30535 Applied Problem Set 4

Lee Kyung Ko & Mia Jiang

05/25/2022

Front matter
This submission is my work alone and complies with the 30535 integrity policy.

Add your initials to indicate your agreement: **LKK,MJ**

Late coins used this pset: 2. Late coins left: 3.

Submission Notes: Total page of the file is less than 25 pages and we've submitted on github

1 Prelim questions

1.1

Answer YES. We deleted the *city*, *confidence*, *nThumbsUp*, *country* clolumns since they are not useful for data analysis. All the data are from Chicago and in the US so those two columns could not provide additional information for us when analzying. For the *nThumbsUp* and *confidence* columns, we do not need to use them in the problem set analysis so we can delete them to reduce the memory it takes for R to run codes.
Additionally, we can also drop the *geo* and *geoWKT* columns since we create new columns for *latitude* and *longitude* separeately. We can also delete the original time column *ts* since we converted it to *Central Time*.

1.2.

Answer The subtype of *JAM*' alert includes 4 meaningful values. There are *JAM_LIGHT_TRAFFIC*, *JAM_STAND_STILL_TRAFFIC*,*JAM_MODERATE_TRAFFIC*, and *JAM_HEAVY_TRAFFIC*. In the later part of the problem set, we recode each category into numeric value from 1 to 4. Getting hints from assigning numeric values to each subtype, we propose introducing more objective and quantifiable values to the level of jam. Depending on how much traffic is considered usual, jam level could imply different level of traffics. Therefore, we would suggest setting some numeric ranges for jams and group levels based on the average speed cars can move. For example, we can assign ***extremely_heavy_traffic*** value to those alerts with **average speed of 0~10km/h**. The variable would be meaningfully used when drawing the overall traffic flow map and present the expected speed of vehicle operation compared to the previous string categories.

1.3

Answer *self-selection* means when we are using a research project which the data is reported by participants who has the right to choose whether to join or not, then we may get bias since the participant group may differ from those who opt out.

From the data provided by Waze, we can only observe consumers who use Waze service and agree to provide their data to the company. The Waze consumer group itself could be a subgroup with specific characteristics

that are meaningfully different from the whole population. As mentioned in the question, most consumer data would face this self-selection because they only carry data of their consumer group. For example, those who choose to use Waze may have higher level of literacy, then the report of traffic accidents may be lower than the actual frequency since people with lower literacy may fail to understand traffic rules and thus encounter more accidents. Therefore, analysis on the consumer group is necessary in addition to decomposing the consumer group.

Reference

2 Obtaining data from cloud using SQL

Used the data posted on Canvas

3 Data exploration in JSON

3.1

Answer:

- JSON stands for ***JavaScript Object Notation*** and it consists of name & value pair wrapped by curly brackets with commas for separating the pairs. It's comparatively less compact than **CSV**, widely used for data configuration and API thanks to its scalability and convertibility. It is better for presenting hierarchical and relational data than less versatile-csv. It's also known as a *light-weight* data format type, *human-readability* and *nesting features*, while usually has larger file size than CSV and less secured.
- CSV stands for ***Comma separated value***. Compared with JSON, CSV stores smaller size of file usually in tabular format and plain text, which is separated by commas. It does not use a data type and is difficult to integrate or scale. Also, when displaying hierarchical data, errors will occur. However, **CSV** file is more secured and require less memory.
- In this file specifically, it is made possible by JSON's *nesting feature* and we can see the nesting by repeated variable name with different values for each object in JSON file

Reference https://www.geeksforgeeks.org/difference-between-json-and-csv/
https://coresignal.com/blog/json-vs-csv/

3.2

```
# Convert JSON file into a list
library(jsonlite)
json_list <- fromJSON("69c3e9b3-182c-4ec9-a5f3-c0e176568a3d.json")

# Convert to a tibble
json_list %>% as_tibble()
```

```
## Error:
## ! Tibble columns must have compatible sizes.
## * Size 199: Column `jams`.
## * Size 221: Column `alerts`.
## i Only values of size one are recycled.
```

Answer:
The error message told us that *Tibble columns must have compatible sizes*, which means all columns in a tibble must have the same length. However, in the JSON list we just created, the *jams* column and *alerts* column have different length. The *jams* column has 228 rows while the *alerts* column has 264 rows. Therefore, we cannot convert it to a tibble.

3.3

```
length(json_list)
```

```
## [1] 6
```

```
names(json_list)
```

```
## [1] "alerts"          "endTimeMillis"   "startTimeMillis" "startTime"
## [5] "endTime"         "jams"
```

```
summary(json_list)
```

```
##                 Length Class      Mode
## alerts          15     data.frame list
## endTimeMillis    1     -none-     numeric
## startTimeMillis  1     -none-     numeric
## startTime        1     -none-     character
## endTime          1     -none-     character
## jams            19     data.frame list
```

Answer:
The length of the list is 7 and the names of each item are *alerts*, *endTimeMillis*, *irregularities*, *startTimeMillis*, *startTime*, *endTime*, *jams*

- Based on the vector type, we can store all list items which is not a list as an atomic vector, i.e., *endTimeMillis*, *startTimeMillis*, *startTime*, *endTime*. That is because *list* is **recursive vectors** instead of **atomic vectors**
- Data type: *endTimeMillis*, *startTimeMillis* are **numeric**, *startTime* and *endTime* are **character**
- List could include items with different lengths, therefore it can not be stored as tibbles, which require equal length for each item. Therefore, all the other items(atomic vectors), i.e., *endTimeMillis*, *startTimeMillis*, *startTime*, *endTime* can be tibbles themselves.

Note One of us read the json file and found 7 list items but the other person somehow cannot read the *irregularities* list after multiple try(it occured once and then disappered again). We found there is a list item called *irregularities* in the original file. Therefore, we followed the *json* file with 7 list items and included *irregularities* although it's not shown here

4 Data cleaning

4.1

```
# Used the data posted on Canvas
all_alerts <- readRDS("all_alerts_all_cols.rds")
# Test the number of rows
test_that("We have the right number of rows", expect_equal(nrow(all_alerts), 982147))
```

```
## Test passed
```

4.2

```
# Dates
all_alerts <- all_alerts %>%
  mutate(ts_date = as.Date(ts))
summary(all_alerts$ts_date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2021-03-02" "2021-06-21" "2021-09-19" "2021-09-28" "2022-01-01" "2022-05-05"
```

```
# City
table(all_alerts$city)
```

```
##
## Chicago, IL
##      982147
```

