Tidy data structure and visualisation to support exploration and modeling of temporal data

A thesis submitted for the degree of

Doctor of Philosophy

by

Earo Wang

B.Comm. (Hons), Monash University

Department of Econometrics and Business Statistics

Monash University

Australia

February 2019

Contents

Acknowledgements iii

Declaration v

Preface vii

Abstract ix

1 Introduction 1

1.1 Research framework 1

1.2 Scope 3

2 Calendar-based graphics for visualizing people's daily schedules 5

Abstract 5

2.1 Introduction 6

2.2 Creating a calendar display 10

2.3 Case study 25

2.4 Discussion 33

Acknowledgements 33

Bibliography 35

Acknowledgements

This document was created with bookdown (Xie, 2016), and the raw files contain all the R code to produce the plots and tables.

Declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any university or equivalent institution, and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Earo Wang

Preface

Chapter 2 has been submitted to *Journal of Statistical Software* and is currently under review. It has won the 2018 ASA Statistical Graphics Student Paper Award.

Abstract

Temporal-context data sets often include a richness of information that is not possible to include in the typical data formats that are used in time series analysis. They also present some complications for modelling and visualisation, such as a long time spans, multiple factor variables, heterogeneous data types, low time resolutions, implicit missing values, and multilevel temporal components. In this thesis, we extend the conceptual tidy data framework, which provides the foundation for good modern data practice, to encompass time series data, and develop new graphics for displaying temporal data. First, we develop a new calendar view for visualising sub-daily time series data, which is particularly useful for viewing people patterns. Second, we have extended the tidy data concept to temporal data, and note that the "molten" data structure is flexible enough to handle the full richness of these complex data sets, heterogeneous variables, missing values. The tidy temporal data also builds on ideas of a data pipeline which improves and supports an organised worflow for data analysis. The third topic will focus on providing a visualisation framework for time series with nested and crossed factors. All the methods have been implemented in the R packages **sugrrants** for visualisation and **tsibble** for tidy time series data.

Chapter 1

Introduction

The conventional plot for time series is a line plot, where the measured variable is plotted against a time variable, and consecutive points are connected by lines. The line plot assumes: (1) there are no missing values, (2) the series is not too long so that seasonality can be viewed along with trend, (3) the time series stands alone without any complementary information, and (4) there is a single measured variable. The problem is that data rarely comes in this form. Quite often the temporal component is just one aspect of multi-facetted data, some observations are missing, the series could be very long, so that seasonality is lost relative to any long-term trend, measurements might be taken at irregular intervals, and we may have a big collection of time series. This thesis addresses these broader issues of better data structures and visualisation for big temporal-context data.

1.1 Research framework

Data visualisation in statistics has evolved in recent years to have graphics formally described using a grammar (Wilkinson, 2005**wickham2010layered**), and also data structures for statistical analysis have been organised into a conceptual framework of "tidy data" (Wickham, 2014). Both aspects are critical for evaluating the strength of the research, and for building on a foundation of sound data practice.

1.1.1 Tidy data structure

Wickham (2014) developed a set of tidy data principles to standardise and facilitate the data analysis process, and also suggested that each variable is a column, each observation is a row. An attribute describing a unit's properties (such as temperature, precipitation, pedestrian counts) is thought of as a variable. An observation refers to the same unit measured across each variable. A variable together with an observation defines the values that essentially comprise of a dataset. The newer concept of tidy data is the "long" form: every measured value is described by a unique set of identifiers. From the long form, data can be summarised, and shaped, and arranged in many different forms, making different types of analysis on the same data easier.

However, in some ways it is the classical statistical format of a rectangular table where each time series is a column, and each time index is a row. This format is the "wide" form of tidy data, and it is what is typically required for many statistical models as it proves computationally efficient when doing matrix operations.

The common data structure for time series data in statistical software, like R (**r**), is actually short-hand. For example, a time series defined as a `ts` object in R is a vector of values, annotated with a starting time and frequency. Associated with the data object are methods that can be applied to plot, summarise and model. However, this approach is model-centric rather than data-centric. Data typically doesn't come in this form. Data typically has a lot more with it, and to get it in this form requires extracting a small part of all the data, and maybe, even massaging it into regular time measurements. Working from a tidy format this specialist form could be created with several wrapper functions, whilst maintaining the connections to the complete data.

1.1.2 The grammar of graphics

A grammar of graphics was first proposed by Wilkinson (2005) and extended by **wickham2010layered**. These methods establish a conceptual framework for mapping data to graphical form. It is the analogue to thinking of statistics as a mapping of random

variables. For example, a mean is the mapping of n independent and identically distributed random variables, $X_1, \ldots, X_n$, $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. With this functional mapping, the statistic can be computed on a sample, and the properties of the statistic can be analytically studied.

The grammar formulates a set of rules to map variables to visual quantities. Therefore, we can make the comparisons between different displays and describe the differences under a standard framework. The grammar enables one to understand the process of creating visual graphics and the data structure behind the statistical plots. It is efficient and replicable to make any type of graphic display. The R package **ggplot2** (Wickham et al., 2018) is an implementation of the grammar of graphics.

1.2 Scope

This thesis research proposes a data-centric structure to represent time series by extending the "tidy data" principles. It allows rich data information to be included, which is not currently supported by time series model objects. It also provides a mapping bridging the semantics of a dataset to its physical representation, and consequently fosters ideas of a data pipeline that supports thinking of operations on the data variables. A collection of verbs are developed to smooth out the workflow for analysis of temporal data.

A new calendar-based graphics have been created and implemented to better visualise sub-daily data related to human behaviours. Tidy temporal data builds the foundation of calendarising the data using the data re-structuring approach. The grammar of graphics further adds the plotting capacities to the calendar format.

Tidy temporal data also integrates nested and crossed factors. A visualisation framework will be proposed and formulated for time series with nested and crossed factors.

Chapter 2

Calendar-based graphics for visualizing people's daily schedules

Abstract

Calendars are broadly used in society to display temporal information, and events. This paper describes a new R package with functionality to organize and display temporal data, collected on sub-daily resolution, into a calendar layout. The function `frame_calendar` uses linear algebra on the date variable to restructure data into a format lending itself to calendar layouts. The user can apply the grammar of graphics to create plots inside each calendar cell, and thus the displays synchronize neatly with **ggplot2** graphics. The motivating application is studying pedestrian behavior in Melbourne, Australia, based on counts which are captured at hourly intervals by sensors scattered around the city. Faceting by the usual features such as day and month, was insufficient to examine the behavior. Making displays on a monthly calendar format helps to understand pedestrian patterns relative to events such as work days, weekends, holidays, and special events. The layout algorithm has several format options and variations. It is implemented in the R package **sugrrants**.

2.1 Introduction

We develop a method for organizing and visualizing temporal data, collected at sub-daily intervals, into a calendar layout. The calendar format is created using linear algebra, giving a restructuring of the data, that can then be piped into grammar of graphics definitions of plots, as used in **ggplot2** (Wickham et al., 2018). The data restructuring approach is consistent with the tidy data principles available in the **tidyverse** (Wickham, 2017) suite. The methods are implemented in a new package called **sugrrants** (Wang, Cook, and Hyndman, 2018).

The purpose of the calendar-based visualization is to provide insights into people's daily schedules relative to events such as work days, weekends, holidays, and special events. This work was originally motivated by studying foot traffic in the city of Melbourne, Australia (City of Melbourne, 2017). There have been 43 sensors installed that count pedestrians every hour across the inner-city area till the end of 2016 (Figure 2.1). 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 2.2, give an overall picture of the foot traffic at three different sensors over 2016. Further faceting by day of the week (Figure 2.3) provides a better glimpse of the daily and sub-daily pedestrian patterns.

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 (2018) which provides methods in the **geofacet** package to arrange data plots into a grid, while preserving the geographical position. Both of these show data in a spatial context.

In contrast, calendar-based graphics unpack the temporal variable, at different resolutions, to digest multiple seasonalities, and special events. There is some existing work in this area. For example, Van Wijk and Van Selow (1999) developed a calendar view of the

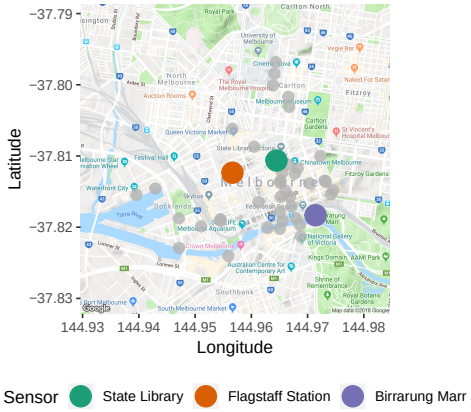

Figure 2.1: *Map of the Melbourne city area with dots indicating sensor locations. These three highlighted sensors will be inspected in the paper: (1) the State Library–a public library, (2) Flagstaff Station–a train station, closed on non-work days, (3) Birrarung Marr: an outdoor park hosting many cultural and sports events.*

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. 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 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 and McGill, 1984; Lam, Munzner, and Kincaid, 2007).

We propose a new algorithm to go beyond the calendar-based heatmap. The approach is developed with three conditions in mind: (1) to display time-of-day variation in addition to longer temporal components such as day-of-week and day-of-year; (2) to incorporate line graphs and other types of glyphs into the graphical toolkit for the calendar layout; (3) to enable overlaying plots consisting of multiple time series. The proposed algorithm has been implemented in the `frame_calendar` function in the **sugrrants** package using R.

Figure 2.2: *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.*

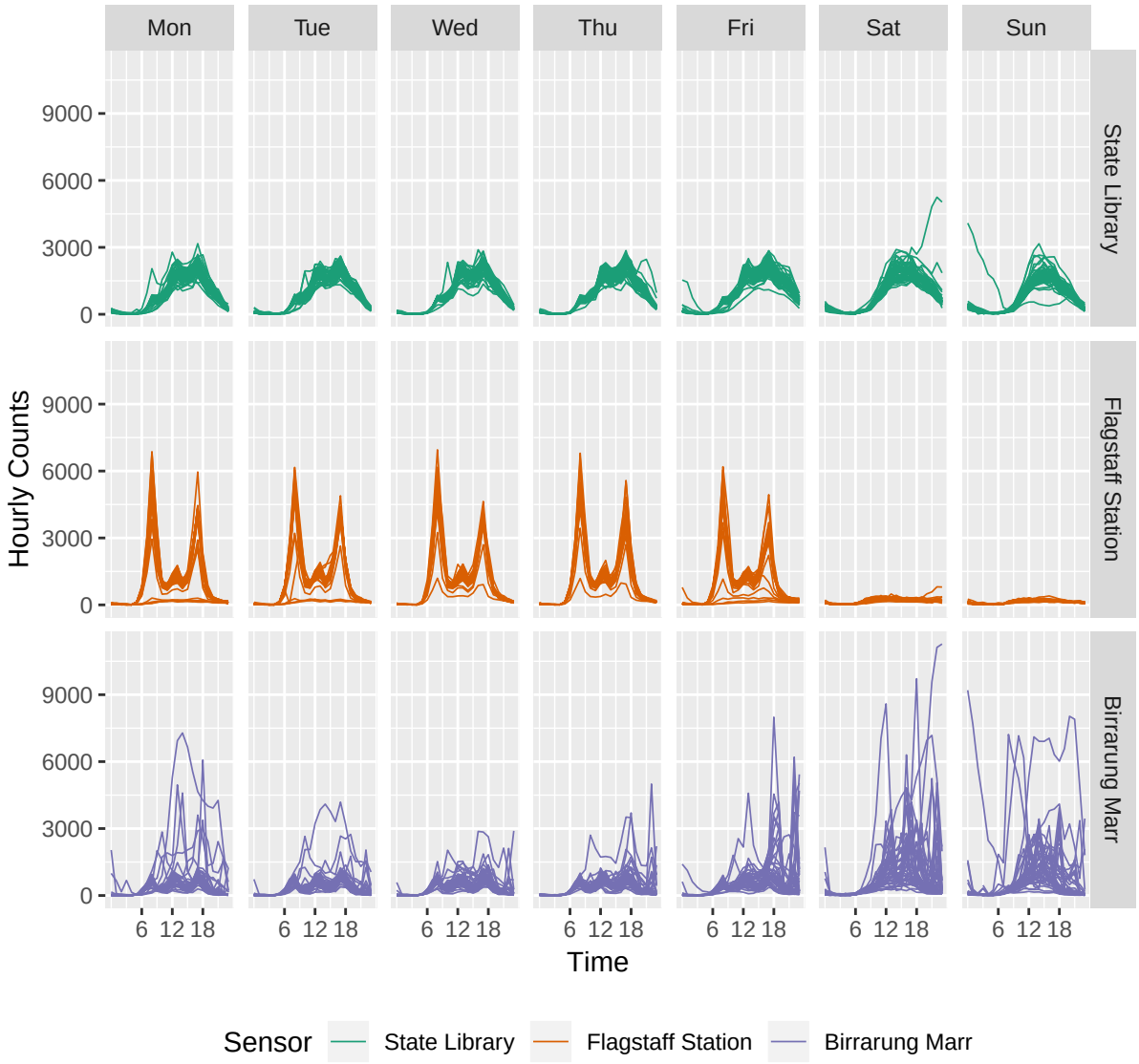

Figure 2.3: *Hourly pedestrian counts for 2016 faceted by sensors and days of the week using lines. It primarily features two types of seasons—time of day and day of week—across all the sensors. Apparently other factors have influence over the number of pedestrians, which cannot be captured by the faceted plots, such as the overnight White Night traffic on Saturday at the State Library and a variety of events at Birrarung Marr.*

The remainder of the paper is organized as follows. Section 2.2 demonstrates the construction of the calendar layout in depth. Section 2.2.1 describes the algorithms of data transformation. Section 2.2.2 lists and describes the options that come with the `frame_calendar` function. Section 2.2.3 presents some variations of its usage. Graphical analyses of sub-daily people's activities are illustrated with a case study in Section 2.3. Section 2.4 discusses the limitations of calendar displays and possible new directions.

2.2 Creating a calendar display

2.2.1 Data transformation

Figure 2.4 shows the line glyphs framed in the monthly calendar over the year 2016. This is achieved by the `frame_calendar` function, which computes the coordinates on the calendar for the input data variables. These can then be plotted using the usual **ggplot2** package (Wickham et al., 2018) functions. All of the grammar of graphics (Wilkinson, 2005; Wickham, 2009) can be applied.

The algorithm for constructing a calendar plot uses linear algebra, similar to that used in the glyph map displays for spatio-temporal data (Wickham et al., 2012). To make a year long calendar requires cells for days, embedded in blocks corresponding to months, organized into a grid layout for a year. Each month can be captured with 35 (5×7) cells, where the top left is Monday of week 1, and the bottom right is Sunday of week 5 by default. These cells provide a micro canvas on which to plot the data. The first day of the month could be any of Monday–Sunday, which is determined by the year of the calendar. Months are of different lengths, ranging from 28 to 31 days, and each month could extend over six weeks but the convention in these months is to wrap the last few days up to the top row of the block. The notation for creating these cells is as follows:

- $k = 1, \ldots, 7$ 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.

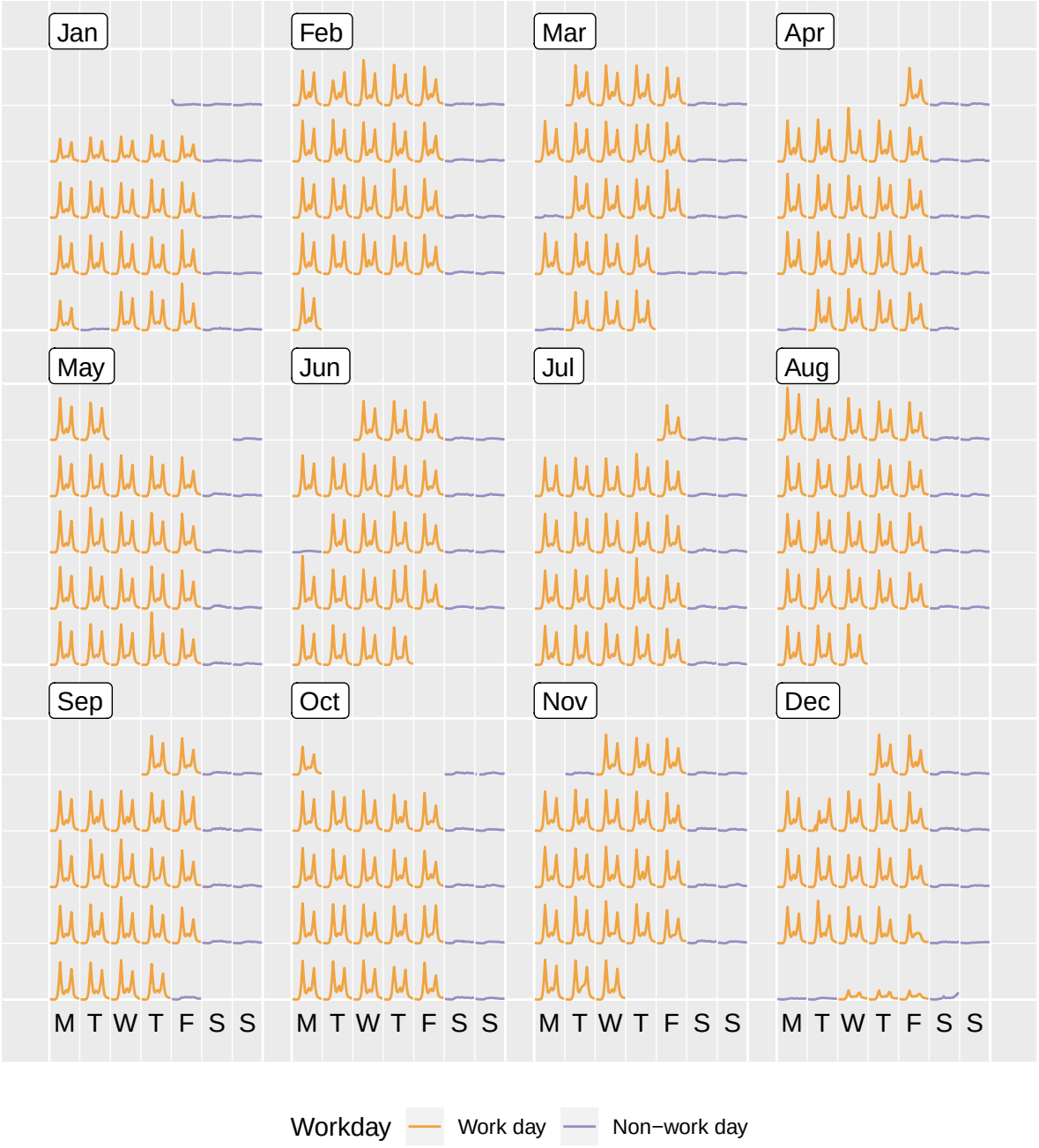

Figure 2.4: *The calendar-based display of hourly foot traffic at Flagstaff Station using line glyphs. The arrangement of the data into a* 3×4 *monthly grid represents all the traffic in 2016. The disparities between week day and weekend along with public holiday are immediately apparent.*

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

Figure 2.5: *Illustration of the indexing layout for cells in a month, where k is day of the week, g is day of the month, (i, j) indicates grid position.*

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

$$\begin{aligned} i &= \lceil (g \bmod 35)/7 \rceil, \\ j &= g \bmod 7. \end{aligned} \tag{2.1}$$

Figure 2.5 illustrates this (i, j) layout for a month where $k = 5$.

To create the layout for a full year, (m, n) denotes the position of the month arranged in the plot, where $1 \leq m \leq M$ and $1 \leq n \leq N$. Between each month requires some small amount of white space, denoted by b. Figure 2.6 illustrates this layout where $M = 3$ and $N = 4$.

Each cell forms a canvas on which to draw the data. Initialize the canvas to have limits $[0, 1]$ both horizontally and vertically. For the pedestrian sensor data, within each cell, hour is plotted horizontally and count is plotted vertically. Each variable is scaled to have values in $[0, 1]$, using the minimum and maximum of all the data values to be displayed, assuming fixed scales. Let h be the scaled hour, and c the scaled count.

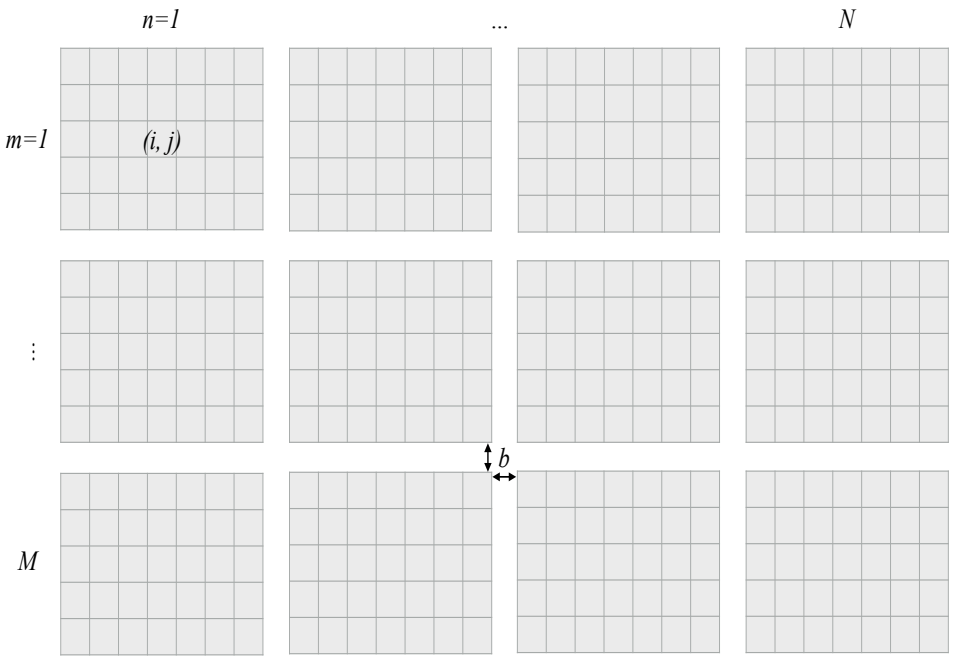

Figure 2.6: *Illustration of the indexing layout for months of one year, where M and N indicate number of rows and columns, b is a space parameter separating cells.*

Then the final points for making the calendar line plots of the pedestrian sensor data is given by:

$$\begin{aligned} x &= j + (n-1) \times 7 + (n-1) \times b + h, \\ y &= i - (m-1) \times 5 - (m-1) \times b + c. \end{aligned} \tag{2.2}$$

Note that for the vertical direction, the top left is the starting point of the grid (in Figure 2.5) which is why subtraction is performed. Within each cell, the starting position is the bottom left.

In order to make calendar-based graphics more accessible and informative, reference lines dividing each cell and block as well as labels indicating week day and month are also computed before plot construction.

Regarding the monthly calendar, the major reference lines separate every month panel and the minor ones separate every cell, represented by the thick and thin lines in Figure 2.4, respectively. The major reference lines are placed surrounding every month block: for each m, the vertical lines are determined by $\min(x)$ and $\max(x)$; for each n, the horizontal

lines are given by $\min(y)$ and $\max(y)$. The minor reference lines are only placed on the left side of every cell: for each i, the vertical division is $\min(x)$; for each j, the horizontal is $\min(y)$.

The month labels located on the top left using $(\min(x), \max(y))$ for every (m, n). The week day texts are uniformly positioned on the bottom of the whole canvas, that is $\min(y)$, with the central position of a cell $x/2$ for each j.

