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

It is challenging to visualise people's daily schedules as such data are often of multiple seasons and of long time span. One example is Melbourne pedestrian sensor data which are collected at hourly interval and at a number of locations. Owing to human activities, the data features multiple temporal scales such as time of day, day of week, day of year. We develop the `frame_calendar` function that provides a method for rearranging such kind of data in a calendar-based format and accomplishing visualisation with the grammar of graphics. We describe its construction along with a few options. We provide some variations to demonstrate its usage by applying to the pedetrian data over one-year period of 2016.

Keywords: calendar-based visualisation, temporal-context data.

1. Introduction

This work was originally motivated by studying foot traffic in the city of Melbourne (City of Melbourne 2017). There have been 43 installed sensors counting pedestrians every hour across the downtown area until the end of 2016 (see Figure 1). The dataset can shed lights 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. Small multiples, shown as Figure 2, give an overall picture of foot traffic at a selection of 3 sensors. Figure 3, which is complementary to the Figure 2, provides another aspect of temporal patterns in the more detailed level by faceting day of the week. It lends itself to a number of issues that make exploratory data visualisation challenging in many temporal-context applications involving human behaviours:

1. Variations primarily result from multiple time scales including time of day, day of week, and day of year (such as public holiday and recurred events).

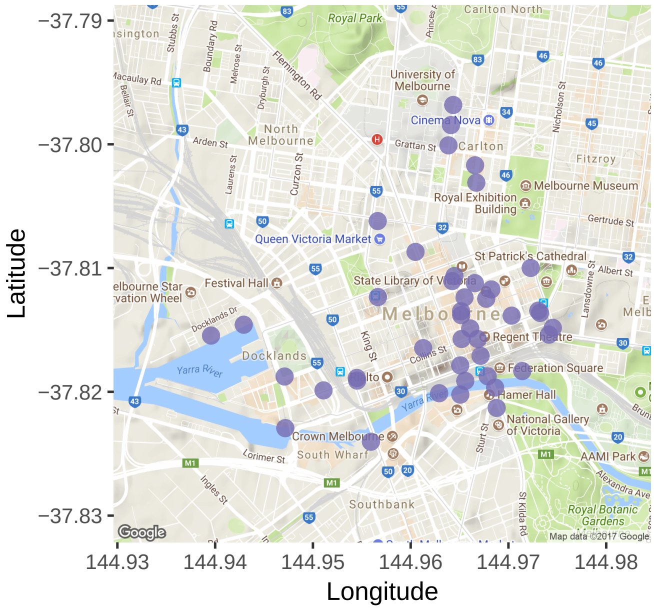

Figure 1: Map of the Melbourne city along with purple dots indicating sensor locations.

2. Since the data are often collected at daily frequency or a time scale more frequent than daily, they typically involve a large number of observations spanning over a long time period.
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.

Calendar-based graphics turn out to be a useful tool in unfolding human-related activities in temporal context. For example, Van Wijk and Van Selow (1999) developed a calendar view of heatmap to represent the number of employees in the work place over a year, where colours indicate different clusters derived from the days. It remarkably contrasts weekdays and weekends, highlights public holiday, 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 a couple of R packages: **ggTimeSeries** (Kothari and Ather 2016) and **ggcal** (Jacobs 2017). However, these techniques are too constrained to colour-encoding graphics and day of week as the smallest time scale. Time of day, which serves as one of the most important aspect 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 accurately judge the shape positions and changes, although the glyphs come with the cost of more display capacity (Cleveland and McGill 1984; Lam, Munzner, and Kincaid 2007).

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 as the `frame_calendar` function in the **sugrrants** package (Wang, Cook, and Hyndman 2017) using R (R Core Team 2017).