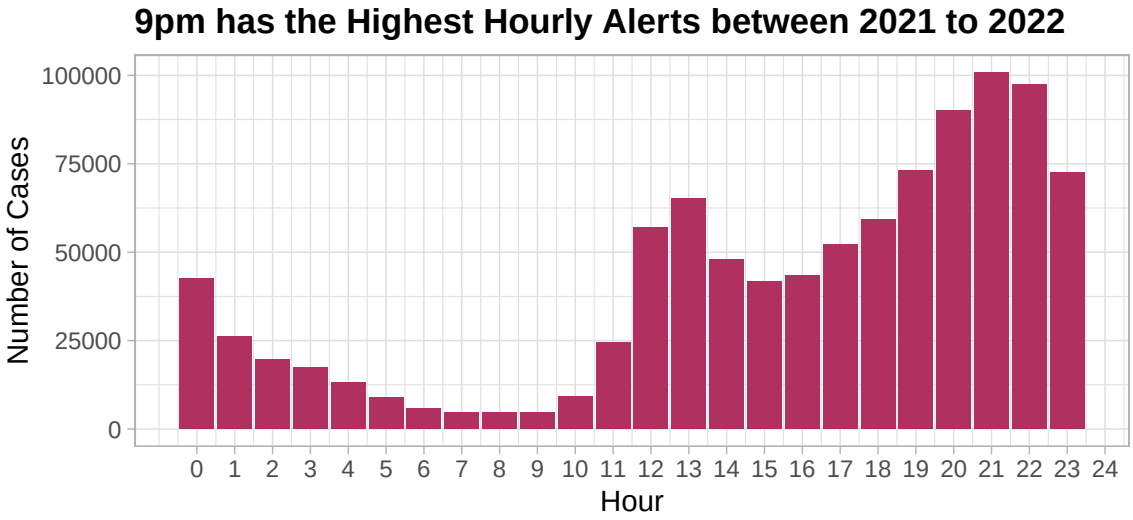
Answer:
It covers data from *2021-03-02* to *2022-05-05*, and all data are from Chicago

4.3

```
all_alerts <- all_alerts %>%
  select(-city, -confidence, -nThumbsUp, -country) %>%
  mutate(geo_numeric = str_sub(geo, 7, -2)) %>%
  separate(geo_numeric, into = c("long", "lat"), sep = "\\s") %>%
  mutate(
    lat = as.numeric(lat),
    long = as.numeric(long)
  )
```

4.4

```
all_alerts %>%
  mutate(hour = hour(ts)) %>%
  count(hour) %>%
  ggplot(aes(x = hour, y = n)) +
  geom_col(
    stat = "identity",
    fill = "maroon"
  ) +
  scale_x_continuous(breaks = seq(0, 24, 1)) +
  labs(
    title = "9pm has the Highest Hourly Alerts between 2021 to 2022",
    x = "Hour",
    y = "Number of Cases",
    caption = "Source: WAZE"
  ) +
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0, face = "bold"),
    plot.caption = element_text(hjust = 0., face = "italic")
  )
```



Answer:
As shown in the plot above, we observe the most alerts at 9pm at night within a single day from *2021-03-02* to *2022-05-05*.

4.5

```
# Default timezone
tz(all_alerts$ts)
```

```
## [1] "UTC"
```

```r
# Convert to Central Time and rounded to nearest 5 mins
all_alerts <- all_alerts %>%
  mutate(
    ts_central = with_tz(ts, tzone = "America/Chicago"),
    ts_central_round = round_date(ts_central, "5 mins"),
    ts_date_central = as.Date(ts_central, tz = "America/Chicago")
  )
```

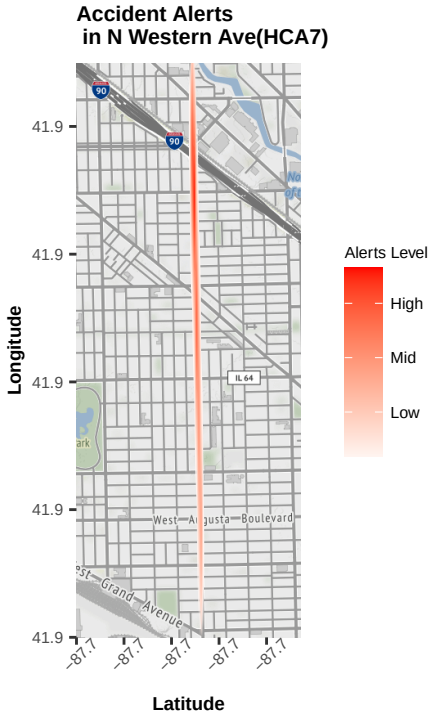
Answer:
The default timezone is *UTC* and we successfully converted it to **Central Time** as well as a new column with time rounded to the nearest 5 minutes

5 Waze vision zero

5.1

```r
library(ggmap)
library(RColorBrewer)
?get_stamenmap
# Define surrounding area
surrounding <- c(
  left = -87.7,
  bottom = 41.889994,
  right = -87.676392,
  top = 41.934958
)

surrounding_stamen <- get_stamenmap(
  bbox = surrounding,
  zoom = 14
)
# Filter accident data along N. Western Ave
N_Western <- all_alerts %>%
  filter(street == "N Western Ave" & lat <= 41.932657 & lat >= 41.895509 &
    type == "ACCIDENT")
# Plot
ggmap(surrounding_stamen) +
  stat_density_2d(
    data = N_Western,
    aes(long, lat,
      fill = stat(level)
    ),
    bins = 20,
    geom = "polygon"
  ) +
  labs(
    title = "Accident Alerts\n in N Western Ave(HCA7)",
    x = "Latitude",
    y = "Longitude",
    caption = "Source: Wbez data, Vision Zero Project"
  ) +
```

```
  scale_fill_gradient2(
    name = "Alerts Level",
    low = "grey",
    high = "red",
    breaks = c(20000, 40000, 60000),
    labels = c("Low", "Mid", "High")
  ) +
  theme(
    plot.title = element_text(size = 8, face = "bold"),
    axis.text.x = element_text(size = 6, angle = 45),
    axis.text.y = element_text(size = 6),
    axis.title = element_text(size = 7, face = "bold"),
    legend.title = element_text(size = 6),
    legend.text = element_text(size = 6),
    legend.key.size = unit(0.5, "cm"),
    plot.caption = element_text(size = 6, face = "italic", hjust = -1)
  )
```



