# STA2453_Lab1

Elisa Du 1006840885

**Reference**: Lab 1 - Data Wrangling

Import necessary R packages first, then import the emergency department patient encounters dataset.

```
# import libraries
library(dplyr)
library(tidyverse)
library(lubridate)
library(ggplot2)

# read in data
ed_data <- read_csv('raw_ed_data.csv')
```

## Exercise 1

1. There are 80,464 observations in the dataset. We would ask the chief of the ED for the total number of ED visits in previous years, as well as the average number of visits per month in past years, to gauge if in any particular month from January 2019 to January 2020 there has been a higher or lower patient volume than usual. For those unusual months, there may be duplicate existing records, undocumented ED visits, amongst other reasons.
2. The correction made to the `present_complaints` column is reasonable, since it reduces 27 manually typed values to the unique 17 values as listed in the dataset documentation. For completeness, we perform the tidying below using the code in-class.

```
## Step 1
# get rid of leading and trailing white spaces
# convert to lowercase
# replace multiple spaces w single space
clean_complaint <- function(column){
  clean_col = trimws(column)
  clean_col <- gsub(pattern = ' +', replacement = ' ', x = clean_col)
  clean_col = tolower(clean_col)
  return (clean_col)
}
#corrected: 'back pain','chest pain', 'burns', 'rash'
ed_data <- ed_data %>%
  mutate(presenting_complaint = clean_complaint(presenting_complaint))

## Step 2
# fix spelling errors
ed_data <- ed_data %>%
  mutate(presenting_complaint = case_when(
    presenting_complaint == "chest pian" ~ "chest pain",
    presenting_complaint == "burns" ~ "burn",
    presenting_complaint == "traumatic injuries" ~ "traumatic injury",
```

```
    presenting_complaint %in% c("unk", "missing") ~ "unknown",
    presenting_complaint == "headach" ~ "headache",
    TRUE ~ presenting_complaint))

# check
unique(ed_data$presenting_complaint) %>% knitr::kable()
```

| x |
|---|
| sore throat |
| lower extremity injury |
| back pain |
| abdominal pain |
| loss of hearing |
| upper extremity injury |
| confusion |
| rash |
| chest pain |
| unknown |
| headache |
| trouble breathing |
| bizarre behaviour |
| traumatic injury |
| hallucinations |
| general weakness |
| burn |

```
length(unique(ed_data$presenting_complaint))
```

```
## [1] 17
```

3. We identify some additional data quality issues:

- Spanning Jan 2019 to Jan 2020, we expect an average of 80464/13 ~ 6190 visits per month. We take a look at per-month arrival volumes to check for anything unusual, such as missing or undocumented visits.

```
ed_data$ed_start_YrMonth<-format(ed_data$ed_start_time,"%Y-%m")
ed_data %>%
  count(ed_start_YrMonth) %>%
  knitr::kable()
```

| ed_start_YrMonth | n |
|---|---|
| 2019-01 | 6615 |
| 2019-02 | 6031 |
| 2019-03 | 6906 |
| 2019-04 | 5509 |
| 2019-05 | 7025 |
| 2019-06 | 6668 |
| 2019-07 | 7209 |
| 2019-08 | 7062 |
| 2019-09 | 6659 |
| 2019-10 | 6804 |
| 2019-11 | 6621 |

| ed_start_YrMonth | n |
|---|---|
| 2019-12 | 6527 |
| 2020-01 | 35 |
| NA | 793 |

There are only 35 observations in January 2020, something to ask the chief of ED about. We should also ask about the 793 missing observations in `ed_start_time`. Why are the arrival times for those specific patients missing?

We next examine if there are any missing values in other columns of the dataset. The code below is taken from the class demo.

```
count_missing <- function(x){

  num_missing = sum(is.na(x))
  return(num_missing)

}
# count no. of missing values in each column
ed_data %>% summarise_all(.funs = count_missing) %>% knitr::kable()
```

| ENCOUNTER_NO | CHASN_NO | CTAS | CTAS_Desc | ed_start_time | ed_end_time | ed_pia_time | admin_start_admitted | los | presenting_complaint | ed_start_YrMonth |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 793 | 396 | 0 | 68849 | 0 | 1186 | 0 | 793 |

- 396 patients do not have their `ed_end_time` documented.
- There are 1186 observations missing for the system-generated `LOS` variable. Why did the system fail to compute them?
- We should also check for each of the 3 date columns if all dates are within the bound of year 2019 to 2020.

```
unique(format(ymd_hms(ed_data$ed_start_time), '%Y'))
```

```
## [1] "2019" NA     "2020"
```

```
unique(format(ymd_hms(ed_data$ed_end_time), '%Y'))
```

```
## [1] "2019" "2018" NA     "2020"
```

```
unique(format(ymd_hms(ed_data$ed_pia_time), '%Y'))
```

```
## [1] "2019" "2099" "2020"
```

We see that for `ed_end_time` column, certain record(s) contains the year `2018`, and for the `ed_pia_time` column, certain record(s) contains the typo of year `2099`.

- We should also check that the patient arrival time recorded is *earlier* than the patient departure time, and similarly, the time a physician is seen is later than patient arrival time but earlier than the departure time.

```
all(ed_data$ed_start_time < ed_data$ed_end_time)
```

```
## [1] FALSE
```

```
all(ed_data$ed_start_time < ed_data$ed_pia_time)
```

```
## [1] FALSE
```

```r
all(ed_data$ed_pia_time < ed_data$ed_end_time)
```

```
## [1] FALSE
```

We see that in all 3 cases there are errors made in the date-time recorded.

- We also check for number of unique days on which there are ED arrivals recorded.

```r
length(unique(as.Date(ed_data$ed_start_time)))
```

```
## [1] 363
```

There are 363 days on which ED arrivals were recorded, which is less than the number of days in a year. Since we are told the data is from January 2019 to January 2020, spanning 13 months, we should inquire why there are missing dates on which encounters occurred.

- Next we check for duplicates in the data.

```r
dup_ed_data <- unique(ed_data[duplicated(ed_data) | duplicated(ed_data, fromLast = TRUE) , ])
#glimpse(dup_ed_data)
```

Next, we take a closer look at the `ENCOUNTER_NUM` recorded for each patient.

```r
patient_ID <- ed_data$ENCOUNTER_NUM
# any(duplicated(patient_ID))
length(patient_ID[duplicated(patient_ID)])
```

```
## [1] 216
```

We find there is 1 duplicate patient ID, even after duplicate records are removed. Perhaps this indicates 1 record where a unique patient was mistakenly assigned a duplicate ID.

- We also note there are 68849 entries missing for the `adm_start_time` column. We should expect 68849 0's in the corresponding `admitted` column. Let's check this:

```r
count(ed_data, admitted)
```

```
## # A tibble: 2 x 2
##   admitted     n
##      <dbl> <int>
## 1        0 68848
## 2        1 11616
```

Turns out there are 68848 patients who were not admitted, one less than the number of missing values in the `adm_start_time` column. This indicates admission time was not recorded for 1 patient that was in fact admitted.

- We take a look at the CTAS score values, which are in the `CTAS_CD` and `CTAS_DESCR` columns:

```r
count(ed_data, CTAS_DESCR, sort = T)
```

```
## # A tibble: 6 x 2
##   CTAS_DESCR        n
##   <chr>         <int>
## 1 URGENT        34772
## 2 EMERGENCY     26088
## 3 SEMI-URGENT   12046
## 4 RESUSCITATION  3303
## 5 NON URGENT     3049
## 6 N/A            1206
```

```r
count(ed_data, CTAS_CD, sort = T)
```

```
## # A tibble: 6 x 2
##   CTAS_CD     n
##   <chr>   <int>
## 1 3       34772
## 2 2       26088
## 3 4       12046
## 4 1        3303
## 5 5        3049
## 6 N/A      1206
```

```r
# unique(ed_data$CTAS_DESCR)
# unique(ed_data$CTAS_CD)
```

We see there are 1201 missing entries in each of the `CTAS_CD` and `CTAS_DESCR` columns. They were previously not detected, since they are recorded as `N/A` rather than `NA`. We should ask the chief about these, and perhaps consider manually inputting the triage code based on their health records, on the pretense their privacy is protected.

## Exercise 2

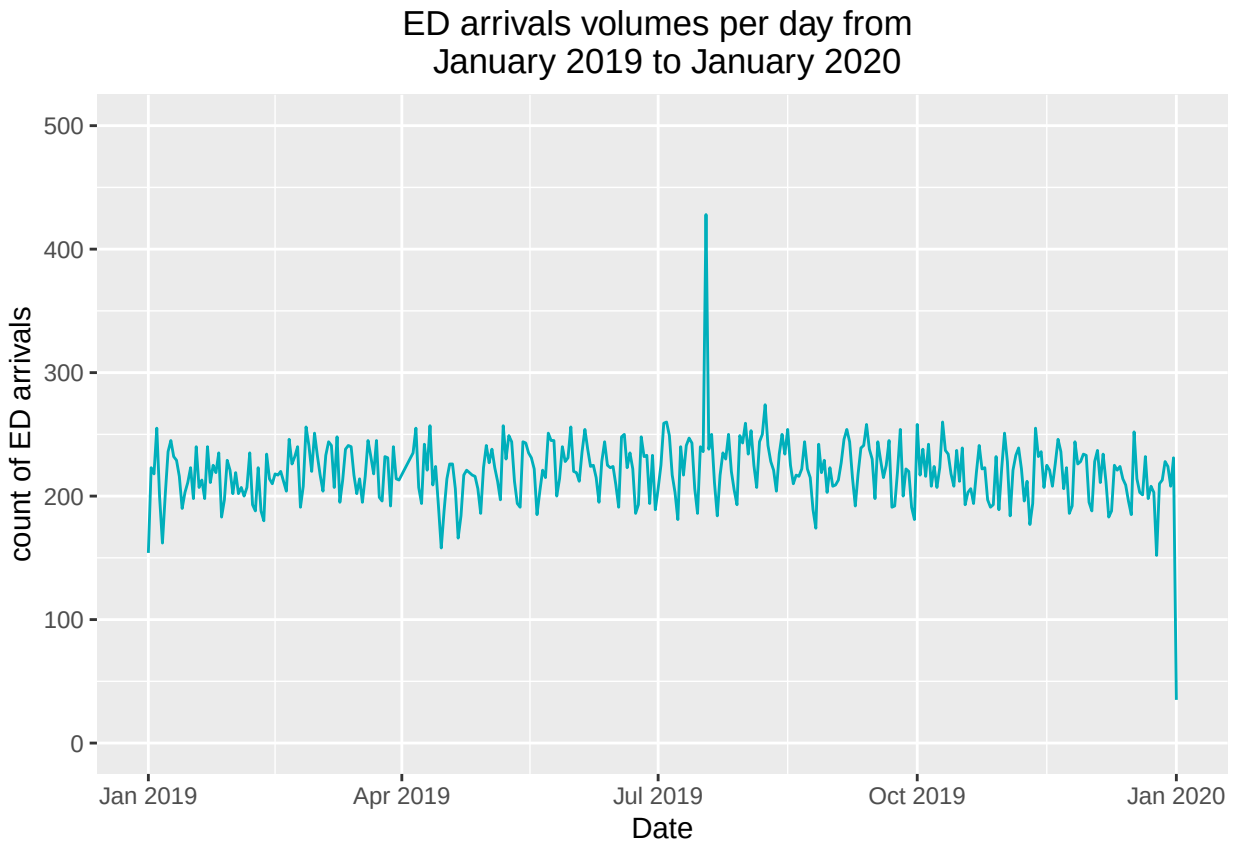
To plot the number of ED arrivals per day, we first generate a dataset of arrival volumes per day. Note than the line graph generated is without correcting for any data quality issues mentioned prior.

```r
# arrival volumes per day
df <- ed_data
df$date_of_visit = as.Date(ed_data$ed_start_time) # extract year-month

count_by_day = df %>% count(date_of_visit) %>% # counts of visits per day
  filter(!is.na(date_of_visit)) # remove NA values for plotting

count_by_day %>%
  head() %>%
  knitr::kable()
```

| date_of_visit | n |
|---|---|
| 2019-01-01 | 154 |
| 2019-01-02 | 223 |
| 2019-01-03 | 218 |
| 2019-01-04 | 255 |
| 2019-01-05 | 199 |
| 2019-01-06 | 162 |

```r
# base plot
g <- count_by_day%>%
  ggplot(aes(x = date_of_visit, y = n )) +
  geom_line(color = '#00AFBB', size = 0.5)

# plot formatting
g + ggtitle('ED arrivals volumes per day from \n January 2019 to January 2020' ) +
   theme(plot.title = element_text(hjust = 0.5)) +
  xlab('Date') + ylab('count of ED arrivals') +
```

```
  ylim(0,500)
```




We see the line graph stops abruptly at the beginning of Jan 2020 which could indicate missing records for that particular month. Furthermore, there is a drastic dip in January 2020 which indicates missing ED records for a particular day in January 2020. The sudden peak in ED arrivals around mid-July 2019 should also be investigated.

```
count_by_day[which.max(count_by_day$n),]
```

```
## # A tibble: 1 x 2
##   date_of_visit     n
##   <date>        <int>
## 1 2019-07-18      428
```

```
count_by_day[which.min(count_by_day$n),]
```

```
## # A tibble: 1 x 2
##   date_of_visit     n
##   <date>        <int>
## 1 2020-01-01       35
```

**We see there are 428 documented ED arrivals on July 18, 2019. But only 35 documented ED arrivals on January 1, 2020. These are data quality issues to investigate further.**

The average number of arrivals to the ED per day is calculated below. Note that this does not account for the missing values in `ed_start_time` column.

```
round(mean(count_by_day$n))
```

```
## [1] 220
```

**There is an average of 220 arrivals to the ED per day from January 2019 to January 2020.**

## Exericse 3

To find the average and standard deviation of volumes on weekends and weekdays, we need to do some data aggregation first. We extract the year and month from the `ed_start_time` column, and count the ED arrival volumes per day. Then, we label each row as weekday or weekend. This generates the dataset `count_by_day`.

```
# prep data
df <- ed_data
df$arrival_date = as.Date(ed_data$ed_start_time) # extract year-month

count_by_day = df %>% count(arrival_date) %>% # count ED arrivals per day
  filter(!is.na(arrival_date)) # remove missing ED arrival times

count_by_day$day_of_week <- wday(count_by_day$arrival_date, label = T, abbr = T) #BM

count_by_day$day_of_week <- as.character(count_by_day$day_of_week)
count_by_day <- count_by_day %>%
  mutate(day_type = case_when(
    day_of_week %in% c('Mon', 'Tue', 'Wed', 'Thu', 'Fri') ~ 'weekday',
    day_of_week %in% c('Sat', 'Sun') ~ 'weekend',
    TRUE ~ day_of_week
  ))

count_by_day
```

```
## # A tibble: 362 x 4
##    arrival_date     n day_of_week day_type
##    <date>       <int> <chr>       <chr>
##  1 2019-01-01     154 Tue         weekday
##  2 2019-01-02     223 Wed         weekday
##  3 2019-01-03     218 Thu         weekday
##  4 2019-01-04     255 Fri         weekday
##  5 2019-01-05     199 Sat         weekend
##  6 2019-01-06     162 Sun         weekend
##  7 2019-01-07     201 Mon         weekday
##  8 2019-01-08     236 Tue         weekday
##  9 2019-01-09     245 Wed         weekday
## 10 2019-01-10     232 Thu         weekday
## # ... with 352 more rows
```

We find the average and standard deviation of volumes on weekends and weekdays.

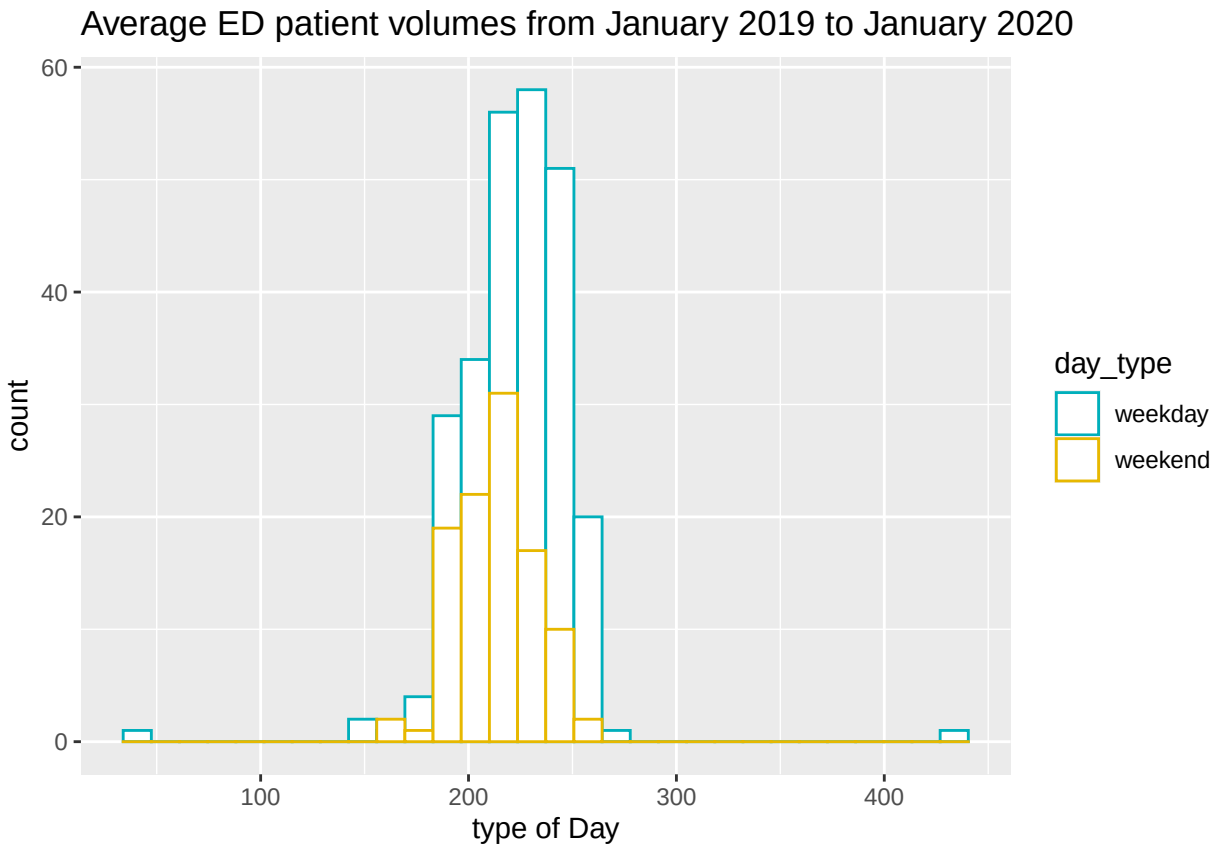
```
list_day <- list(count_by_day$day_type)
mean_temp <- count_by_day$n %>%
        aggregate(list_day, FUN = mean) %>%
        rename(avg=x)
sd_temp <- count_by_day$n %>%
  aggregate(list_day, FUN = sd) %>%
  rename(sd=x)
mean_sd <- merge(mean_temp, sd_temp, by = 'Group.1')
mean_sd
```

```
##   Group.1      avg       sd
## 1 weekday 222.8682 27.45378
```

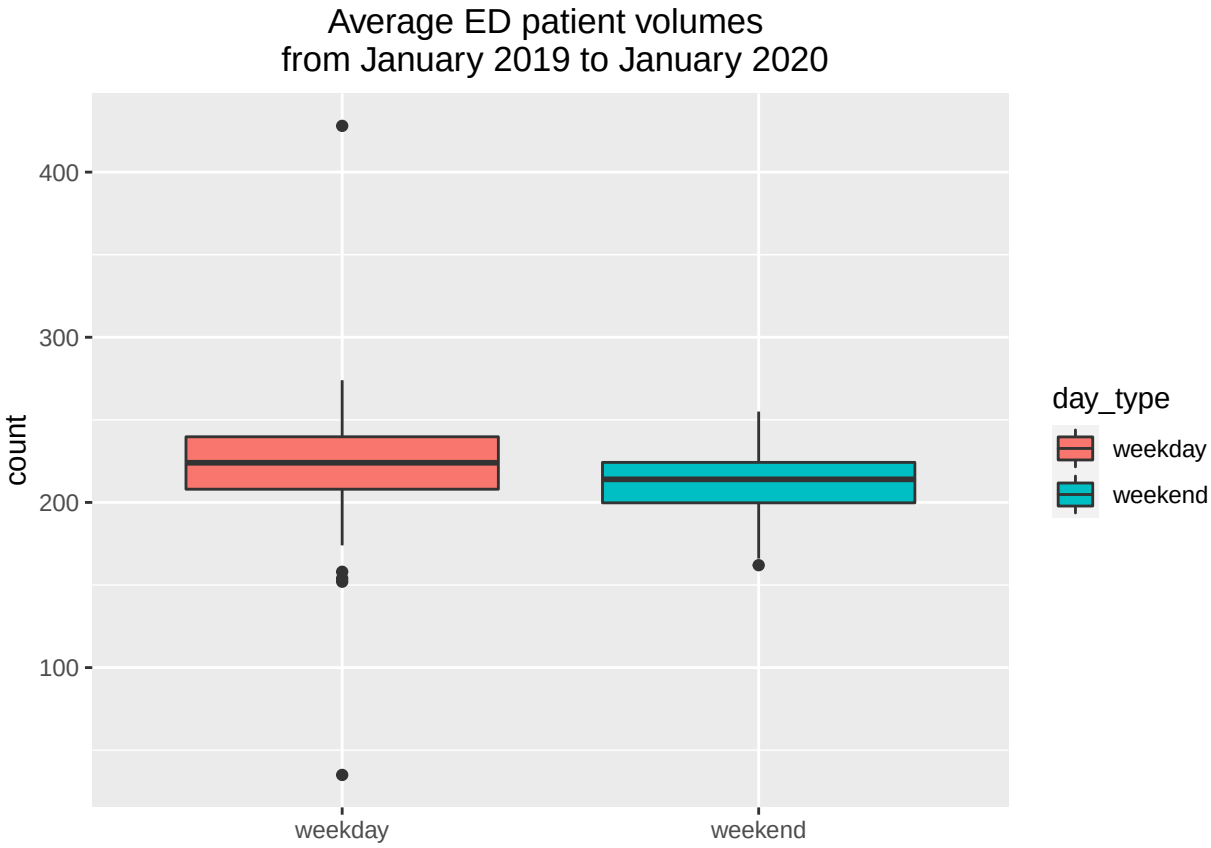
```
## 2 weekend 213.1827 19.06034
```

Using `mean_sd` dataset, we can then plot the histogram and boxplot.

```
# histogram
g2 <- ggplot(count_by_day, aes(x = n)) +
  geom_histogram(aes(color = day_type),fill = 'white',
                 position = "identity", bins = 30) +
  scale_color_manual(values = c("#00AFBB", "#E7B800"))

g2 + ggtitle('Average ED patient volumes from January 2019 to January 2020') +
  ylab('count') +
  xlab('type of Day')
```




```
# boxplot
g3 <- ggplot(data = count_by_day, aes(x = day_type, y = n, fill = day_type)) +
  geom_boxplot()
g3 +ggtitle('Average ED patient volumes \n from January 2019 to January 2020') +
  ylab('count') +
  theme(axis.title.x = element_blank(), plot.title = element_text(hjust = 0.5))
```




From the plots we see there is an apparent difference in arrival volumes between weekdays and weekends. But further hypothesis testing (see Exercise 4) would be needed to confirm this. Although histograms are typically more easily understood by a non-technical audience, a boxplot is more effective in showing the difference in patient arrival distributions between weekdays and weekends. Specifically, a boxplot shows the *spread* of the distribution, including the quartiles and the outliers in the arrivals distribution, as well as any skewness. However, boxplots cannot depict peaks in the distribution, but since the data is not multimodal but rather clustered around the mean, this makes boxplot more ideal for visualizing the difference in distribution. By observing the boxplot, we see there is a larger centre of distribution (i.e. median), as well as a higher variance around the way, for weekday ED arrivals than that for weekends.

## Exercise 4

We can use the same dataset `count_by_day` in Exercise 4 to conduct a two-sample t-test, with the null hypothesis being that there is no difference in the mean weekday and weekend ED arrival volumes.

```
# two-sample t-test
t.test(n~day_type, data = count_by_day, var.equal = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  n by day_type
## t = 3.8242, df = 271.27, p-value = 0.0001628
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   4.699269 14.671780
## sample estimates:
## mean in group weekday mean in group weekend
```

```
##                222.8682                213.1827
```

We get a t-statistic value of ~ 3.82, and p-value ~ 0.00016 which is much smaller than 0.05. Also notice the 95% confidence interval (4.699269, 14.671780) does not include mean difference of 0 in the interval. This indicates at the 5% significance level, we can reject the null hypothesis of there being no difference in mean arrival counts on weekends versus weekdays. In other words, the oberved difference in average arrivals between weekends and weekdays is statistically significant, rather than occurring by chance.