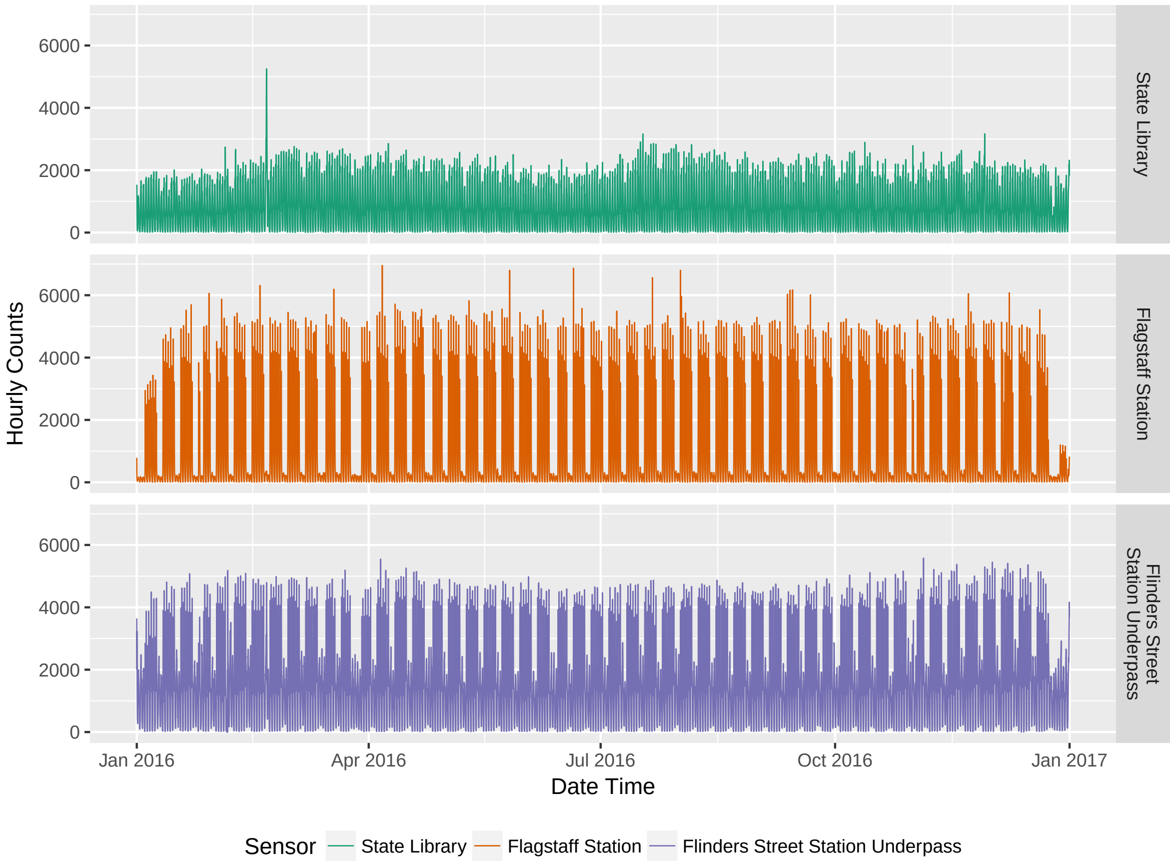

Figure 2: Time series plot about the number of pedestrians over the year of 2016 at a selection of 3 sensors in the city of Melbourne. Coloured by the sensors, small multiples of lines show that the foot traffic varies from one sensor to another in terms of both time and number. The weekly patterns look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

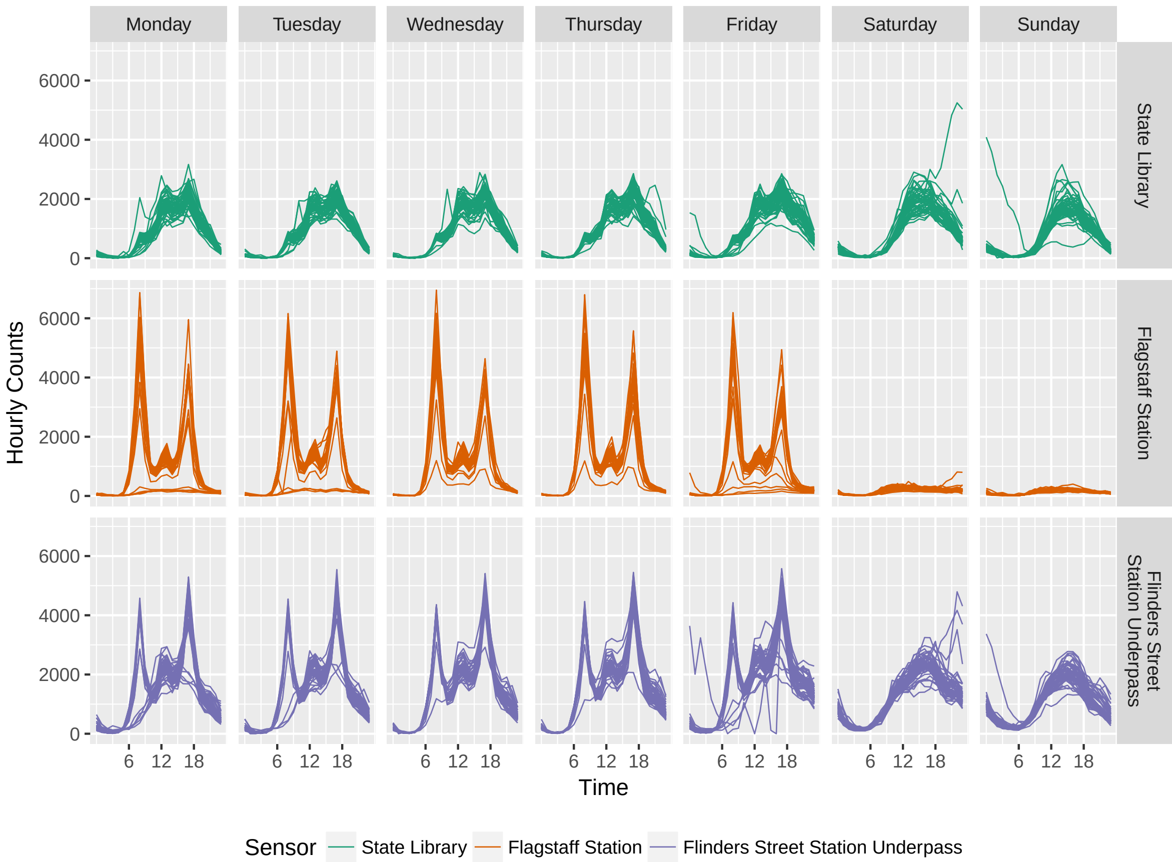

Figure 3: Hourly pedestrian counts facetted by sensors and days of the week using lines. It features at least two types of seasons–time of day and day of week–across all the sensors. The temporal patterns are subject to the sensor locations too.

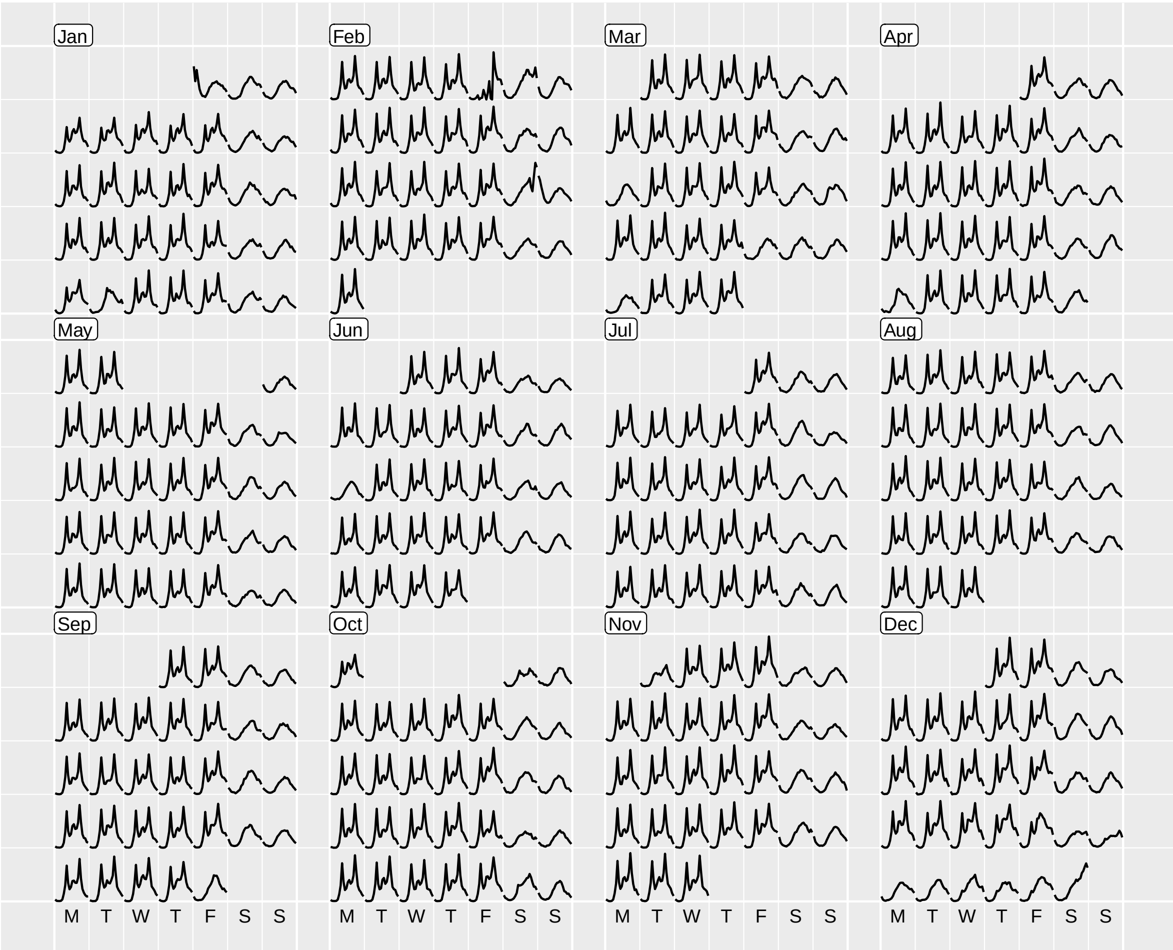

Figure 4: The calendar-based display of hourly foot traffic at Flinders Street Station using line glyphs. The arrangement of 3 by 4 on the monthly calendar format presents all the traffic in 2016. The disparities between weekday and weekend along with public holiday pop out to viewers in the clearer manner.

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 the 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

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

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

the glyph map displays for spatio-temporal data (Wickham, Hofmann, Wickham, and Cook 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 form 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:

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

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

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

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.

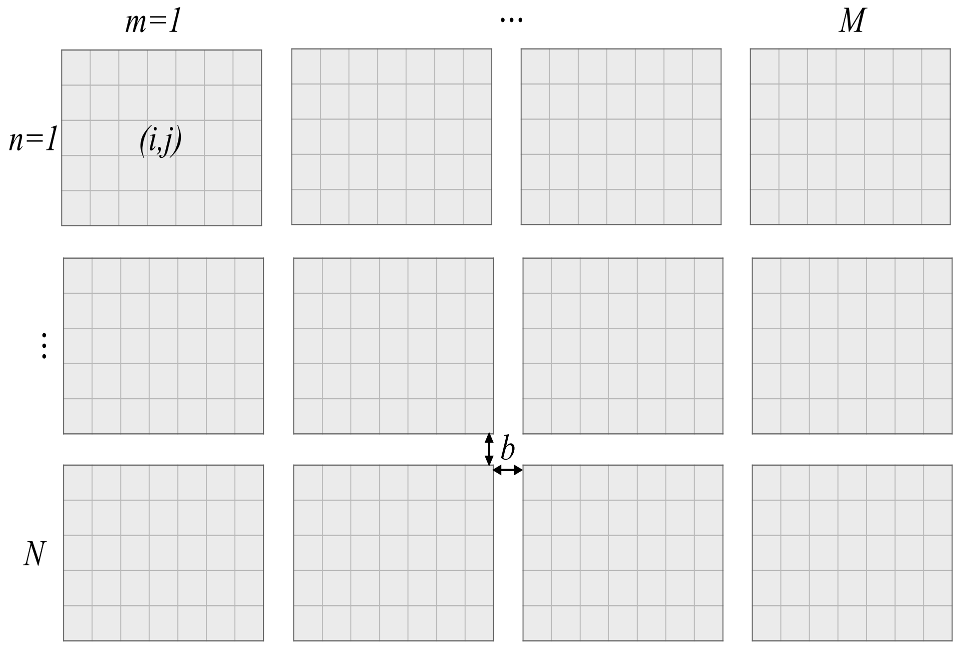

Figure 6: Illustration of the indexing layout for months of one year.

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 between $[0, 1]$, using the minimum and maximum of all the data values to be displayed assuming of 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, hence the subtractions, and resulting negative values to lay out the cells. Within each cell, the starting position is bottom left.

The algorithm can be relatively easily extended to layout from a single month to a few years, based on the period of time to be visualised. (M, N) can be determined by the user using the arguments of `nrow` and `ncol`. If the one would like to visualise data spanning over three years, for example, $M = 12$ and $N = 3$ seem to be an appropriate choice when assessing the differences of a given month across the years.

We only illustrate that grids are laid out over the horizontal direction in the paper. The vertical direction can be enabled by swapping i and j in the algorithm stated above when the argument `dir` is set to `"v"`. It particularly benefits those users who get accustomed to calendars of vertical organisation in some countries.

3. Options

There are a few options provided for the `frame_calendar` function to adjust the display and they are shown as follows:

```
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 underlining data structure, the `x` takes a variable that will be mapped to the x axis and the `y` mapped to the y axis for the later plotting. In Figure 4, for example, the `x` is the variable specifying the time of the day, and the `y` is the variable suggesting the hourly counts. The `date` argument is given by the date variable that determines the correct order in the calendar layout. We shall describe some of the arguments that allow for different displays.

3.1. Layouts

The algorithm described in Section 2 is for the most common calendar layout–"monthly" calendar, and 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 "weekly" and "daily" calendar respectively. Which layout to be used is controlled by the `calendar` argument. Due to the way of the arrangement, the weekly calendar puts more emphases on days of a week over days of a year, whereas the daily one serves as the opposite. The monthly format can be considered as the advanced twist of both weekly and daily. Temporal patterns lead to which format to be employed. The weekly calendar can be a nice attempt if the most variations can be characterised by days of a week. On the other hand, the absence of weekly effect but the presence of yearly effect can direct to the daily calendar. When both are present, the monthly calendar appears to be a better choice.

3.2. Polar transformation

When `polar = TRUE`, the polar transformation is carried out for the data. Again the computation remain 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 the fixed scale which is using the whole range of all the data values. The `scale` argument controls the scaling of the display. In addition to the fixed scale (`fixed`), the remaining options include free scale on all the days (`free`), day of the week only (`free_wday`), and day of the month only (`free_mday`) for different comparisons.

Grouping the cells according to the common period gives rise to these scales. The minimums and maximums obtained for every (i, j) and (m, n) allow for individual scale over all the single

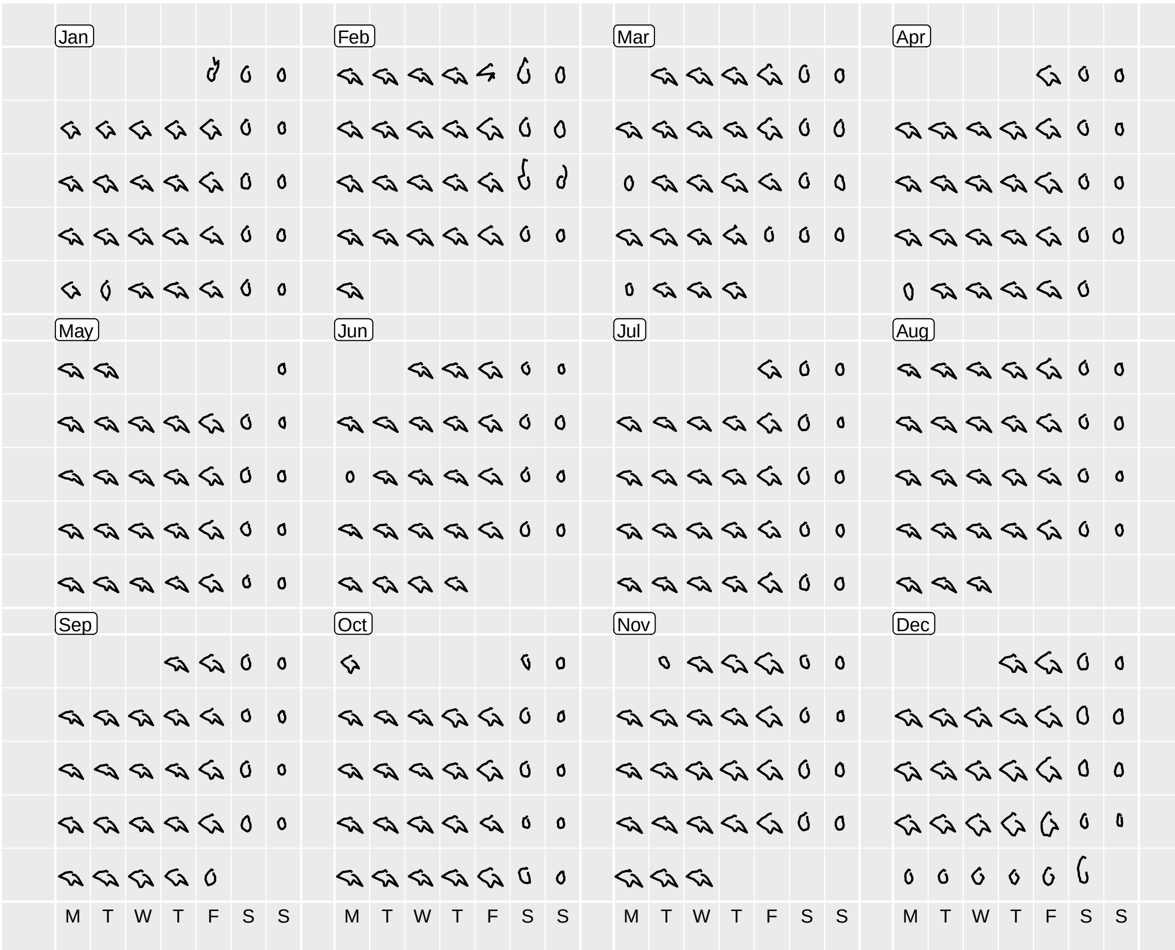

Figure 7: Line glyphs on polar coordinates.

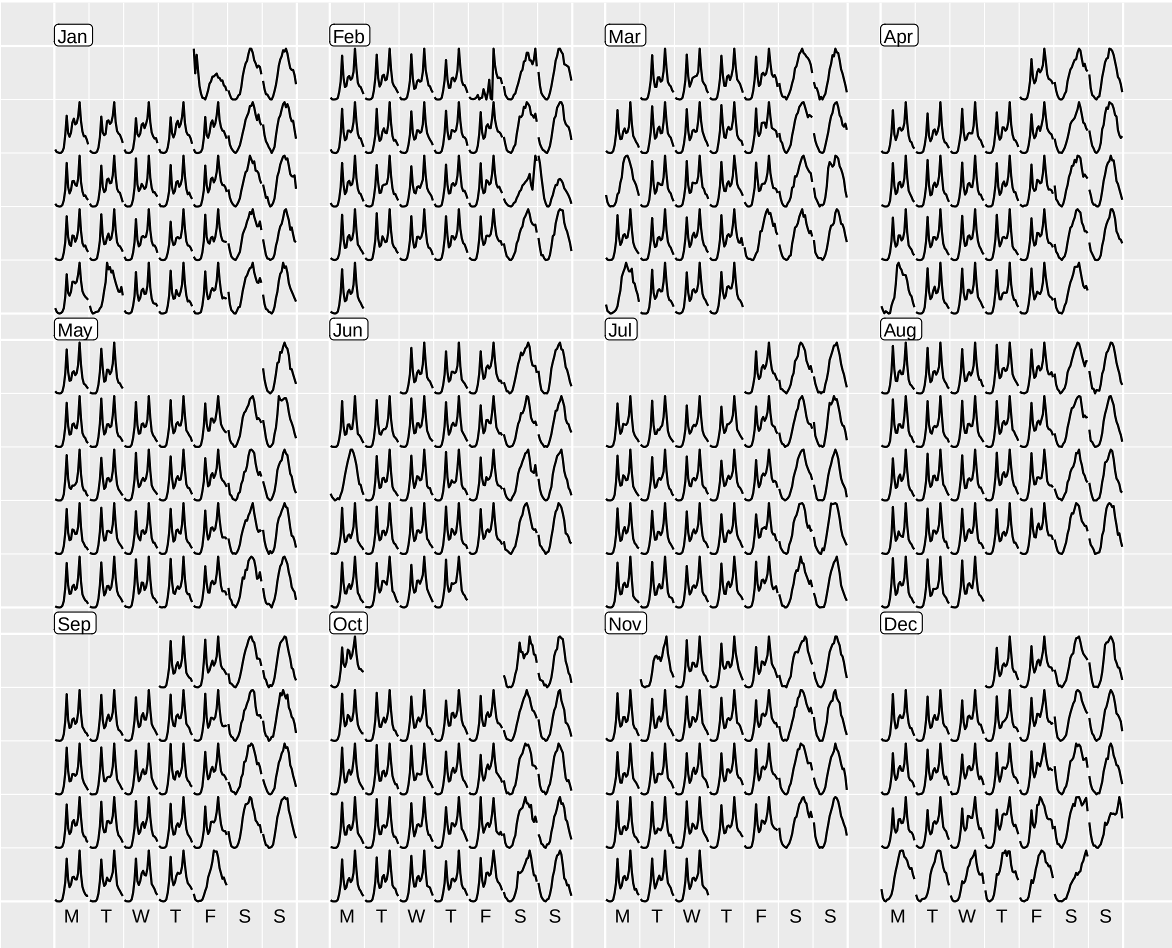

Figure 8: Line glyphs on the calendar format showing hourly foot traffic at Flinders Street Station, scaled over all the days. The individual shape on a single day becomes more distinctive at the expense of the magnitude comparison loss.

days. Similarly, j indexing day of the week is grouped to determine the scales on days of a week, and g representing the day in the month is for the scales on days of a month.

Figure 4 is drawn on the fixed scale so that it makes the magnitudes comparable over the whole period of time. On the other hand, some sub-series of small magnitudes yet worth-noting behaviours may become invisible whilst embedded in the overall picture. These sorts of scales, as a result, complements the calendar-based visualisation in general. Figure 8 provides an example of the free scale on all the days.

3.4. Reference lines and labels

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 placed on the bottom of the whole canvas, that is $\min(y)$, with the central position of a cell $x/2$ for each j.

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 occurred to both train stations–Flinders Street Station and Flagstaff station–are together driven by the commuters on the work days; however, the former is overwhelmed by the number of pedestrians during the weekends and public holidays, compared to the latter. It suggests that Flagstaff station limits its functionality as simply the public transport hub for day-to-day commuters, but various activities take place around the Flinders Street Station other than commuting. Because Flinders Street Station closely lies to South Bank