Hands-on Exercise 5: Visualising and Analysing Time-oriented Data with R

Dr. Kam Tin Seong
Assoc. Professor of Information Systems

School of Computing and Information Systems,
Singapore Management University

2020-2-15 (updated: 2022-05-14)

Learning Outcome

In this hands-on exercise, you will gain hands-on experience on:

- plotting a calender heatmap by using ggplot2 functions,
- plotting a cycle plot by using ggplot2 function,
- plotting a horizon chart

Getting Started

Write a code chunk to check, install and launch the following R packages:

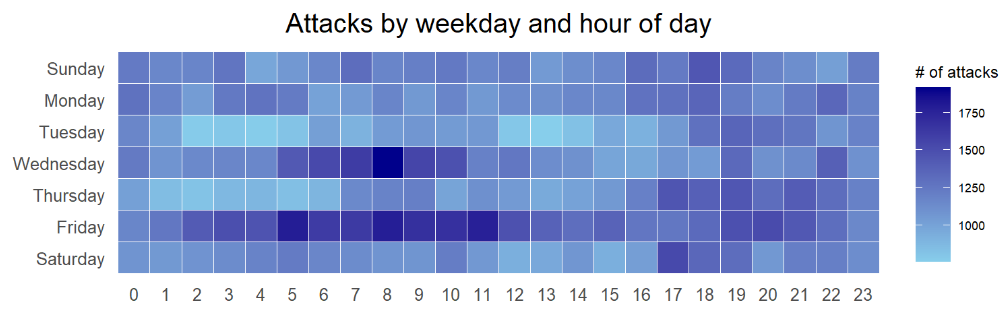
- 'scales',
- 'viridis',
- 'lubridate',
- 'ggthemes',
- 'gridExtra',
- 'tidyverse',
- 'readxl',
- 'knitr',
- data.table

The solution:

```
packages = c('scales', 'viridis',
             'lubridate', 'ggthemes',
             'gridExtra', 'tidyverse',
             'readxl', 'knitr',
             'data.table')

for (p in packages){
  if(!require(p, character.only = T)){
    install.packages(p)
  }
  library(p,character.only = T)
}
```

Calendar Heatmap

In this section, you will learn how to plot a calender heatmap programmetically with R.



By the end of this section, you will be able to:

- plot a calender heatmap by using ggplot2 functions and extension,
- to write function using R programming,
- to derive specific date and time related field by using base R and lubridate packages
- to perform data preparation task by using tidyr and dplyr packages.

The Data

For the purpose of this hands-on exercise, *eventlog.csv* file will be used. This data file consists of 199,999 rows of time-series cyber attack records by country.

Importing the data

First, you will use the code chunk below to import ***eventlog.csv*** file into R environment and called the data frame as ***attacks***.

```
attacks <- read_csv("data/eventlog.csv")
```

Examining the data structure

It is always a good practice to examine the imported data frame before further analysis is performed.

For example, *kable()* can be used to review the structure of the imported data frame.

```
kable(head(attacks))
```

There are three columns, namely *timestamp*, *source_country* and *tz*.

- *timestamp* field stores date-time values in POSIXct format.
- *source_country* field stores the source of the attack. It is in *ISO 3166-1 alpha-2* country code.
- *tz* field stores time zone of the source IP address.

timestamp	source_country	tz
2015-03-12 15:59:16	CN	Asia/Shanghai
2015-03-12 16:00:48	FR	Europe/Paris
2015-03-12 16:02:26	CN	Asia/Shanghai
2015-03-12 16:02:38	US	America/Chicago
2015-03-12 16:03:22	CN	Asia/Shanghai
2015-03-12 16:03:45	CN	Asia/Shanghai

Data Preparation

Step 1: Deriving *weekday* and *hour of day* fields

Before we can plot the calender heatmap, two new fields namely *wkday* and *hour* need to be derived. In this step, we will write a function to perform the task.

```r
make_hr_wkday <- function(ts, sc, tz) {
  real_times <- ymd_hms(ts,
                       tz = tz[1],
                       quiet = TRUE)
  dt <- data.table(source_country = sc,
                   wkday = weekdays(real_times),
                   hour = hour(real_times))
  return(dt)
  }
```

Note: `ymd_hms()` and `hour()` are from **lubridate** package and `weekdays()` is a **base** R function.

Data Preparation

Step 2: Deriving the attacks tibble data frame

```
wkday_levels <- c('Saturday', 'Friday',
                  'Thursday', 'Wednesday',
                  'Tuesday', 'Monday',
                  'Sunday')

attacks <- attacks %>%
  group_by(tz) %>%
  do(make_hr_wkday(.$timestamp,
                   .$source_country,
                   .$tz)) %>%
  ungroup() %>%
  mutate(wkday = factor(
    wkday, levels = wkday_levels),
    hour  = factor(
      hour, levels = 0:23))
```

Note: Beside extracting the necessary data into *attacks* data frame, `mutate()` of **dplyr** package is used to convert *wkday* and *hour* fields into **factor** so they'll be ordered when plotting

Table below shows the tidy tibble table after processing.

tz	source_country	wkday	hour
Africa/Cairo	BG	Saturday	20
Africa/Cairo	TW	Sunday	6
Africa/Cairo	TW	Sunday	8
Africa/Cairo	CN	Sunday	11
Africa/Cairo	US	Sunday	15
Africa/Cairo	CA	Monday	11

Building the Calendar Heatmaps

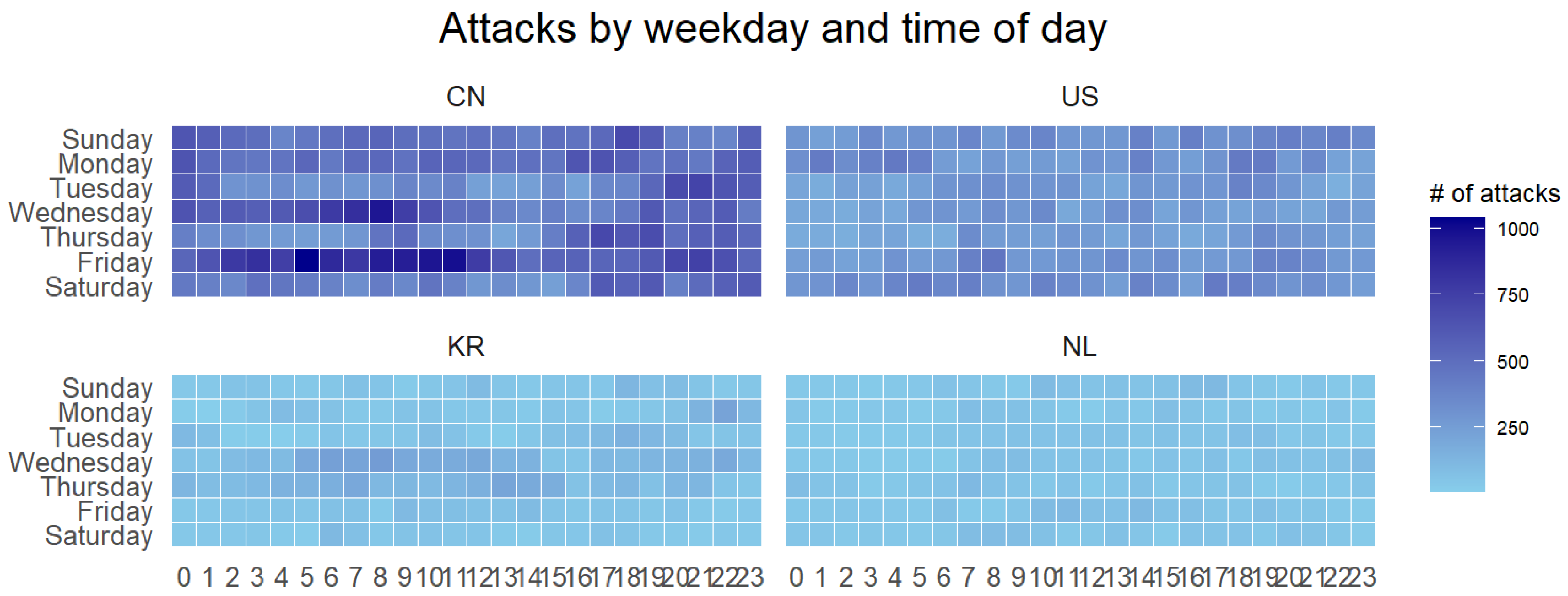
```
grouped <- attacks %>%
  count(wkday, hour) %>%
  ungroup() %>%
  na.omit()

ggplot(grouped,
       aes(hour,
           wkday,
           fill = n)) +
geom_tile(color = "white",
          size = 0.1) +
theme_tufte(base_family = "Helvetica") +
coord_equal() +
scale_fill_gradient(name = "# of attacks",
                    low = "sky blue",
                    high = "dark blue") +
labs(x = NULL,
     y = NULL,
     title = "Attacks by weekday and time of da
theme(axis.ticks = element_blank(),
      plot.title = element_text(hjust = 0.5),
      legend.title = element_text(size = 8),
      legend.text = element_text(size = 6) )
```

Things to learn from the code chunk:

- a tibble data table called *grouped* is derived by aggregating the attack by *wkday* and *hour* fields.
- a new field called *n* is derived by using `group_by()` and `count()` functions.
- `na.omit()` is used to exclude missing value.
- `geom_tile()` is used to plot tiles (grids) at each x and y position. `color` and `size` arguments are used to specify the border color and line size of the tiles.
- `theme_tufte()` of **ggthemes** package is used to remove unnecessary chart junk. To learn which visual components of default ggplot2 have been excluded, you are encouraged to comment out this line to examine the default plot.
- `coord_equal()` is used to ensure the plot will have an aspect ratio of 1:1.
- `scale_fill_gradient()` function is used to creates a two colour gradient (low-high).

Building Multiple Calendar Heatmaps

Challenge: Building multiple heatmaps for the top four countries with the highest number of attacks.



Plotting Multiple Calendar Heatmaps

Step 1: Deriving attack by country object

In order to identify the top 4 countries with the highest number of attacks, you are required to do the followings:

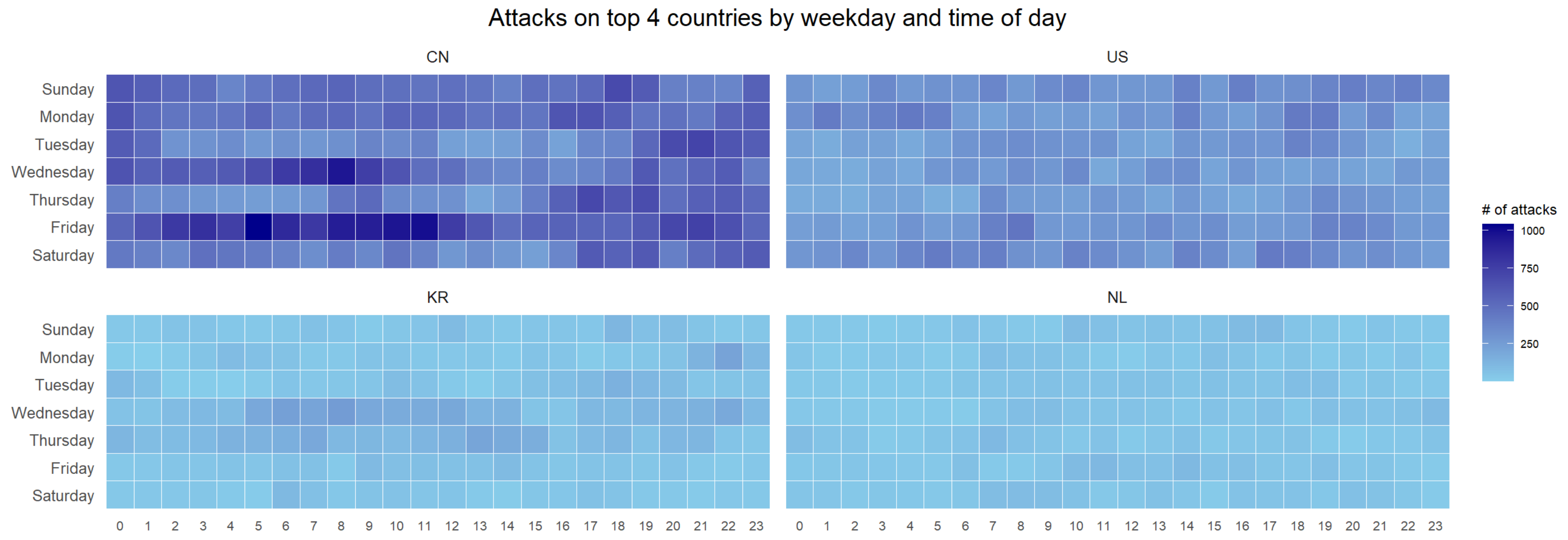
- count the number of attacks by country,
- calculate the percent of attackes by country, and
- save the results in a tibble data frame.

Step 2: Preparing the tidy data frame

In this step, you are required to extract the attack records of the top 4 countries from *attacks* data frame and save the data in a new tibble data frame (i.e. *top4_attacks*).