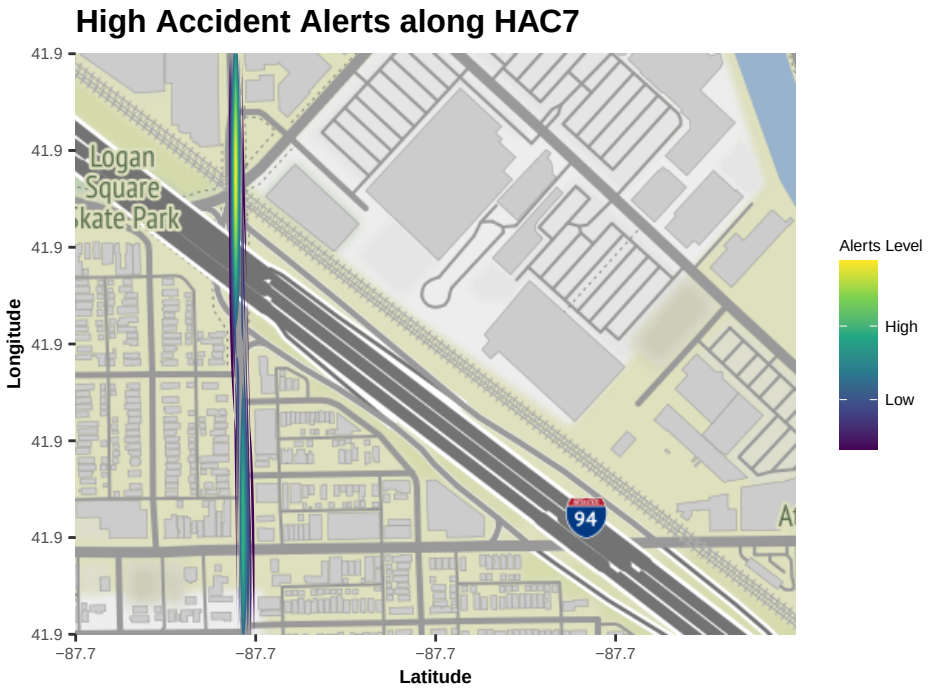
Source: Wbez data, Vision Zero Project

Reference
https://cfss.uchicago.edu/notes/raster-maps-with-ggmap/

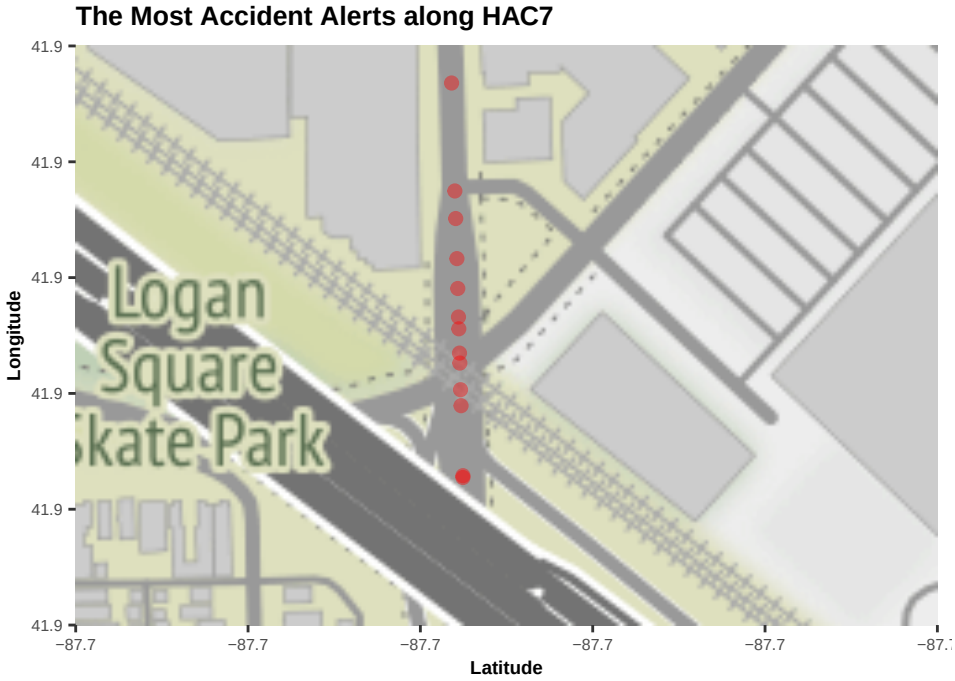
5.2

```
# Focus on the area which showed the highest alert level in Q5.1 plot
map_area_focus <- c(left = -87.69, bottom = 41.924, right = -87.68, top = 41.930)

map_stamen_focus <- get_stamenmap(
```

```
  bbox = map_area_focus,
  zoom = 16
)

ggmap(map_stamen_focus) +
  stat_density_2d(
    data = N_Western,
    aes(long, lat,
      fill = stat(level)
    ),
    bins = 10,
    geom = "polygon"
  ) +
  labs(
    title = "High Accident Alerts along HAC7",
    x = "Latitude",
    y = "Longitude",
    caption = "Source: Wbez data, Vision Zero Project"
  ) +
  scale_fill_continuous(
    name = "Alerts Level",
    type = "viridis",
    breaks = c(500000, 1000000),
    labels = c("Low", "High")
  ) +
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    axis.text = element_text(size = 6),
    axis.title = element_text(size = 7, face = "bold"),
    legend.title = element_text(size = 6),
    legend.text = element_text(size = 6),
    legend.key.size = unit(0.5, "cm"),
    plot.caption = element_text(size = 6, face = "italic", hjust = 0)
  )
```




```
# West Logan Blvd
map_area_most <- c(left = -87.69, bottom = 41.9275, right = -87.685, top = 41.930)

map_stamen_most <- get_stamenmap(
  bbox = map_area_most,
  zoom = 16
)

ggmap(map_stamen_most) +
  geom_point(
    data = N_Western,
    aes(long, lat),
    color = "red",
    size = 2,
    alpha = 0.4
  ) +
  labs(
    title = "The Most Accident Alerts along HAC7",
    x = "Latitude",
    y = "Longitude",
    caption = "Source: Wbez data, Vision Zero Project"
  ) +
  theme(
    plot.title = element_text(size = 10, face = "bold"),
    axis.text = element_text(size = 6),
    axis.title = element_text(size = 7, face = "bold"),
    plot.caption = element_text(size = 6, face = "italic", hjust = 0)
  )
```



Answer:
As we can see in Q5.1, those areas with darker red color are areas with higher level of alerts, and we can narrow down the bounding box to latitude between(41.924, 41.930), with the detailed plot, we can further observe that the location with the most common alerts is the **intersection of N Western Ave & W Logan Blvd**.
Accidents are very likely to take place around this intersection as there exist two intersections right adjacent to each other. There is an *expressway that goes over these intersections* so it could be hard for drivers to secure their view. Further, *the two roads do not meet at a right angle but instead as a curve*. The curvy intersection makes drivers hard to know whether cars or pedestrians are coming from the opposite side. There are also *crosswalks on all sides* that drivers need to slow down and check for traffic lights more carefully. Right next to the exit to an intersection, there is an *entrance to a street-side building* which is highly like to trigger accidents between cars going into the building and other cars that just want to pass on the road. Taken all together, it is understandable that many accidents take place in this intersection.

Reference http://bboxfinder.com

6 Waze Single Event

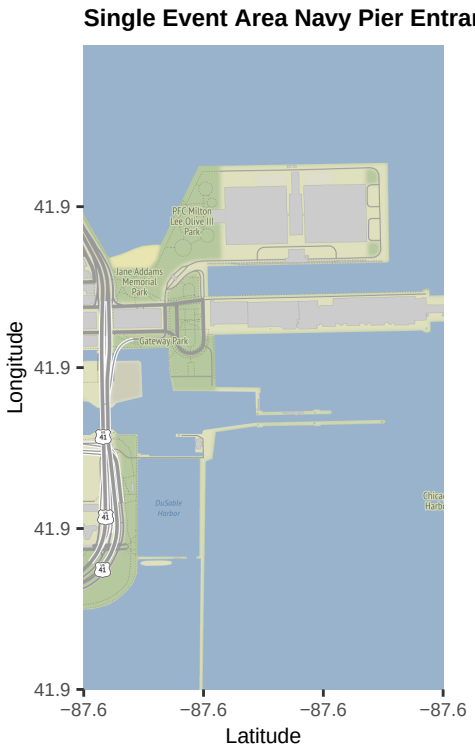
6.1.a

```
#location: entrance to Navy Pier
#long: -87.6139
#lat: 41.89105
all_alerts %>%
  filter(uuid == "a42bc14b-e080-4621-9221-29dd86e553ce") %>%
  select(long, lat)
```

```
## # A tibble: 1 x 2
```

```
##    long   lat
##   <dbl> <dbl>
## 1 -87.6  41.9
```

```
# Define the Bounding Box
long_lowerleft <- -87.615
lat_lowerleft <- 41.88
long_upperright <- -87.60
lat_upperright <- 41.90

event_bb <- c(
  long_lowerleft,
  lat_lowerleft,
  long_upperright,
  lat_upperright
)

event_stamen <- get_stamenmap(
  bbox = event_bb,
  zoom = 16
)
# Plot
ggmap(event_stamen) +
  labs(
    title = "Single Event Area Navy Pier Entrance",
    x = "Latitude",
    y = "Longitude"
  ) +
  theme(
    plot.title = element_text(size = 9, face = "bold"),
    axis.text = element_text(size = 7),
    axis.title = element_text(size = 8)
  )
```




```
# Data frame around the event
event_df <- all_alerts %>%
  filter(between(long, long_lowerleft, long_upperright) &
    between(lat, lat_lowerleft, lat_upperright) &
    type %in% c("ACCIDENT", "JAM"))
```

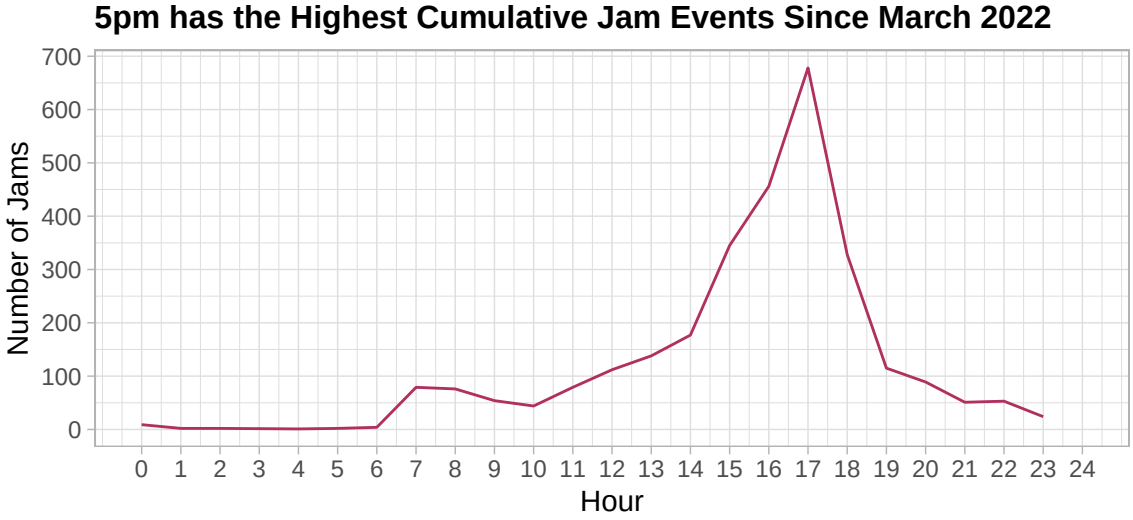
6.1.b

The given location is **South DuSable Lake Shore Drive** where there also is railroad tracks.

- Since it's the entrance to **Navy Pier**, where major celebration events will be held in Chicago, it's highly likely to have traffic jams especially during *holidays*.
- Also, based on research,we get to know that it's always one the most dangerous traffic area in Chicago. This is caused by the design of the surrounding transportation environment. Along lakefront, roads were designed more as *Highway-style* to allow drivers to speed up and increase the capacity. *More traffic lights, grade-crossings were removed* while this area is the intersection of multiple overpasses, making more cars drive at teh same time and easy to cause jams. The special requirement of left-turn only lanes also worsen the situation. Plus, inconvenient design for bicyclists and pedestrians increased the chance of traffic jams. The need to take tunnels to cross the street while sometimes those are not accessible, then they need to cross directly in the path of cars. Considering the high amount of population going to lakefront and Navy Pier everyday, it would have significant effect on traffic jams.
- Besides that, by Googling, we also found that DuSable Lake Shore Drive is closed **due to construction of accessible bridge**. Old bridge was demolished in Summer 2021 and the new pedestrian bridge is planned to open at the end of this year. **Drivers need to take a detour around this location**. In sum, it's highly likely to encounter jams around this area.

6.1.c

```
# Clean data
event_time <- event_df %>%
  filter(type == "JAM") %>%
  mutate(ts_hour = hour(ts_central)) %>%
  filter(between(ts_hour, 0, 24))
# Plot
event_time_plot <- event_time %>%
  count(ts_hour)
event_time_plot %>%
  ggplot(aes(ts_hour, n)) +
  geom_line(color = "maroon") +
  scale_x_continuous(
    breaks = seq(0, 24, 1),
    limits = c(0, 24)
  ) +
  scale_y_continuous(breaks = seq(0, 700, 100)) +
  labs(
    title = "5pm has the Highest Cumulative Jam Events Since March 2022",
    x = "Hour",
    y = "Number of Jams",
    caption = "Source: Wbez; Note: Data only include 1 location"
  ) +
  theme_light() +
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    plot.caption = element_text(size = 8, face = "italic", hjust = 0)
  )
```



Answer

Plot as above. We notice that from midnight to midnight, at this location, 5pm has the highest number of jams between *2021-03-02* to *2022-05-05*. We can also notice that there is a small peak around 7am to 8am, which may be caused by the fact that people drive to their office during the morning. In total, the number

of jams continue to increase since 10am, and started to decline since 5pm. The marginal increase rate also goes up after 2pm.

