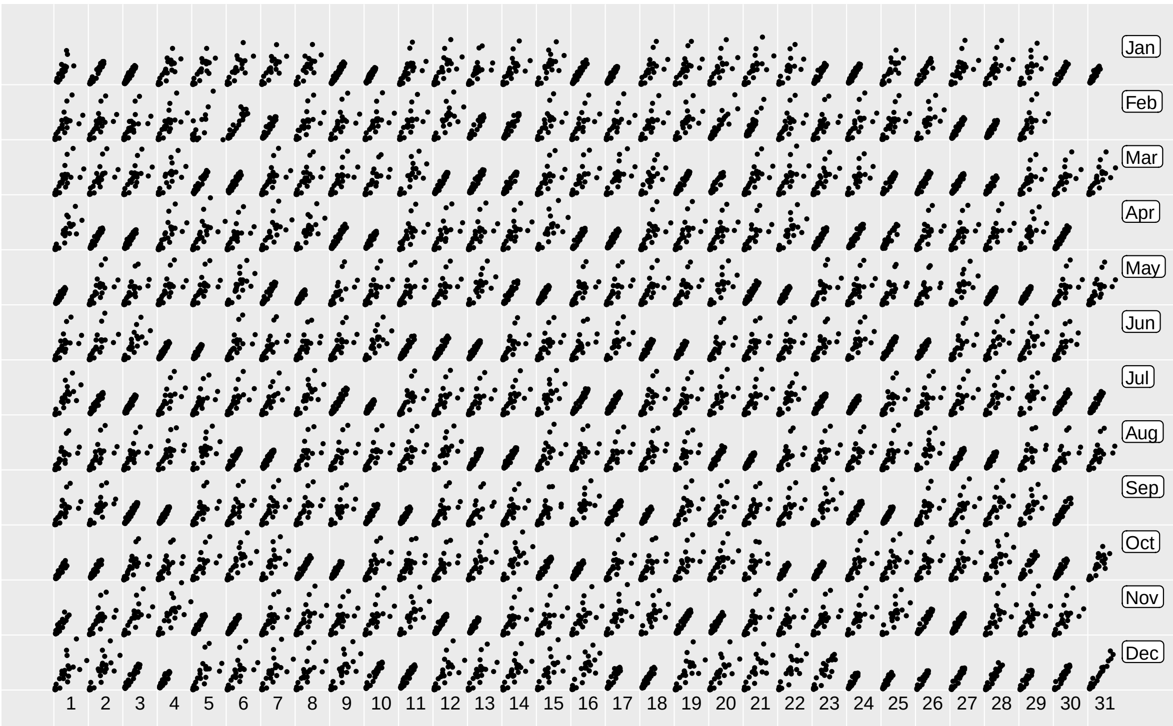

Figure 11: Lag scatterplot in the daily calendar layout. Previous hour's count is plotted against each hour's count at Flinders Street Station to demonstrate the autocorrelation at lag 1. The correlation between them is more consistent on non-work days than work days.