2.2.2 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",
  sunday = FALSE, nrow = NULL, ncol = NULL, polar = FALSE, scale = "fixed",
  width = 0.95, height = 0.95, margin = NULL)
```

It is assumed that the `data` is in tidy format (Wickham, 2014), and `x`, `y` are the variables that will be mapped to the horizontal and vertical axes in each cell. For example, the `x` is the time of the day, and `y` is the count (Figure 2.4). The `date` argument specifies the date variable used to construct the calendar layout.

The algorithm handles displaying a single month or several years. The arguments `nrow` and `ncol` specify the layout of multiple months. For some time frames, some arrangements may be more beneficial than others. For example, to display data for three years, setting `nrow = 3` and `ncol = 12` would show each year on a single row.

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.

Polar transformation

When `polar = TRUE`, a polar transformation is carried out on the data. The computation is similar to the one described in Wickham et al. (2012). Figure 2.7 shows star plots embedded in the monthly calendar layout, which is equivalent to Figure 2.4 placed in polar coordinates. The bimodal work day shape is also visible as boomerangs.

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 2.8 shows the same data as Figure 2.4 scaled locally using `scale = "free"`. The daily trends are magnified.

The `free_wday` scales each week day together. It can be useful to comparing trends across week days, allowing relative patterns for weekends versus week days to be examined. Similarly, the `free_mday` uses free scaling for any day within a given month.

Figure 2.7: *Figure 2.4 in circular layout, which is referred to as star plots. The daily periodicity on work days are clearly visible.*

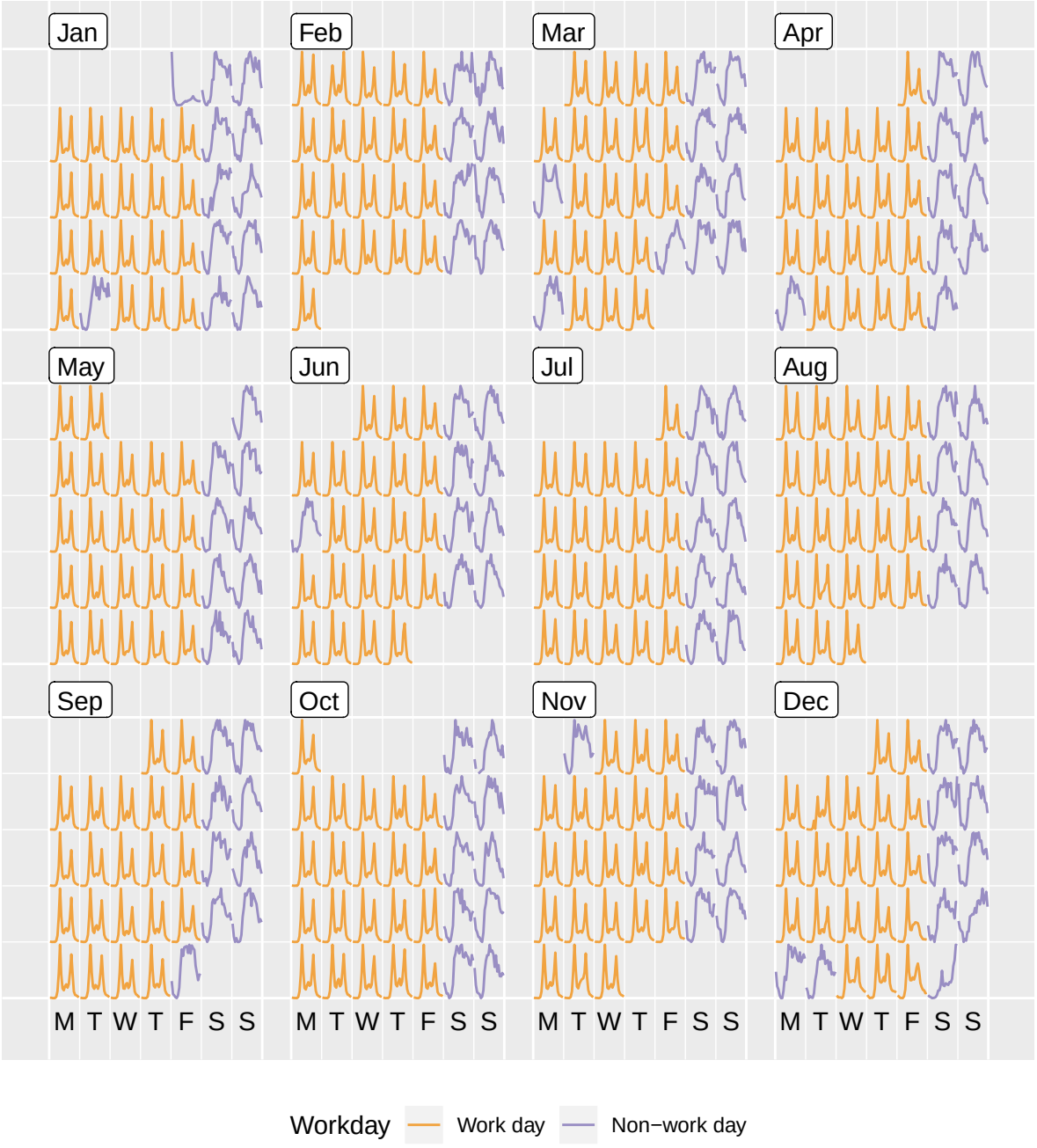

Figure 2.8: *Line glyphs on the calendar format showing hourly foot traffic at Flagstaff Station, scaled over all the days. The individual shape on a single day becomes more distinctive, however it is impossible to compare the size of peaks between days.*

Orientation

By default, grids are laid out horizontally. This can be transposed by setting the `dir` parameter to `"v"`, in which case i and j are swapped in the Equation 2.1. This can be useful for creating calendar layouts for countries where vertical layout is the convention.

Language support

Most countries have adopted this western calendar layout, while the languages used for week day and month would be different across countries. We also offer languages other than English for text labelling. Figure 2.13 shows the same plot as Figure 2.12 labelled using simplified Chinese characters.

2.2.3 Variations

Overlaying and faceting subsets

Plots can be layered. The comparison of sensors can be done by overlaying plot the values for each (Figure 2.9). Differences between the pedestrian patterns at these sensors can be seen. Flagstaff Station exhibits strong commuters patterns, with fewer pedestrian counts during the weekends and public holidays. This suggests that Flagstaff Station has limited functionality on non-work days. From Figure 2.9 it can be seen that Birrarung Marr has a distinct temporal trend 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 2.10 presents the idea of faceting calendar plots. This allows comparing the overall structure between sensors, while emphasizing individual sensor variation. In particular, it can be immediately learned that when Birrarung Marr was busy and packed, for example Australian Open in the last two weeks of January. This is concealed in the conventional graphics.

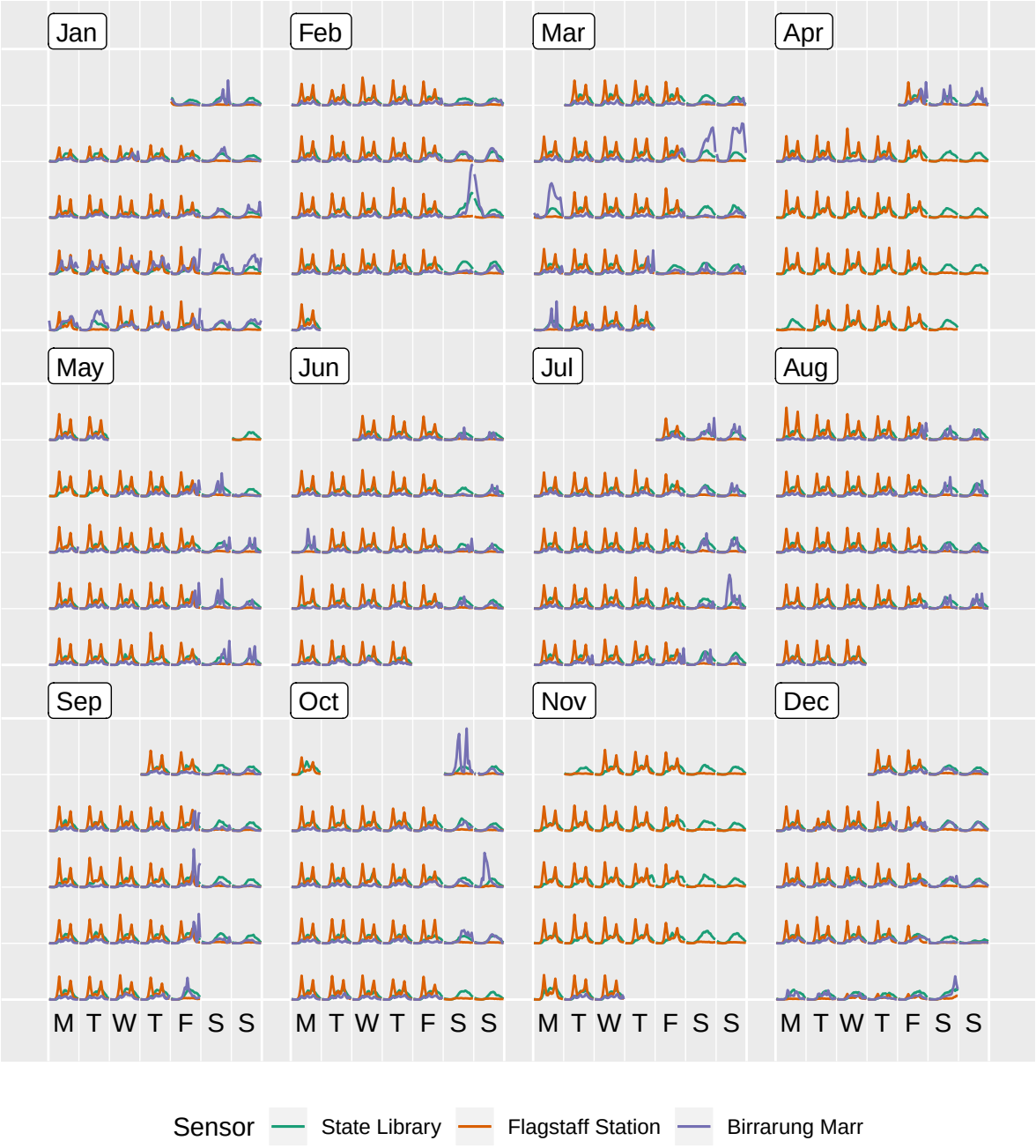

Figure 2.9: *Overlaying line graphs of the three sensors in the monthly calendar. Three sensors demonstrate very different traffic patterns. Birrarung Marr tends to attract many pedestrians for special events held on weekends, contrasting to the bimodal commuting traffic at Flagstaff Station.*

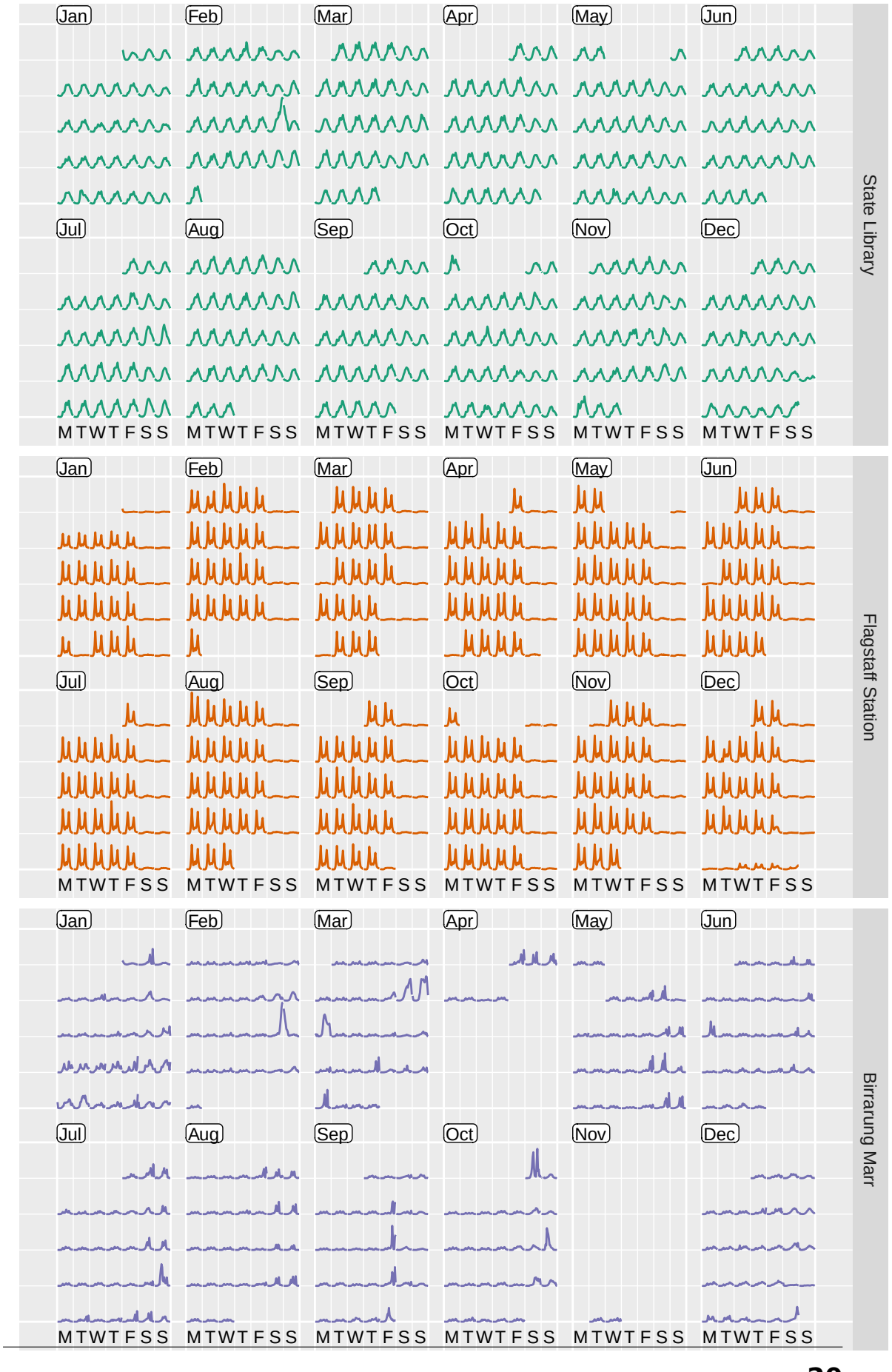

Figure 2.10: *Line charts, embedded in the* 6×2 *monthly calendar, colored and faceted by the 3*

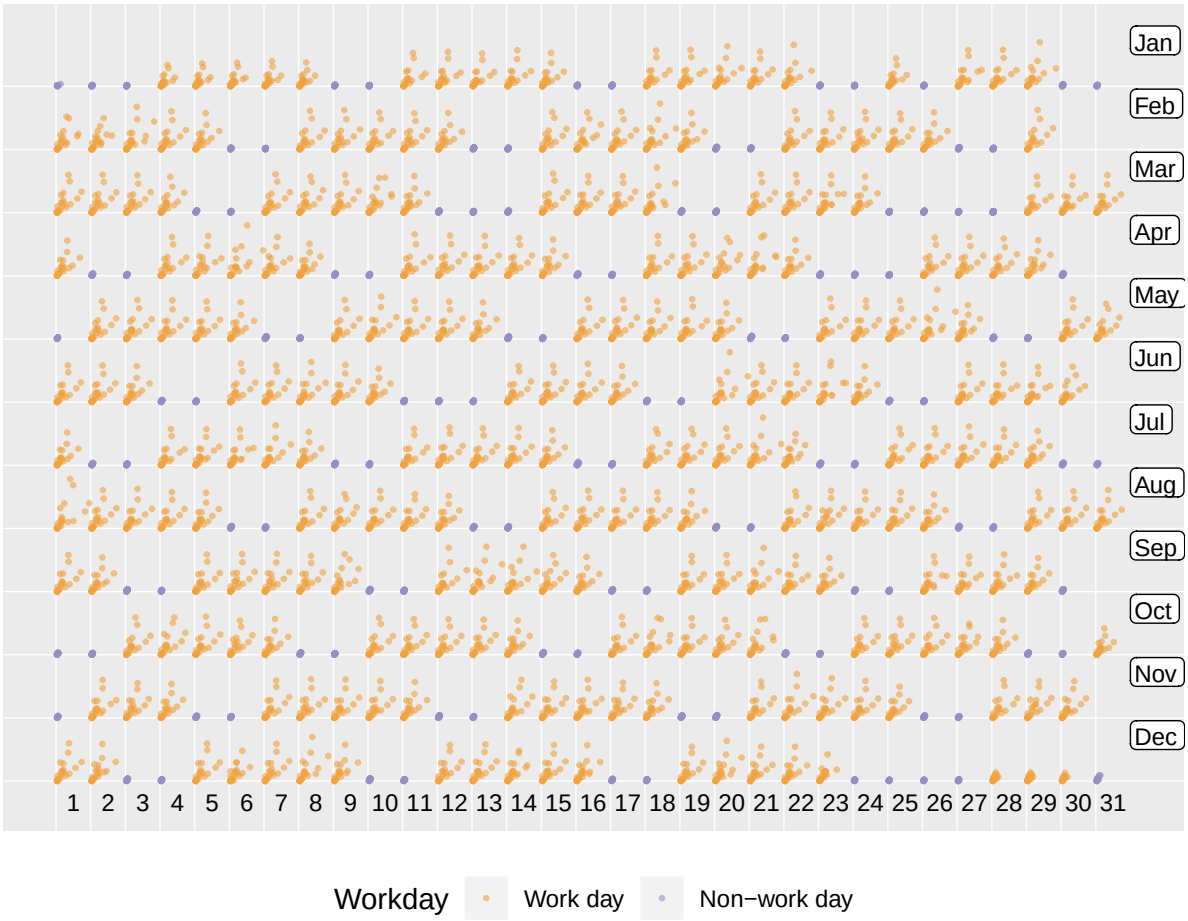

Figure 2.11: *Lag scatterplot in the daily calendar layout. Each hour's count is plotted against previous hour's count at Flagstaff Station to demonstrate the autocorrelation at lag 1. The correlation between them is more consistent on non-work days than work days.*

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 2.11 shows a lag scatterplot at Flagstaff Station, where the lagged hourly count is assigned to the `x` argument and the current hourly count to the `y` argument. This figure is organized in the daily calendar layout. Figure 2.11 indicates two primary patterns, strong autocorrelation on weekends, and weaker autocorrelation on work days. At the higher counts, on week days, the next hour sees possibly substantial increase or decrease in counts, essentially revealing a bimodal distribution of consecutive counts, as supported by Figure 2.4.

The algorithm can also produce more complicated plots, such as boxplots. Figure 2.12 uses a loess smooth line superimposed on side-by-side boxplots. It shows the distribution of hourly counts across all 43 sensors during December. The last week of December is the holiday season: people are off work on the day before Christmas, go shopping on the Boxing day, and stay out for the fireworks on New Year's Eve.

Interactivity

As a data restructuring tool, the interactivity of calendar-based display 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 function `frame_calendar` can be ported to a form of selection button or text input in a graphical user interface like R shiny (Chang et al., 2018). 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, 2018). 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.

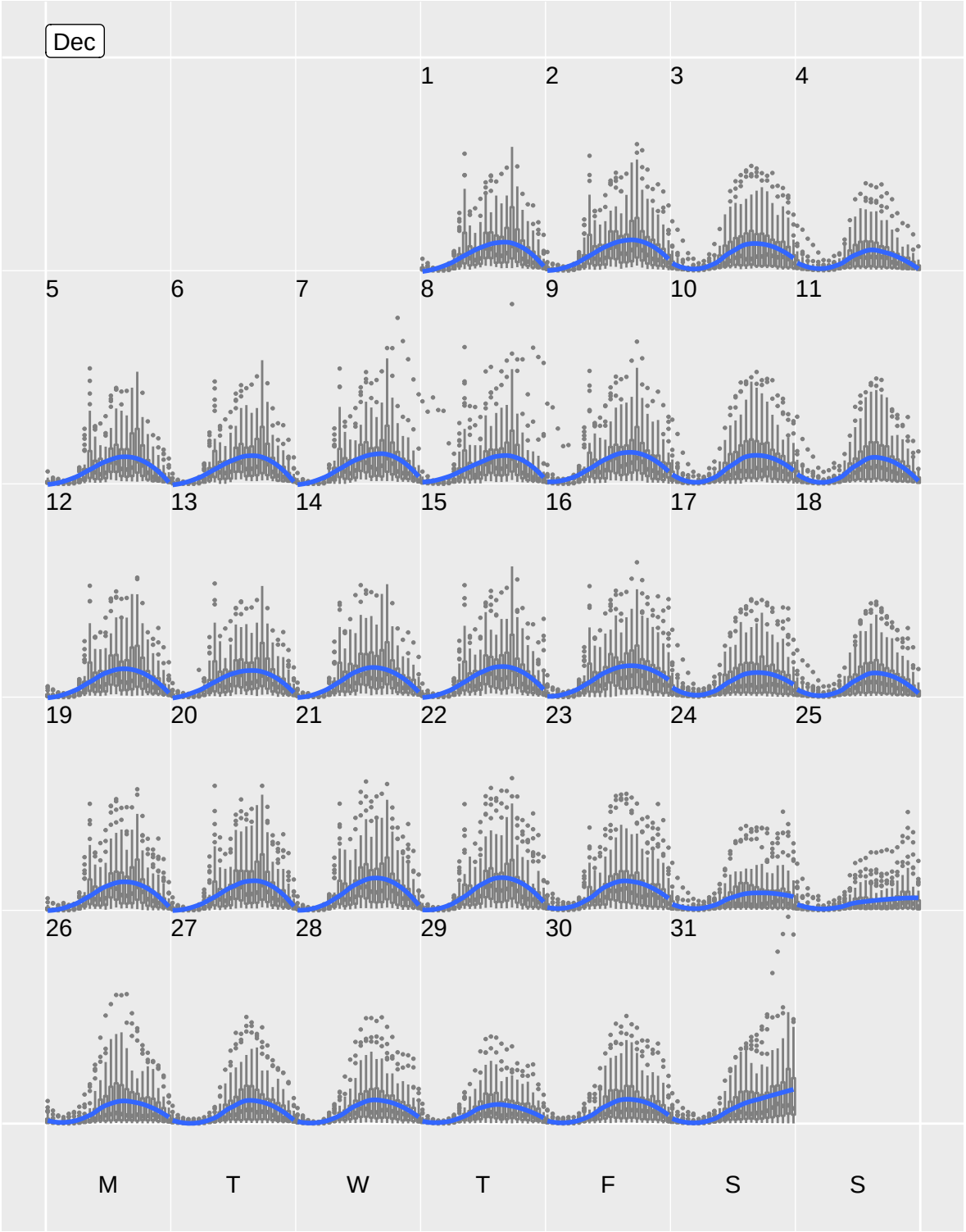

Figure 2.12: *Side-by-side boxplots of hourly counts for all the 43 sensors in December 2016, with the loess smooth line superimposed on each day. It shows the hourly distribution in the city as a whole. There is one sensor attracting a larger number of people on New Year's Eve than the rest.*

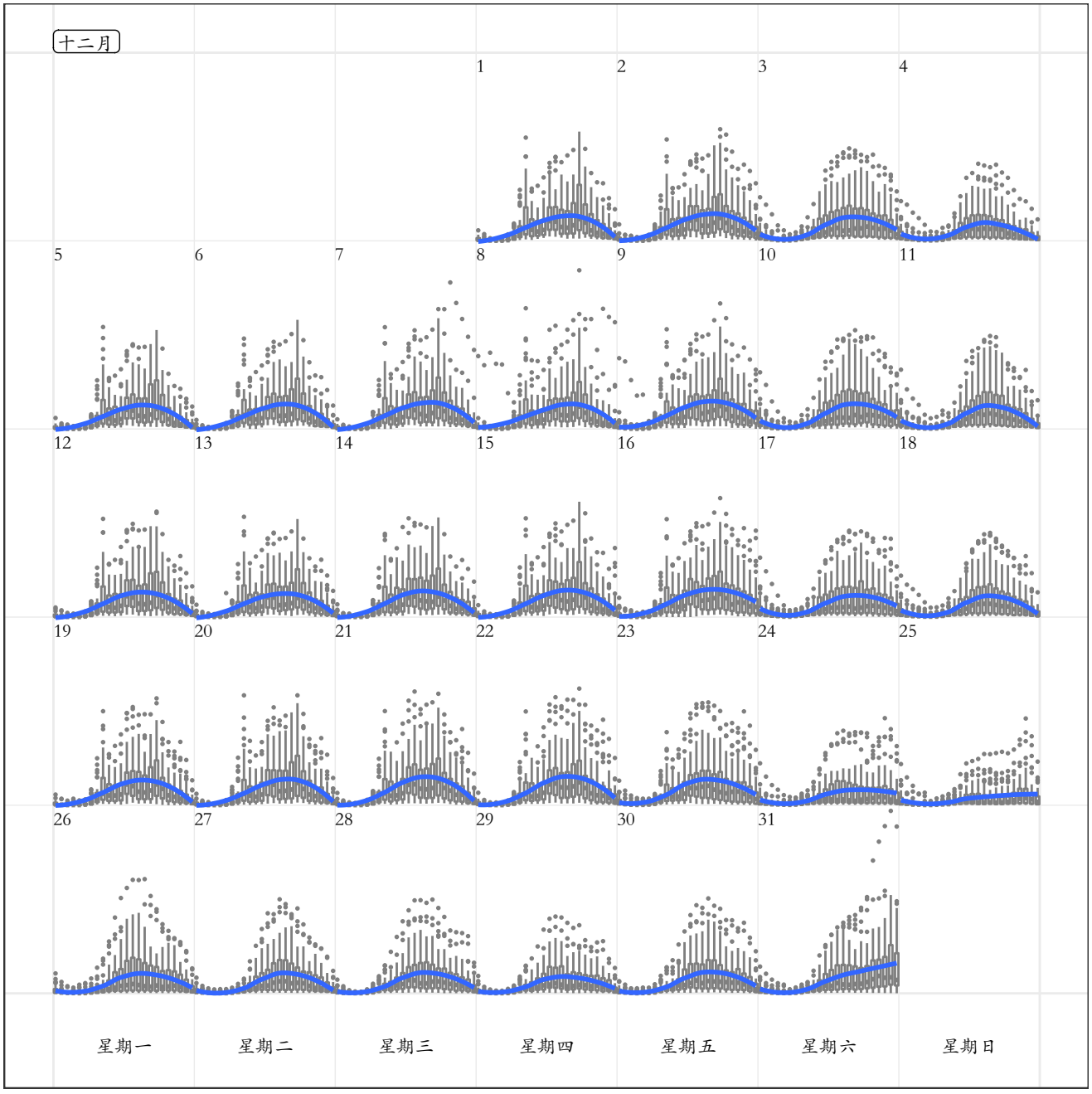

Figure 2.13: *The same plot as Figure 2.12, but with the month and week day labels in Chinese. It demonstrates the natural support for languages other than English.*

2.3 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 company, and these four households are the data of colleagues of the authors. The calendar display is useful to help people understand their energy use. The data contains half-hourly electricity consumption in 2017 and 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 2.14 shows the energy use across days of week in the form of letter value plots (Hofmann, Wickham, and Kafadar, 2017). Letter value plots are a variant of boxplots for large data, with other quantiles represented by boxes. Letters indicate the fraction of the data divisions, for example, F indicates fourths or quartiles, and the two outer ends of the box are the 25th and 75th percentile, the same traditional ends of the box as a boxplot. The letter E indicates eighths, with box ends being 12.5th and 87.5th percentiles of the data. These additional boxes replace the whiskers in a traditional boxplot. The letter value plots for the households, show a line indicating the median (M) and the innermost boxes corresponding to the fourth (F) and the eighth (E) quantile divisions. 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 2.15 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 or heating on when they get home from work and keep it operating till 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 or heating runs continuously for some periods,

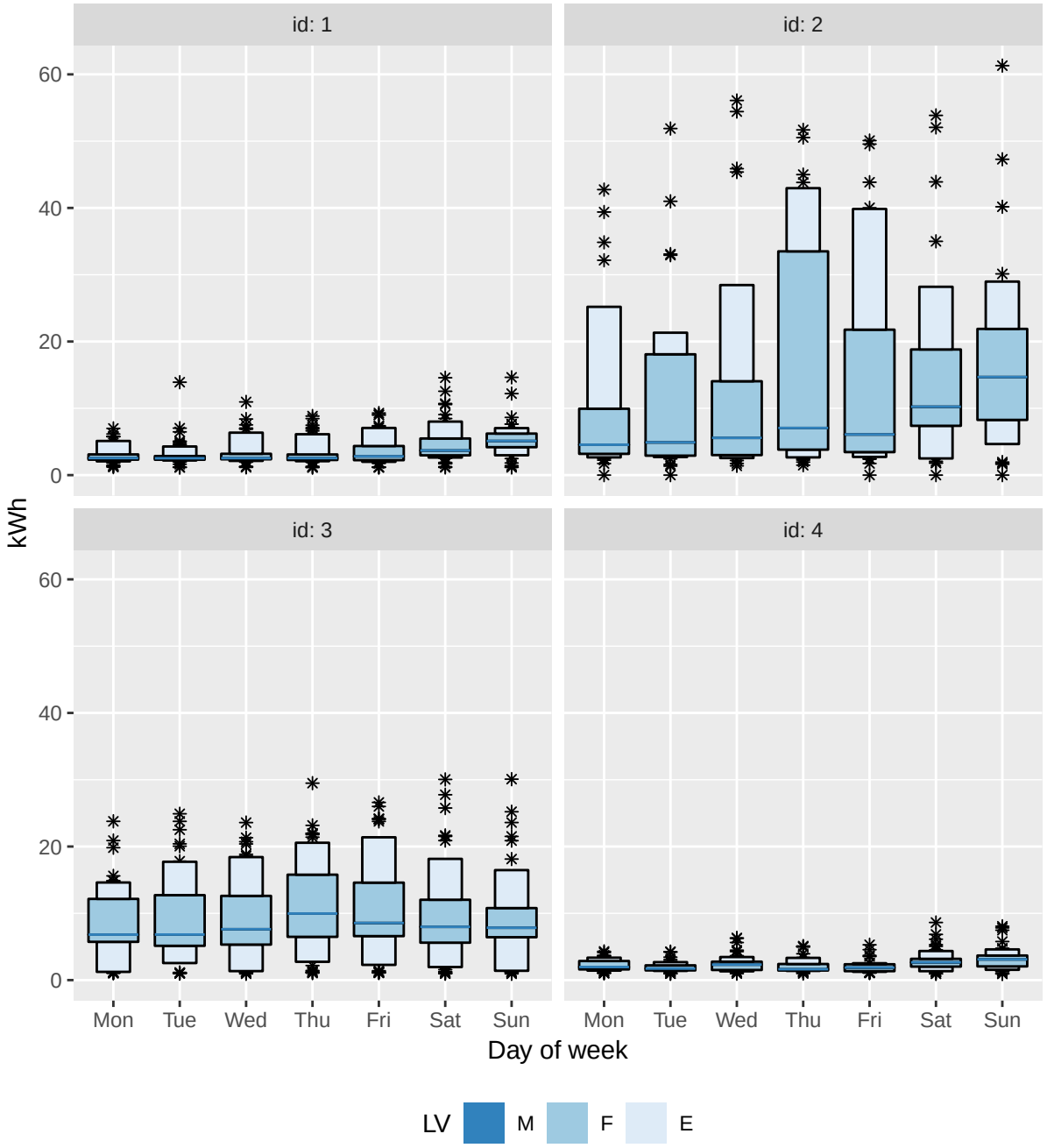

Figure 2.14: *Letter value plots of daily energy usage against day of week for four households, with one line dislaying the median (M) and each box corresponding to the fourth (F) and eighth (E) paired quantile estimates. Suggested by the medians, household 3 uses more enery than the others on the week days, due to a large family size. By constrast, household 2 sees considerally large variability.*