6.1.d

```
event_time %>%
  distinct(subtype)
```

```
## # A tibble: 4 x 1
##   subtype
##   <chr>
## 1 "JAM_HEAVY_TRAFFIC"
## 2 "JAM_STAND_STILL_TRAFFIC"
## 3 "JAM_MODERATE_TRAFFIC"
## 4 ""
```

```
event_time <- event_time %>%
  filter(grepl("JAM", subtype)) %>%
  mutate(severity = case_when(
    subtype == "JAM_STAND_STILL_TRAFFIC" ~ 1,
    subtype == "JAM_MODERATE_TRAFFIC" ~ 2,
    subtype == "JAM_HEAVY_TRAFFIC" ~ 3
  ))
event_time_index <- event_time %>%
  group_by(ts_hour) %>%
  summarise(
    mean_severity = mean(severity),
    case = n(),
    jam_index = mean_severity * case
  ) %>%
  arrange(ts_hour)
```

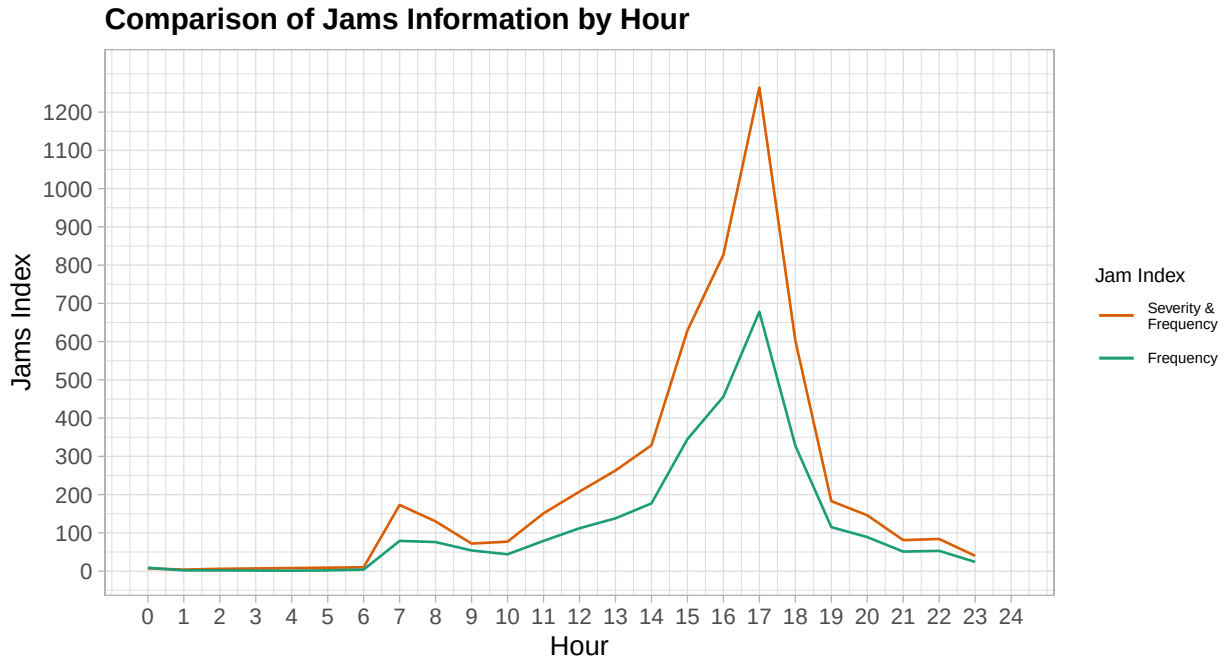
Answer
Among the events classified as "JAM", there are three subtypes(Remove blank type):

- JAM_HEAVY_TRAFFIC
- JAM_STAND_STILL_TRAFFIC
- JAM_MODERATE_TRAFFIC

These values could be recoded into number with hierarchy. For example, **Stand still(1) < Moderate(2) < Heavy in order(3)**. The recoded number could represent the severity of the jam. Thus we would suggest an index for traffic jam that is a **product of *number of jams* and the *recoded values of subtypes*** . Since we are looking at the yearly data, we would **apply the average of the recoded value** when there were multiple jams during the same hour.
It is very simple and has room for improvement (e.g. adding different coefficients, changing the weights for each input) but it is meaningful for integrating both the counts and severity for jams
Note There is another *JAM subtype* called *JAM_SLIGHT_TRAFFIC* but it only contains 7 rows and is not included after we filtered the data. We deleted rows with blank JAM subtype since the solution given during lab session (replace by a common value) is a bit confusing.But if we need to recode blanks with the *slight* type, then slight will be recoded as 1, and heavy will be 4 instead.

6.1.e

```
ggplot() +
  geom_line(
    data = event_time_index,
    aes(ts_hour, jam_index, color = "Severity &\nFrequency")
  ) +
  geom_line(
    data = event_time_plot,
    aes(ts_hour, n, color = "Frequency")
  ) +
  scale_x_continuous(
    breaks = seq(0, 24, 1),
    limits = c(0, 24)
  ) +
  scale_y_continuous(
    breaks = seq(0, 1200, 100),
    limits = c(0, 1300)
  ) +
  scale_color_manual(
    values = c(
      "Severity &\nFrequency" = "#D95F02",
      "Frequency" = "#1B9E77"
    ),
    name = "Jam Index"
  ) +
  labs(
    title = "Comparison of Jams Information by Hour",
    x = "Hour",
    y = "Jams Index",
    caption = "Source: Wbez; Note: Data only include 1 location"
  ) +
  theme_light() +
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    plot.caption = element_text(size = 8, face = "italic", hjust = 0),
    legend.title = element_text(size = 8),
    legend.text = element_text(size = 6)
  )
```



Source: Wbez; Note: Data only include 1 location

Answer

Based on the plot above, we observe quite similar pattern of the jam index by hour in the location we selected. This reflects that those hours with higher number of jam cases also had more sever jams on average. However, we also notice a significant increase poof jam index around 7am. It indicates that the jam severity was high especially around 7am. Also, we notice huge increase in jam index between 10am to 14pm, it also tells us that although this is not the time period which we observed the most jams, the jam severity was pretty high on average. The trend continues until we reached the jam peak at 5pm, and the decrease rate also has increased, which means the jam severity declines along with less jam cases after 5pm.