and other places of attractions where the locals go for leisure and the tourists for travelling. It can be noted from Figure 9 that State Library follows the similar temporal trend as Flinders Street Station on non-work days. The nighttime events, such as White Night and New Year Evening Firework, has barely affected the operation of Flagstaff Station but heavily affected the incoming and outgoing traffic to Flinders Street Station and State Library on these days.

Placing multiple series on the common scale makes the magnitudes comparable between them; the glyphs are yet small in size, resulting in the overlapping problem. To avoid the problem, the calendar layout can be embedded into a series of plots for the different sensors. Figure 10 presents the idea of the graphs of calendar plots. By doing so, the comparability with respect to the magnitudes is sacrificed for the emphasis of the individual sensor.

4.2. Different types of plots

The `frame_calendar` function does not constrain itself to mapping only temporal variable to the `x`. Figure 11 shows the lag scatterplot, where the current hourly count is assigned to the `x` and the lagged hourly count to the `y` at Flinders Street Station, organised in the daily calendar. It provides a visual tool for identifying repeating patterns. Figure 11 indicates two separate paths in the work-day glyphs, which suggests that an hour of a work day with many pedestrians is likely to be followed by the time of either more pedestrians or fewer. Quite the contrary, the relationship is so consistent on the non-work days. This is a clear sign of bimodalily on work days whereas unimodality on the rest of the days, which is also supported by Figure 4.

The newly computed coordinates not only work with the simple geoms such as lines and points, but also boxplot and other complex geoms. Figure 12 uses the side-by-side boxplots with the loess smooth line superimposed as an example. It shows the distribution of hourly counts across all the 43 sensors. In general, bimodality features work days whereas unimodality

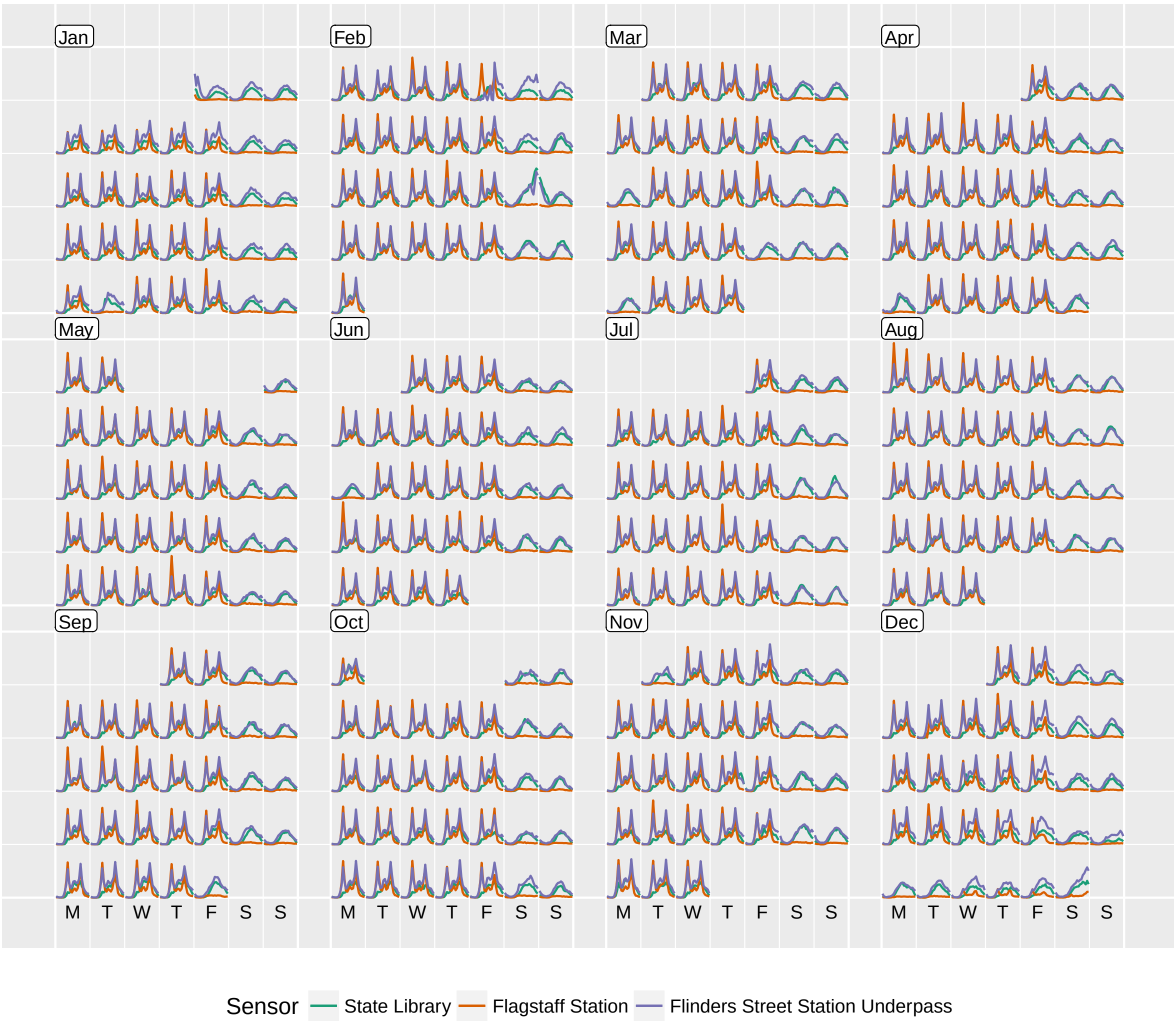

Figure 9: Overlaying line graphs of the 3 sensors in the monthly calendar. Flagstaff station is not as busy as the other two on non-work days.

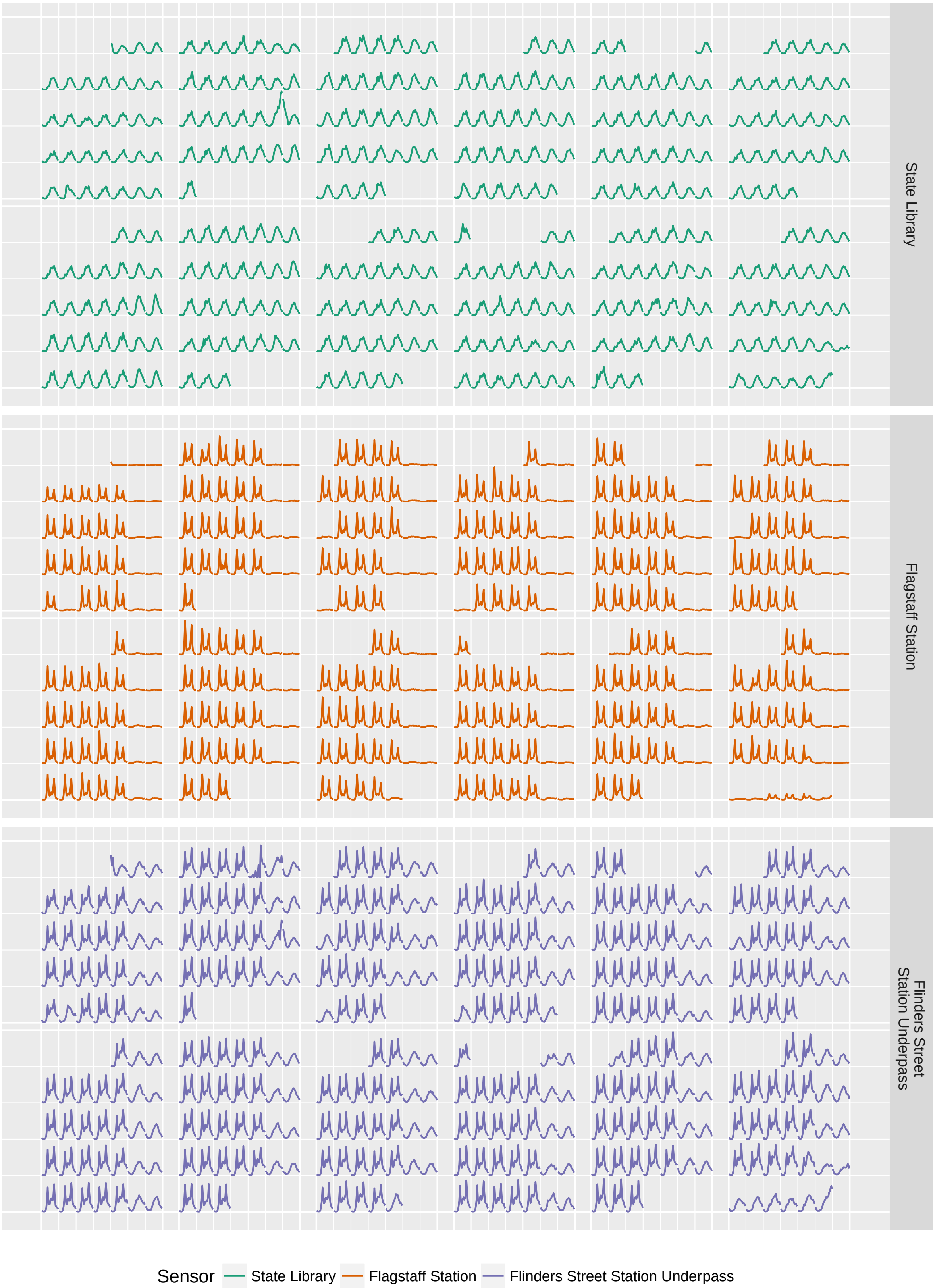

Figure 10: Line graphs faceted by the 3 sensors in the 6 by 2 monthly calendar.

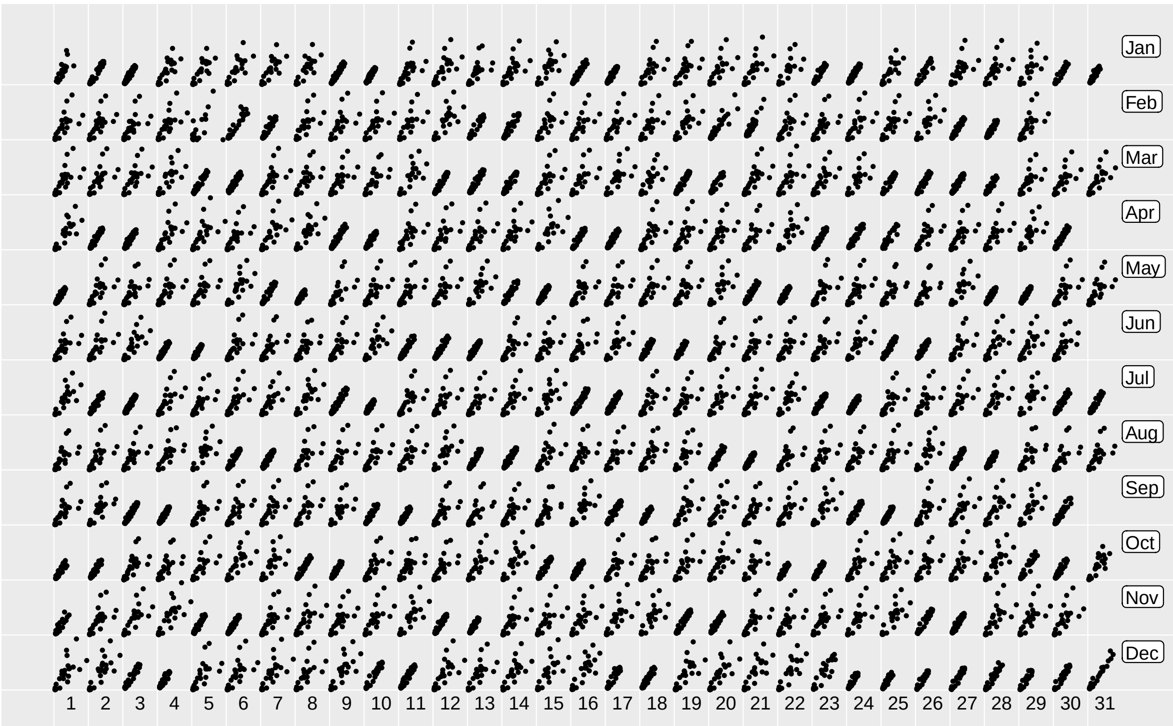

Figure 11: Lag scatterplot in the daily calendar layout. Previous hour's count is plotted against each hour's count at Flinders Street Station to demonstrate the autocorrelation at lag 1. The correlation between them is more consitent on non-work days than work days.

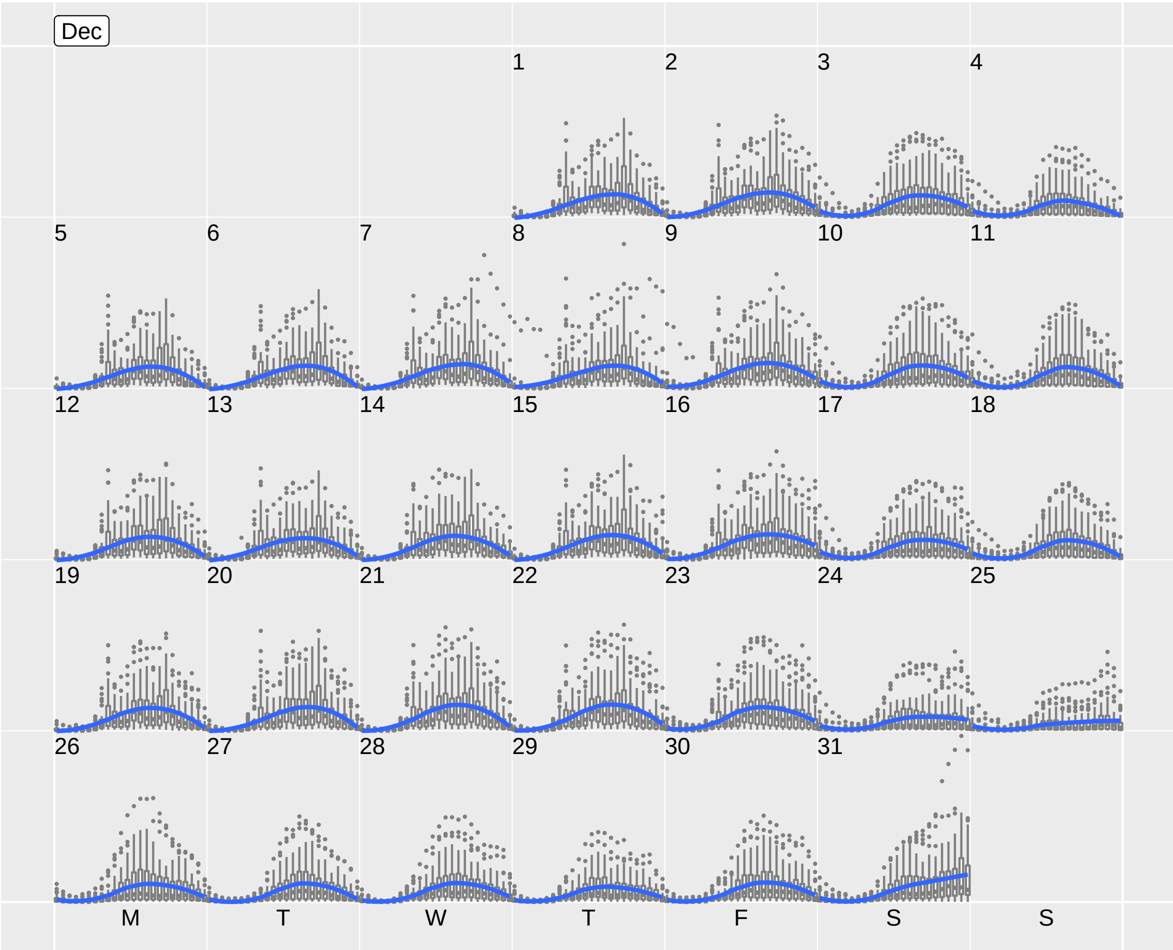

Figure 12: Side-by-side boxplot of hourly counts across all the 43 sensors in December of 2016, with the superimposing loess smooth line for each day. It shows the hourly distribution in the city as a whole. Christmas stands out as the quietest day in December.

features the rest of the days. The last week of December is just the holiday season: people are off work on the day before Christmas, go shopping on the Boxing day, and hanging out for the firework in the New Year Evening.

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.

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.

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