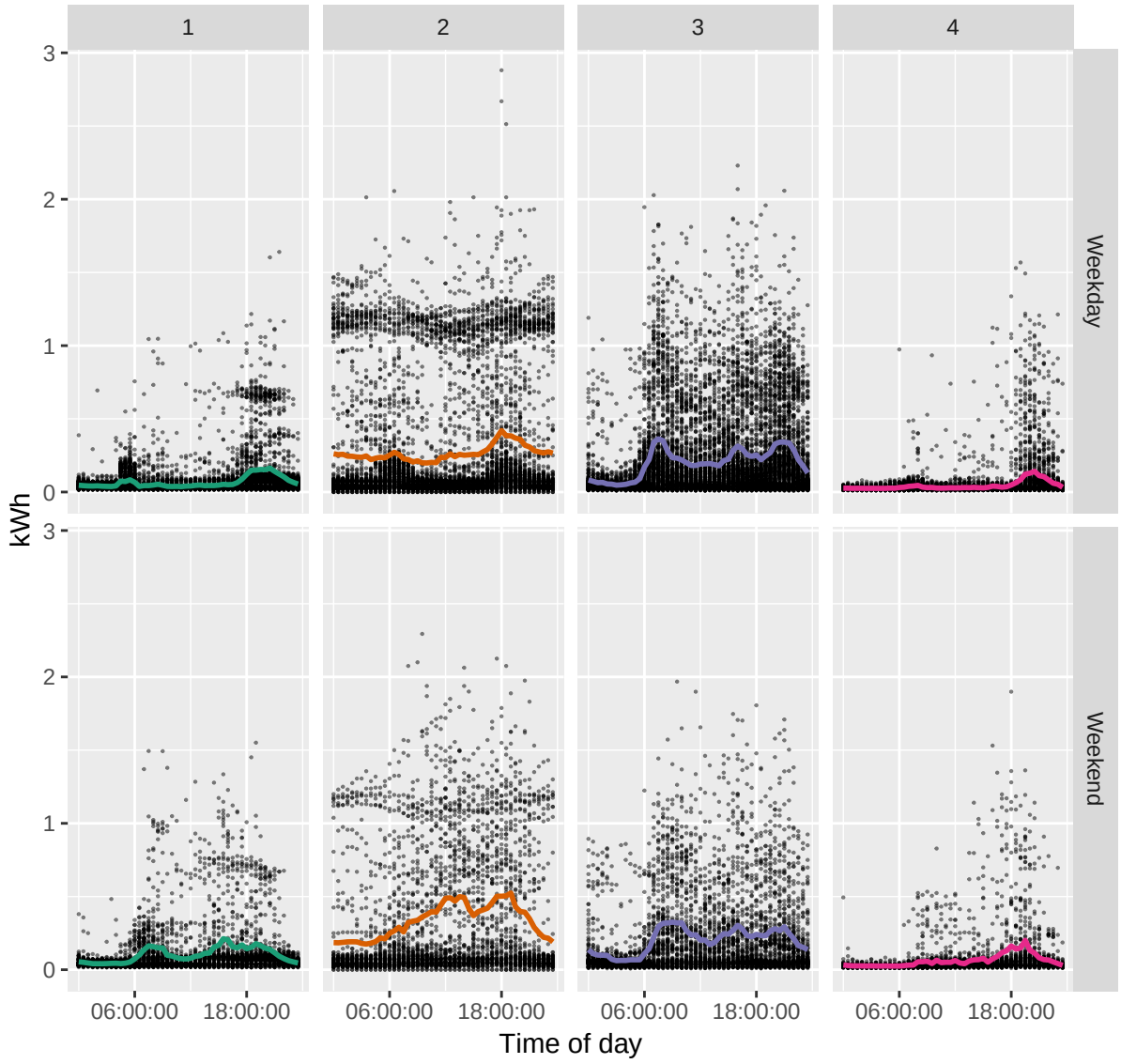

Figure 2.15: *Scatterplot between half-hourly energy usage and time of day with the averages overlaid, constrasting week days and weekends for each household. All have different daily routines between week days and weekends, except for household 3. On week days, household 1 wakes up early before 6, and household 2 around 6, followed by household 3 and 4. The use of air conditioning and heating are noted in household 1 and 2.*

consuming the twice the energy as household 1. A third peak occurs around 3pm 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 2.14 and 2.15, part of a traditional graphical toolkit, are useful for summarizing overall deviations across days and households.

Figure 2.16, 2.17, 2.18 and 2.19 display the data in calendar layout individually for each household, unfolding their day-to-day life. Glancing over household 1, you can see that their overall energy use is low. Their week day energy use is distinguishable from their weekends, indicating a working household. The air conditioner appears to be used in the summer months in the evening and weekends. In household 2, heating keeps functioning for consecutive hours, which is evident in the mid July; while household 1 uses heating cautiously. These observations help to explain the stripes and clusters in Figure 2.15. The calendar plots speak the stories about vacation time that are untold by previous plots. Household 1 is on vacation over three weeks of June, and household 2 was also away for holidays during Christmas and the second week of June. Figure 2.18 shows household 3 takes two one-month-long family trips in September till early October and in June/July, and household 4 is away over two or three weeks in early October, December and late June. The use of air conditioning and heating leaves no trace in these two households.

Figure 2.16: *Calendar display for household 1, indicates higher weekend usage, and in the summer months, November-February. It seems that they took a vacation in June.*

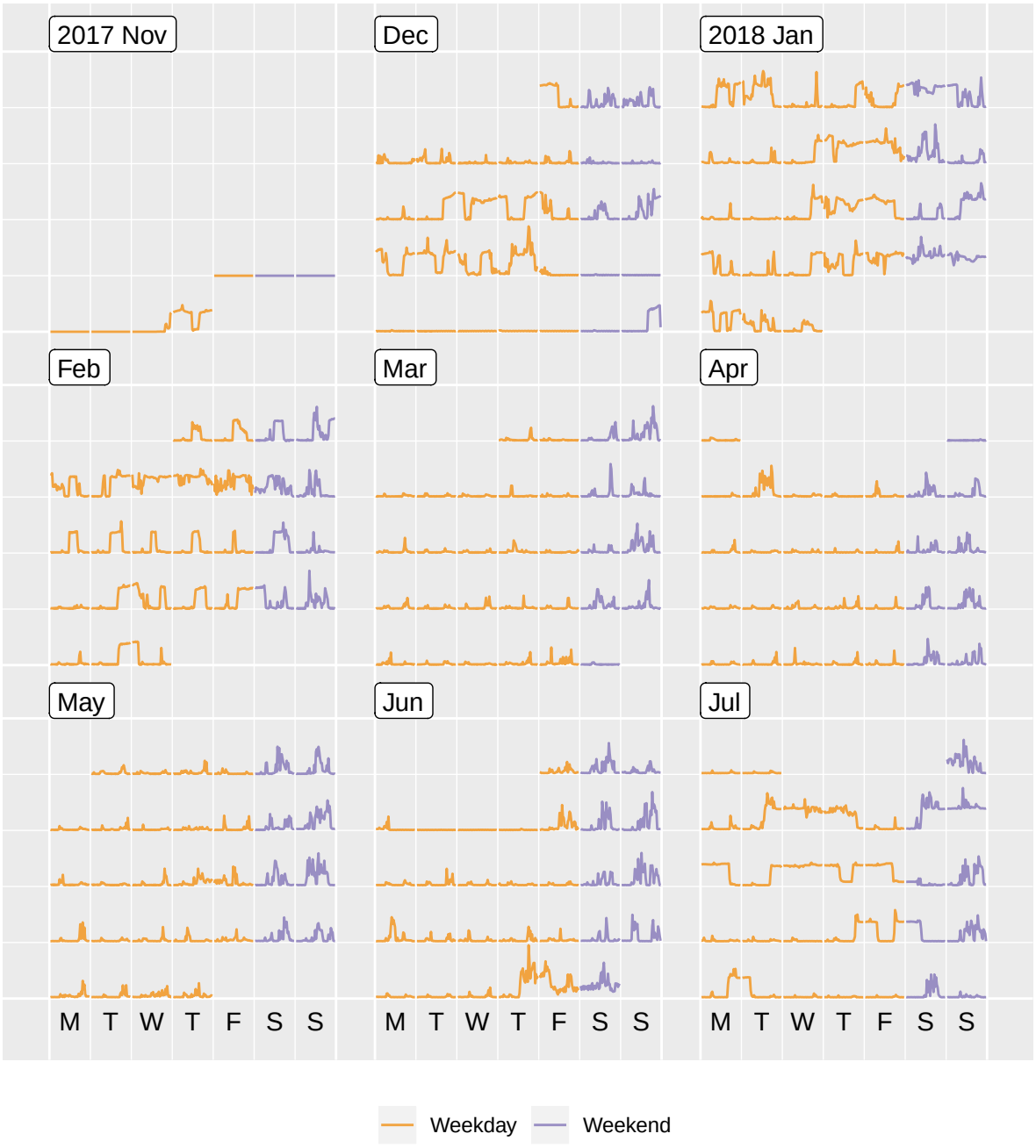

Figure 2.17: *Calendar display for household 2, reveals their tendency to use air conditioning and heating continuously. Not many vacation were taken.*

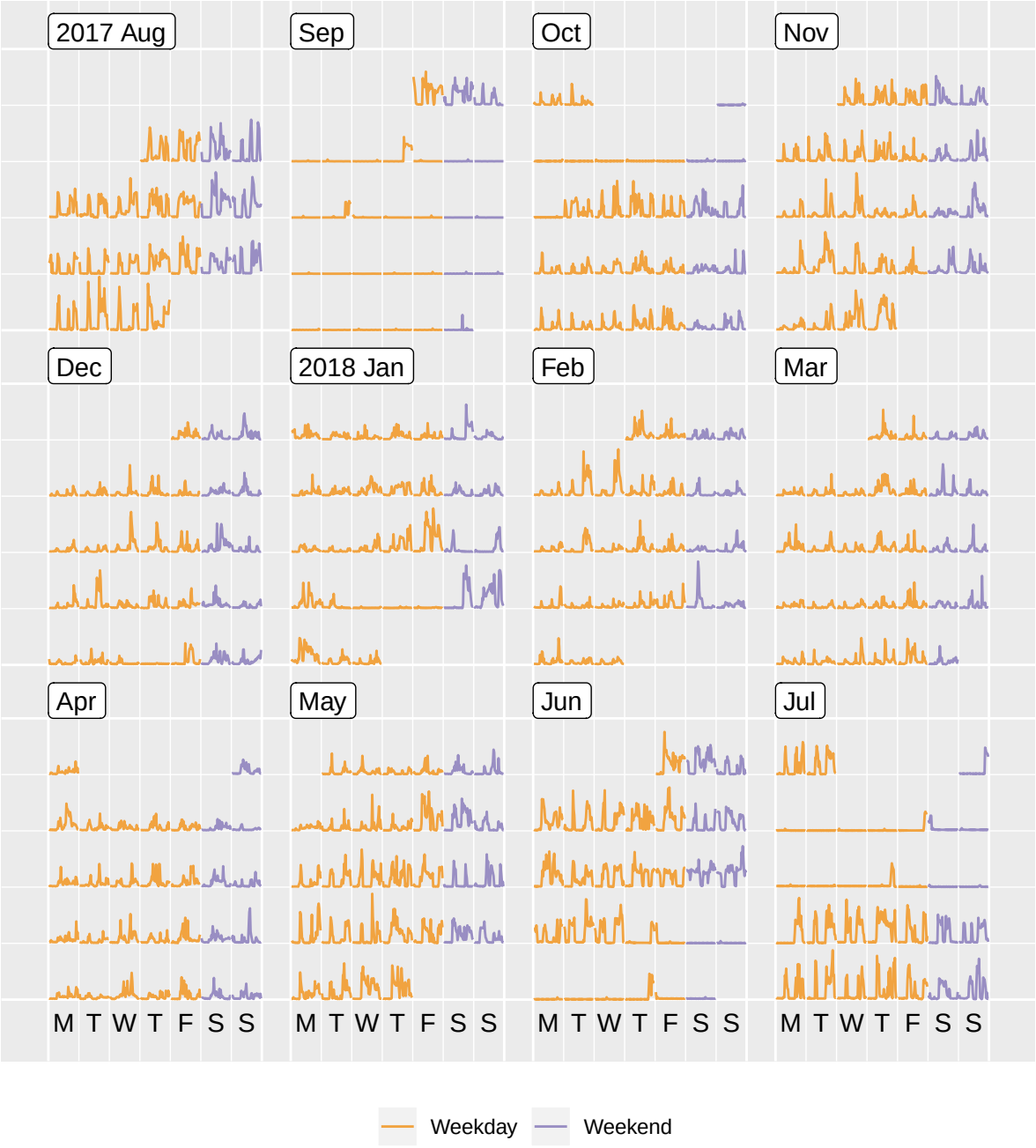

Figure 2.18: *Calendar display for household 3. Their energy use reveals higher energy use in the winter months, with multiple peaks daily on both week days and weekends. There are some high peaks in summer, perhaps indicating occasional air conditioner use. There have been several long vacations in the past year.*