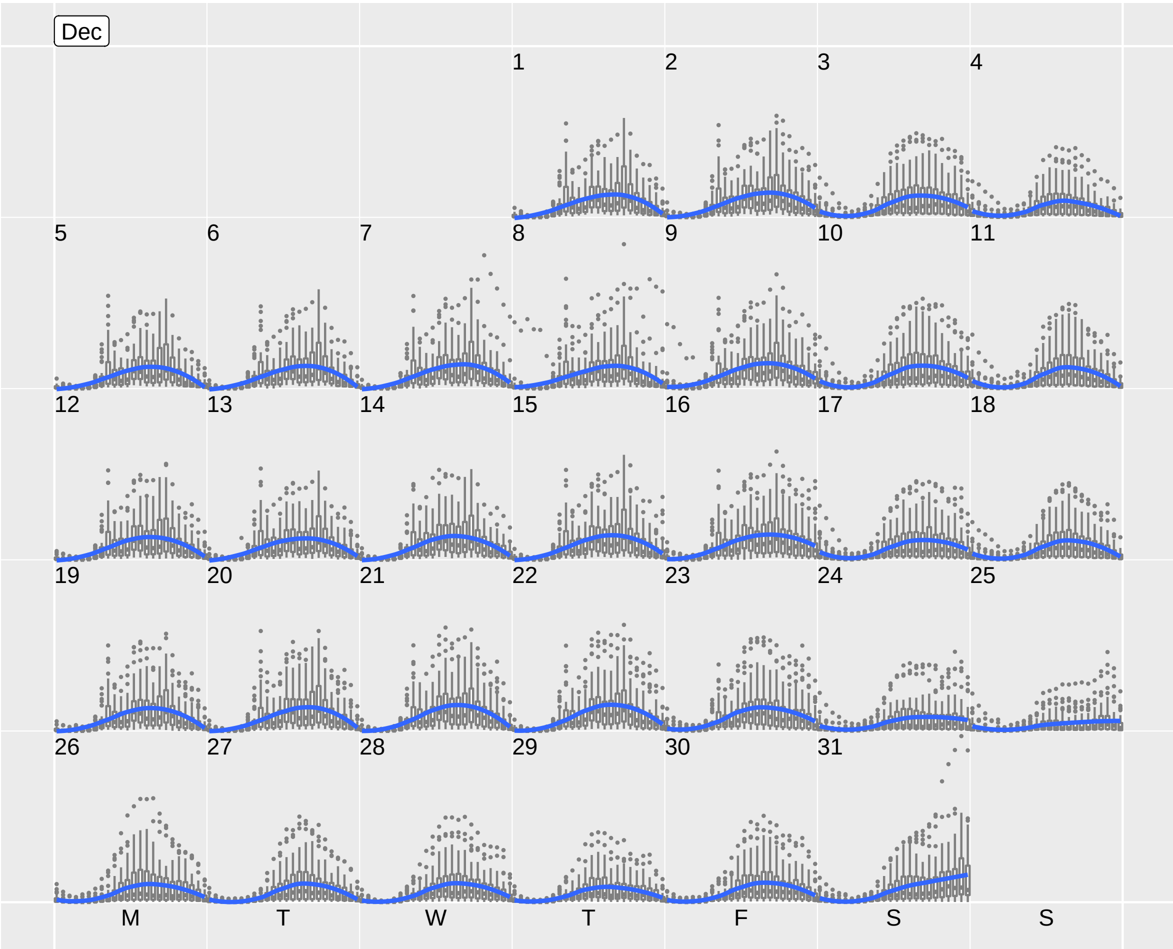

Figure 12: Side-by-side boxplot of hourly counts across all the 43 sensors in December of 2016, with the superimposing loess smooth line for each day. It shows the hourly distribution in the city as a whole. Christmas stands out as the quietest day in December.

References

City of Melbourne (2017). *Pedestrian Volumn in Melbourne*. Town Hall, 90-120 Swanston Street, Melbourne VIC 3000. URL http://www.pedestrian.melbourne.vic.gov.au.

Cleveland WS, 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 (2017). *geofacet: 'ggplot2' Faceting Utilities for Geographical Data*. R package version 0.1.4, 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.

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.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Van Wijk JJ, Van Selow ER (1999). "Cluster and Calendar Based Visualization of Time Series Data." In *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pp. 4–9. IEEE.

Wang E, Cook D, Hyndman R (2017). *sugrrants: Supporting Graphics with R for Analysing Time Series*. R package version 0.0.1.9000, URL http://pkg.earo.me/sugrrants.

Wickham H (2010). "A Layered Grammar of Graphics." *Journal of Computational and Graphical Statistics*, **19**(1), 3–28.

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

Wickham H, Chang W (2016). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 2.2.1, 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 (2006). *The Grammar of Graphics*. Springer Science & Business Media.

Affiliation:

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

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

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

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd