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

Di Cook
Monash Univeristy

Rob J Hyndman
Monash Univeristy

Abstract

The abstract of the article.

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

1. Introduction

Calendar-based visualisation turns out to be a useful tool in unfolding human-related activities in temporal context. For example, Van Wijk and Van Selow 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 effectively 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, their techniques are too constrained to colour-encoding graphics. We shall extend this calendar-based arrangement using linear algebra tools from a single type of heatmap to a broader range of glyphs, from univariate to multivariate cases, from one calendar format to three formats. The proposed algorithm has been developed in the `sugrrants` package (Wang, Cook, and Hyndman 2017) using R (R Core Team 2017).

This work was originally motivated by studying hourly foot traffic in the city of Melbourne (City of Melbourne 2017). There have been 43 sensors installed across downtown Melbourne since 2009 in order to capture the pulse of city life. (FIGURE ?: weekday at multiple sensors) Figure (?) lends itself to addressing some of the issues:

1. to simultaneously exhibit multiple seasons including time of day, day of week, and day of year

2. to visualise long historical time series data in an information-dense way at the overview level
3. to compare and contrast multiple time series on a common scale
4. to quickly look up the date when there are unexpected events happening without interactivity.

2. The function and its construction

In the section, the algorithms to construct calendar grids, different scales, and reference lines and labels are discussed in detail.

```
frame_calendar(
  data, x, y, date, calendar = "monthly", dir = "h", sunday = FALSE,
  nrow = NULL, ncol = NULL, polar = FALSE, scale = "fixed"
)
```

2.1. Grid construction

The algorithm for constructing a calendar plot uses linear algebra, similar to that used in 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 $\times$ 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 1 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 2 illustrates this layout.

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

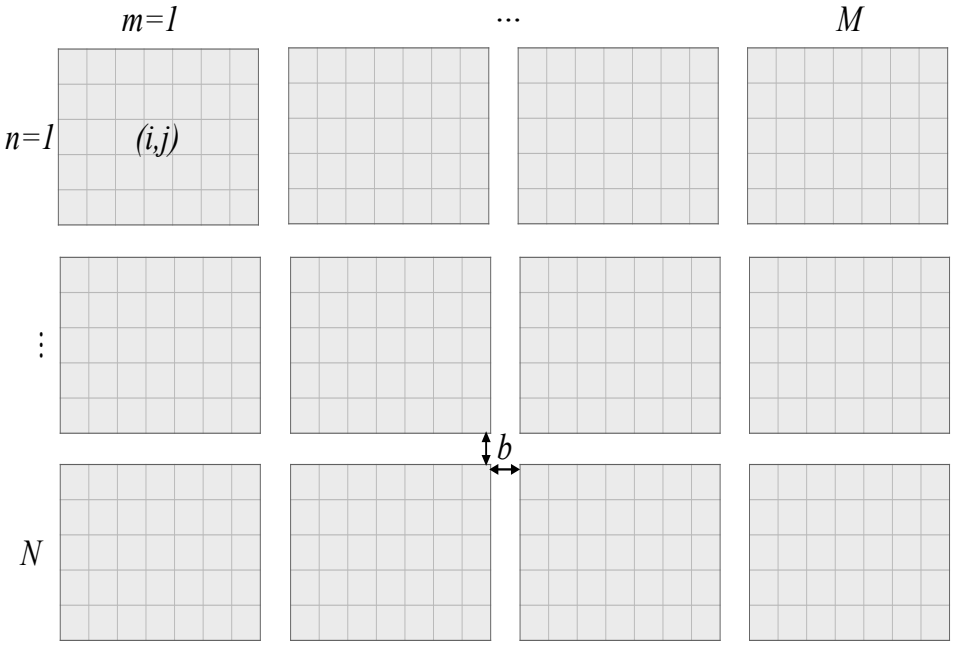

Figure 2: Illustration of the indexing layout for months of one year.

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. 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 extended relatively easily to layout from one single month to multiple years depending on the time span to be visualised. The month-by-month arrangement can be determined by the user's choice through M and N. If one would like to compare and contrast January, February and so on across multiple years, $M = 12$ and N depending on the number of years could be setup.

Monthly calendar format provides an effective layout to emphasise differences between weekdays and weekends as well as day of year (for example, public holiday) simultaneously, however, either one of which is sufficient in certain visualisation applications. If there is apparent depiction of weekday effects but none of yearly effects, weekly calendar format (weeks of a year) could be considered employing; otherwise days of a month could be laid out for standing-out yearly behaviours.

(MAKE TWO DIAGRAMS ON WEEKLY AND DAILY CALENDAR FORMATS IN KEYNOTE). Weekly and daily calendar formats are the simplifications of monthly calendar.

We only illustrate that calendar grids are laid out on the horizontal direction. The vertical direction can be enabled by swapping i and j in the algorithm stated above. It is useful for users in some countries where they get used to calendars of vertical organisation and other possible comparisons like Mondays across the columns rather than rows.

2.2. Scales

One of the advantages using glyphs over heatmaps is evident in easily enabling scales on different temporal scales. All of the line glyphs shown above are drawn on the global scale so that it makes the magnitudes comparable over all the period. On the other hand, some subseries of small magnitudes yet worth-noting shapes possibly become invisible whilst embedded in the overall picture. The individual scale or something else, as a result, complements the calendar-based visualisation in general.

There are other three types of scales in addition to the global scale: individual scales regardless of days, conditional on weekdays, and conditional on days of a month.

2.3. Reference lines and labels

It can be noted that reference lines separating each cell and panel and labels indicating

weekday and month are also constructed in order to make calendar-based graphics more accessible and informative.

Regarding the monthly calendar format, the major reference lines separate every month panel and the minor ones separate every day cell, represented by the think and thin lines respectively. The major reference lines are placed on the far left and right as well as the bottom and top of every month panel: for each m, the vertical lines are determined by $\min(x)$ and $\max(x)$; for each n, the horizontal lines are computed by $\min(y)$ and $\max(y)$. The minor reference lines are placed on the initial positions: for each i, the vertical line is $\min(x)$; for each j, the horizontal line is $\min(y)$.

The abbreviated month labels located on the top left are obtained through $(\min(x), \max(y))$ for every m and n. The weekday texts with a single letter are uniformly placed on the bottom of the whole canvas, that is $\min(y)$.

3. Examples

4. 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.

The limitation is also evident: hard to perceive trend as not on the common scale.

We shall enable interactivity in the calendar-based graphics for time series data. It will allow users to transform different temporal components and switch displays between overlaying and faceting through key strokes or mouse clicks.

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.

Jacobs J (2017). *ggcal: Calendar Plot Using ggplot2*. R package version 0.1.0.

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

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 (????). "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, Hofmann H, Wickham C, Cook D (2012). "Glyph-maps for Visually Exploring Temporal Patterns in Climate Data and Models." *Environmetrics*, **23**(5), 382–393.

Affiliation:

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

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

Rob J Hyndman
Monash Univeristy
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