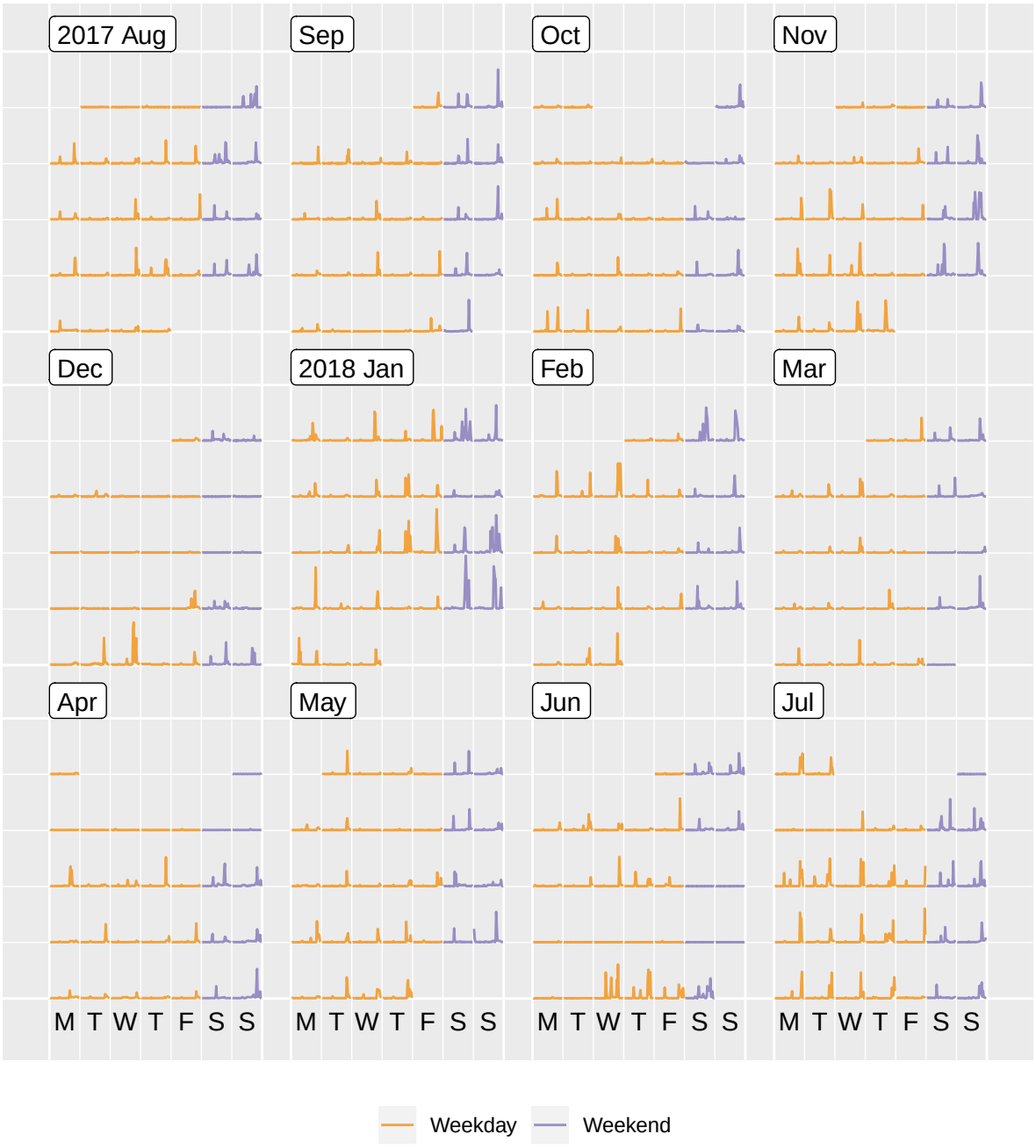

Figure 2.19: *Calendar display for household 4, shows energy use mostly in the evenings and on weekends. Three short trips were taken in October, December, and June.*

2.4 Discussion

The calendar-based visualization provides data plots in the familiar format of an everyday tool. Patterns on special events for the region, like Anzac Day in Australia, or Thanksgiving Day in the USA, are more visible to the viewer as public holidays.

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. Layouts beyond the western calendar could be achieved by the same modular arithmetic approach.

The calendar layout will be useful for studying consumer trends and human behavior. It will not be so 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 layout is a type of faceting and could be useful to develop this into a fully-fledged faceting method, with formal labels and axes. This is a future goal.

Acknowledgements

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

Bibliography

Chang, W, J Cheng, J Allaire, Y Xie, and J McPherson (2018). *shiny: Web Application Framework for R*. R package version 1.1.0. https://CRAN.R-project.org/package=shiny.

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

Cleveland, WS and R McGill (1984). Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association* **79**(387), 531–554.

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

Hofmann, H, H Wickham, and K Kafadar (2017). Letter-Value Plots: Boxplots for Large Data. *Journal of Computational and Graphical Statistics* **26**(3), 469–477.

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

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

Lam, H, T Munzner, and R Kincaid (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*. https://plotly-book.cpsievert.me.

Van Wijk, JJ and ER Van Selow (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 (2018). *wanderer4melb: Shiny App for Wandering Around the Downtown Melbourne 2016*. R package version 0.1.0. https://github.com/earowang/wanderer4melb.

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

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

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. https://CRAN.R-project.org/package=tidyverse.

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

Wickham, H, H Hofmann, C Wickham, and D Cook (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)*. Secaucus, NJ: Springer-Verlag New York, Inc.

Xie, Y (2016). *bookdown: Authoring Books and Technical Documents with R Markdown*. ISBN 978-1138700109. Boca Raton, Florida: Chapman and Hall/CRC. https://github.com/rstudio/bookdown.