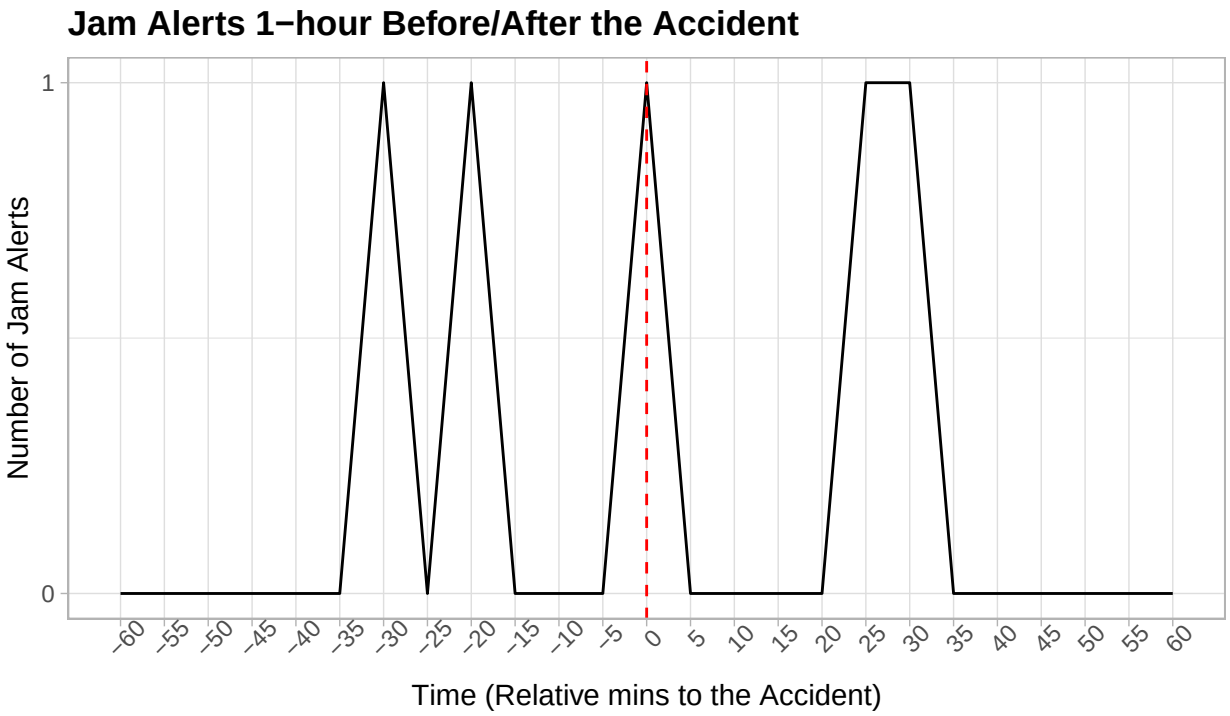
7. Waze aggregate over multiple events

7.1

```
# Select the uuid
alerts_major <- all_alerts %>%
  filter(grepl("MAJOR", subtype))

id <- "2c78c881-16b4-4a95-a67b-db866eaa8344"
id_ts <- alerts_major %>%
  filter(uuid == id) %>%
  select(ts_central_round) %>%
  pull()

id_long <- alerts_major %>%
  filter(uuid == id) %>%
  select(long) %>%
  pull()
```

```r
id_lat <- alerts_major %>%
  filter(uuid == id) %>%
  select(lat) %>%
  pull()

# Turning 1km to latitude/longitude
# 1 degree of latitude = 40075/360
# 1km = 360/40075 km
km_lat <- 360 / 40075
km_long <- 360 / (cos(id_lat * pi / 180) * 40075)

# 5-min intervals
intervals <- seq(-60, 60, 5)

# Create the sample dataset
sample_jam <- all_alerts %>%
  mutate(
    interval = difftime(ymd_hms(ts_central_round, tz = "America/Chicago"),
                        ymd_hms(id_ts, tz = "America/Chicago"), units = "mins"),
    interval = as.numeric(interval)
  ) %>%
  filter(between(interval, -60, 60) &
    between(lat, id_lat - km_lat / 2, id_lat + km_lat / 2) &
    between(long, id_long - km_long / 2, id_long + km_long / 2)) %>%
  filter(type == "JAM") %>%
  count(interval) %>%
  complete(
    interval = intervals,
    fill = list(n = 0L)
  )

# Plot
ggplot() +
  geom_line(
    data = sample_jam,
    aes(x = interval, y = n)
  ) +
  geom_vline(
    xintercept = 0,
    color = "red",
    linetype = 2
  ) +
  scale_x_continuous(
    breaks = seq(-60, 60, 5),
    minor_breaks = seq(-60, 60, 5)
  ) +
  scale_y_continuous(breaks = seq(0, 2, 1)) +
  labs(
    title = "Jam Alerts 1-hour Before/After the Accident",
    x = "Time (Relative mins to the Accident)",
    y = "Number of Jam Alerts",
    caption = "Source: WAZE"
  ) +
```

```
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0, face = "bold"),
    plot.caption = element_text(hjust = 0., face = "italic"),
    axis.text.x = element_text(angle = 45)
  )
```



Source: WAZE

Answer

- Select id: 2c78c881-16b4-4a95-a67b-db866eaa8344
- Selected time period: 2022-04-01 14:50:00 CDT to 2022-04-01 16:50:00 CDT

Reference https://stackoverflow.com/questions/4000886/gps-coordinates-1km-square-around-a-point# https://stackoverflow.com/questions/1253499/simple-calculations-for-working-with-lat-lon-and-km-distance

7.2

```
# Write function, and set geo buffer default to 1km box
alert_around_accident <- function(df, id, dist = 1) {
  # match uuid, extract. long, lat, time
  acc_time <- df %>%
    filter(uuid == id) %>%
    select(ts_central_round) %>%
    pull()
  acc_long <- df %>%
```

```
    filter(uuid == id) %>%
    select(long) %>%
    pull()
  acc_lat <- df %>%
    filter(uuid == id) %>%
    select(lat) %>%
    pull()
  # set scale for converting 1km
  km_lat <- 360 / 40075 * dist
  km_long <- 360 / (cos(acc_lat * pi / 180) * 40075) * dist
  # set interval limits
  intervals <- seq(-60, 60, 5)
  # filter
  df_plot <- df %>%
    mutate(
      interval = difftime(ymd_hms(`ts_central_round`, tz = "America/Chicago"),
        ymd_hms(`acc_time`, tz = "America/Chicago"),
        units = "mins"
      ),
      interval = as.numeric(interval)
    ) %>%
    filter(between(interval, -60, 60) &
      between(lat, acc_lat - km_lat / 2, acc_lat + km_lat / 2) &
      between(long, acc_long - km_long / 2, acc_long + km_long / 2)) %>%
    filter(type == "JAM") %>%
    count(interval) %>%
    complete(
      interval = intervals,
      fill = list(n = 0L)
    )

  return(df_plot)
}
```