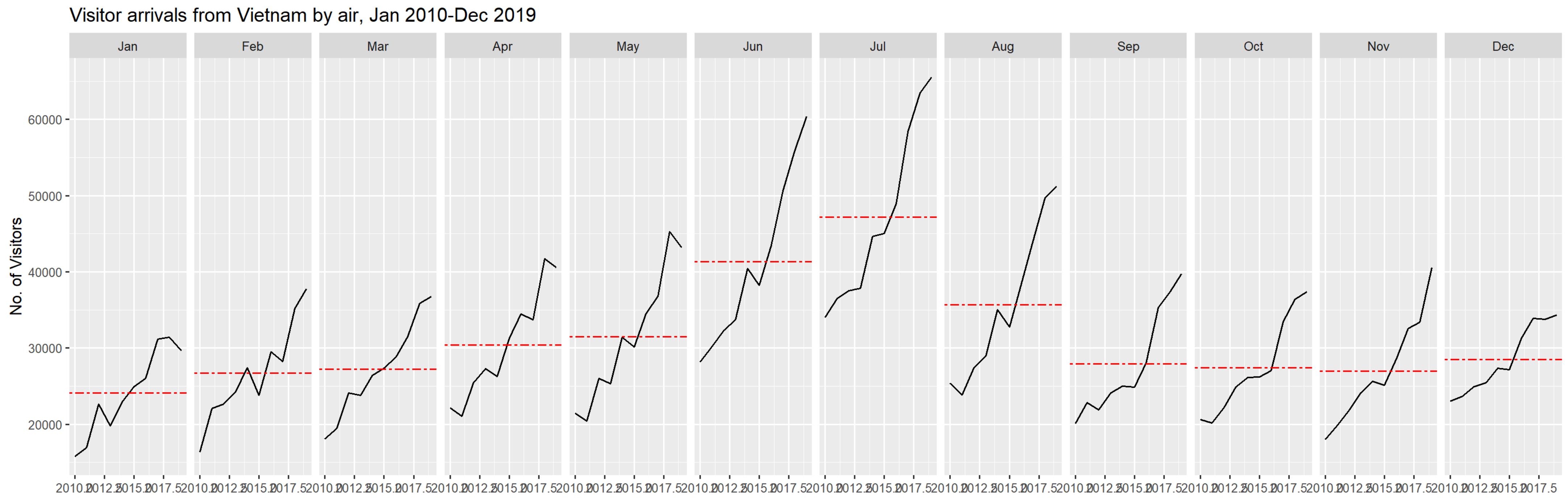
Plotting Multiple Calendar Heatmaps

Step 3: Plotting the Multiple Calender Heatmap by using ggplot2 package.



Cycle Plot

In this section, you will learn how to plot a cycle plot showing the time-series patterns and trend of visitor arrivals from Vietnam programmatically by using ggplot2 functions.



Data Preparation

Step 1: Data Import

For the purpose of this hands-on exercise, *arrivals_by_air.xlsx* will be used.

The code chunk below imports *arrivals_by_air.xlsx* by using `read_excel()` of **readxl** package and save it as a tibble data frame called *air*.

```
air <- read_excel("data/arrivals_by_air.xlsx")
```

Step 2: Deriving month and year fields

Next, two new fields called *month* and *year* are derived from *Month-Year* field.

```
air$month <- factor(month(air$`Month-Year`),
                    levels=1:12,
                    labels=month.abb,
                    ordered=TRUE)
air$year <- year(ymd(air$`Month-Year`))
```

Data Preparation

Step 4: Extracting the target country

Next, the code chunk below is use to extract data for the target country (i.e. Vietnam)

```
Vietnam <- air %>%
  select(`Vietnam`,
         month,
         year) %>%
  filter(year >= 2010)
```

Step 5: Computing year average arrivals by month

The code chunk below uses `group-by()` and `summarise()` of **dplyr** to compute year average arrivals by month.

```
hline.data <- Vietnam %>%
  group_by(month) %>%
  summarise(avgvalue = mean(`Vietnam`))
```

Plotting the cycle plot

The code chunk below is used to plot the cycle plot.

```
ggplot() +
  geom_line(data=Vietnam,
            aes(x=year,
                y=`Vietnam`,
                group=month),
            colour="black") +
  geom_hline(aes(yintercept=avgvalue),
             data=hline.data,
             linetype=6,
             colour="red",
             size=0.5) +
  facet_grid(~month) +
  labs(axis.text.x = element_blank(),
       title = "Visitor arrivals from Vietnam
  xlab("") +
  ylab("No. of Visitors")
```

Visulising Daily Life