7.3

```
# Check with Q7.1
alert_around_accident(all_alerts, "2c78c881-16b4-4a95-a67b-db866eaa8344") %>%
  arrange(desc(n))
```

```
## # A tibble: 25 x 2
##    interval     n
##       <dbl> <int>
##  1      -30     1
##  2      -20     1
##  3        0     1
##  4       25     1
##  5       30     1
##  6      -60     0
##  7      -55     0
##  8      -50     0
```

```
##  9      -45     0
## 10      -40     0
## # ... with 15 more rows
```

```
# use to pick id
alerts_major %>%
  arrange(ts_central) %>%
  filter(uuid == "158f2dd9-a3f9-4c2d-a7a1-f25b491937b6") %>%
  select(uuid, ts_central_round)
```

```
## # A tibble: 1 x 2
##   uuid                                 ts_central_round
##   <chr>                                <dttm>
## 1 158f2dd9-a3f9-4c2d-a7a1-f25b491937b6 2021-03-02 14:05:00
```

```
alert_around_accident(all_alerts, "158f2dd9-a3f9-4c2d-a7a1-f25b491937b6") %>%
  arrange(desc(n))
```

```
## # A tibble: 25 x 2
##    interval     n
##       <dbl> <int>
##  1      -20     1
##  2       -5     1
##  3       60     1
##  4      -60     0
##  5      -55     0
##  6      -50     0
##  7      -45     0
##  8      -40     0
##  9      -35     0
## 10      -30     0
## # ... with 15 more rows
```

Answer
We get the same answer with function as in Q7.1 with the same _uuid___. We also successfully get a dataframe with another testing **uuid**.

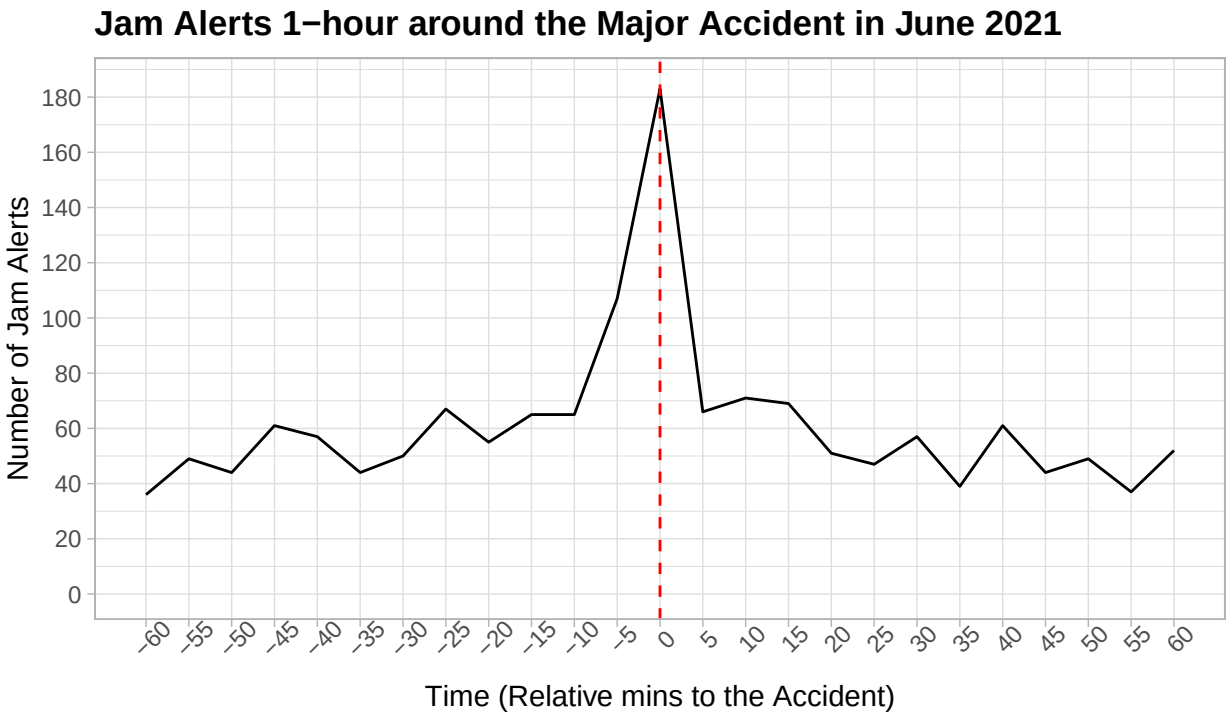
7.4

```
# Major accident in June 2021
major_accident_june <- alerts_major %>%
  filter(ts_date_central < ymd("2021-07-01") & ts_date_central >= ymd("2021-06-01"))
june_alerts <- all_alerts %>%
  filter(ts_central_round <= ymd_hms("2021-07-01 00:40:00", tz = "America/Chicago") &
    ts_central_round >= ymd_hms("2021-05-31 23:20:00", tz = "America/Chicago"))
# Apply to each row
id_loop_major <- c(major_accident_june$uuid)
major <- id_loop_major %>%
  map_dfc(~ alert_around_accident(june_alerts, .)) %>%
  select(-starts_with("interval"))
```

```r
# Calculate the sum and plot
major_plot <- as.data.frame(rowSums(major))
major_plot <- as.data.frame(cbind(intervals, major_plot))
colnames(major_plot) <- c("interval", "jamsum")
```

7.5

```r
# Plot
ggplot(major_plot) +
  geom_line(aes(x = interval, y = jamsum)) +
  geom_vline(
    xintercept = 0,
    color = "red",
    linetype = 2
  ) +
  scale_x_continuous(
    breaks = seq(-60, 60, 5),
    minor_breaks = seq(-60, 60, 5)
  ) +
  scale_y_continuous(
    breaks = seq(0, 180, 20),
    limits = c(0, 185)
  ) +
  labs(
    title = "Jam Alerts 1-hour around the Major Accident in June 2021",
    x = "Time (Relative mins to the Accident)",
    y = "Number of Jam Alerts",
    caption = "Source: WAZE"
  ) +
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0, face = "bold"),
    plot.caption = element_text(hjust = 0., face = "italic"),
    axis.text.x = element_text(angle = 45)
  )
```



Source: WAZE

7.6

Answer

As we can see in the plot above, **the number of jams continue to increase and reached the peak when the major accident happens**, then it starts to decline after the *major accident* happens. If the *major accident causes jams*, then the number of jams should increase rapidly soon after the major accident happens. However, in the plot, we notice the trend of increasing jams prior to the major accident first happened and declining trend after the accident. Therefore, it's **much more likely that the converging jam cases cause accidents**, although we need more detailed and solid analysis to draw the conclusion

7.7

```
# Functions for minor accident
minor_accident_june <- all_alerts %>%
  filter(ts_central_round < ymd("2021-07-01") & ts_central_round >= ymd("2021-06-01")) %>%
  filter(grepl("MINOR", subtype))

id_loop_minor <- c(minor_accident_june$uuid)
minor <- id_loop_minor %>%
  map_dfc(~alert_around_accident(june_alerts,.)) %>%
  select(-starts_with("interval"))
# Calculate the sum and plot
minor_plot <- as.data.frame(rowSums(minor))
minor_plot <- as.data.frame(cbind(intervals, minor_plot))
colnames(minor_plot) <- c("interval", "jamsum")
```

```r
# Calculation
major_plot %>%
  mutate(pect_diff = (jamsum / lag(jamsum)) - 1) %>%
  filter(interval %in% c(-10, -5, 0, 5))
```

```
##   interval jamsum pect_diff
## 1      -10     65     0.000
## 2       -5    107     0.646
## 3        0    183     0.710
## 4        5     66    -0.639
```

```r
minor_plot %>%
  mutate(pect_diff = (jamsum / lag(jamsum)) - 1) %>%
  filter(interval %in% c(-10, -5, 0, 5))
```

```
##   interval jamsum pect_diff
## 1      -10    109    -0.121
## 2       -5    129     0.183
## 3        0    183     0.419
## 4        5     99    -0.459
```

```r
# Increase Rate for Major Accident
major_plot$jamsum[major_plot$interval == "0"] /
  major_plot$jamsum[major_plot$interval == "-10"] -1
```

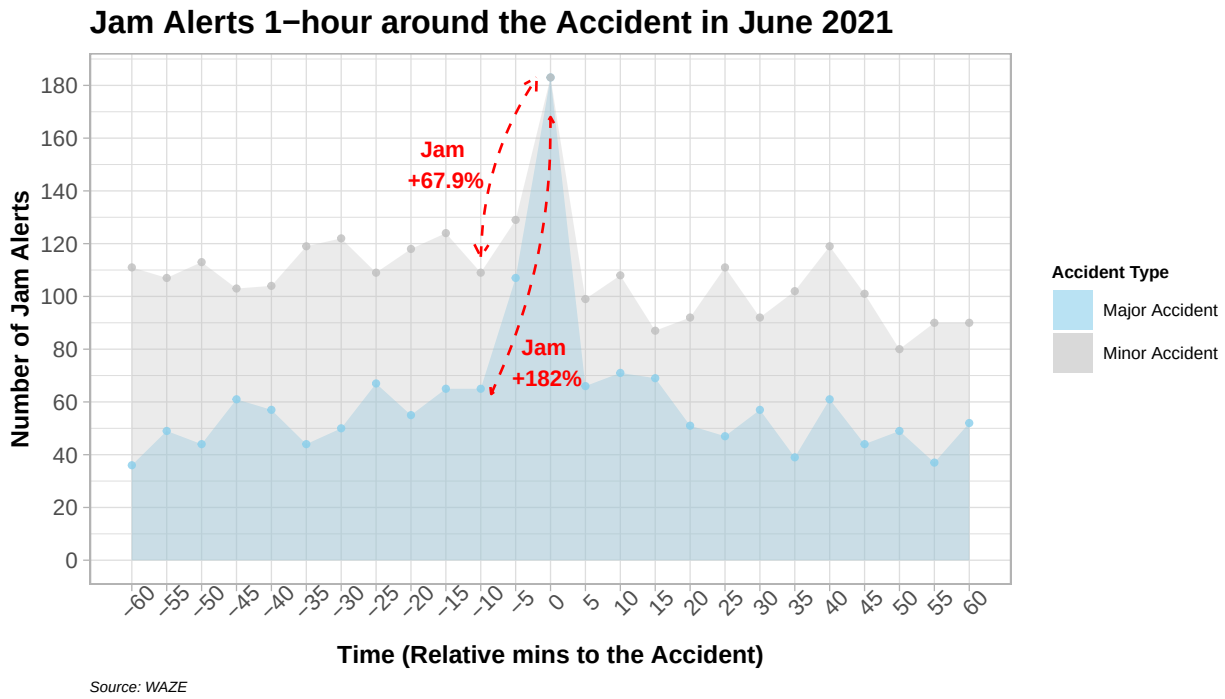
```
## [1] 1.82
```

```r
# Increase Rate for Minor Accident
minor_plot$jamsum[minor_plot$interval == "0"] /
  minor_plot$jamsum[minor_plot$interval == "-10"] -1
```

```
## [1] 0.679
```

```r
# Plot
ggplot() +
  geom_area(data = major_plot,
            aes(interval, jamsum,
                fill = "Major Accident"),
            alpha = 0.4
            ) +
  geom_area(data = minor_plot,
            aes(interval, jamsum,
                fill = "Minor Accident"),
            alpha = 0.3
            ) +
 geom_point(data = major_plot,
            aes(interval, jamsum),
            color = "skyblue",
            alpha = 0.8,
            shape = 20
            ) +
```

```
   geom_point(data = minor_plot,
          aes(interval, jamsum),
          color = "grey",
          alpha = 0.8,
          shape = 20
          ) +
scale_x_continuous(breaks = seq(-60, 60, 5),
                   minor_breaks = seq(-60, 60, 5)) +
scale_y_continuous(breaks = seq(0, 200, 20)) +
scale_fill_manual(values = c("Major Accident" = "skyblue",
                             "Minor Accident" = "grey"),
                   name = "Accident Type") +
labs(title = "Jam Alerts 1-hour around the Accident in June 2021",
     x = "Time (Relative mins to the Accident)",
     y = "Number of Jam Alerts",
     caption = "Source: WAZE") +
theme_light() +
theme(plot.title = element_text(hjust = 0, face = "bold"),
      plot.caption = element_text(size = 6, hjust = 0, face = "italic"),
      axis.title = element_text(size = 10, face = "bold"),
      axis.text.x = element_text(angle = 45),
      legend.title = element_text(size = 7, face = "bold"),
      legend.text = element_text(size = 7)) +
geom_curve(aes(x = -10, y = 115, xend = -2, yend = 183),
         arrow = arrow(ends = "both",length = unit(.2,"cm")),
         color = "red", curvature = -0.15,
         linetype = 2) +
geom_curve(aes(x = -8.5, y = 63, xend = 0, yend = 168),
         arrow = arrow(ends = "both",length = unit(.1,"cm")),
         color = "red", curvature = 0.12,
         linetype = 2) +
annotate('text', x = -15, y = 150,
         label = "Jam \n+67.9%", color = "red", fontface = "bold", size = 3) +
annotate('text', x = -0.5, y = 75,
         label = "Jam \n+182%", color = "red", fontface = "bold", size = 3)
```



Answer

As we can see in the plot above, the increase in jam alerts is large for both major and minor accidents. Specifically, **the increase in jams 10 mins prior to minor accident is 67.9%,** while the increase in jams for **major accident is much higher, which is 182%**. **The difference between the increase percentage is nearly 114%**. It's quite reasonable since major accidents are usually more severe than minor accidents, and it won't be likely to happen if the jam alerts cases haven't increased greatly, given our observation in Q7.5 that *jams cause accidents*