## Exercise 5

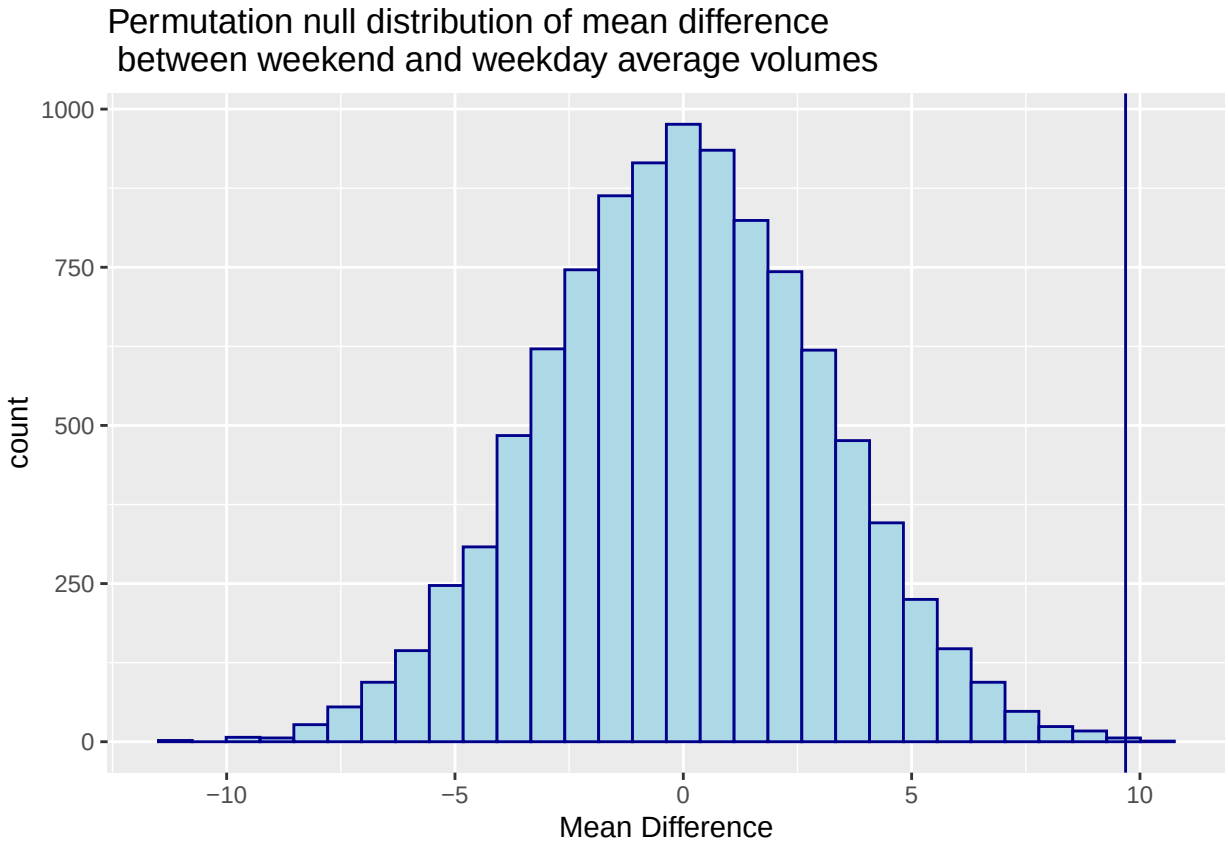
Using the CASI Text reference we conduct a permutation test. We permute the 258 weekend and weekday samples, and under the null hypothesis, randomly sample the 2 group labels (weekdays and weekends), then calculate the difference in mean arrival volumes. We repeat this procedure 10,000 times. Then we plot the distribution for these 10,000 sample mean difference values.

```
# difference in mean for original sample
list_day <- list(count_by_day$day_type) # convert to list to aggregate
mean_day <- aggregate(count_by_day$n, list_day, mean)

diff_means <- mean_day$x[1] - mean_day$x[2]
diff_means
```

```
## [1] 9.685525
```

```
# permutation test
iter <- 10000
# count(count_by_day, day_type)
threshold <- 104 # 104 weekend samples and 258 weekday samples
mean_diff_vec <- rep(0,iter) # store mean difference from permuted samples
arrival_val <- count_by_day$n # arrival counts in original sample
indices <- seq( 1: dim(count_by_day)[1] ) # indices to permute

for (i in 1:iter){
  # set seed for reproducibility
  set.seed(i)
  # random permutation of indices
  permuted_indices <- sample(indices)
  # shuffle sample based on permuted indices
  temp <- arrival_val[permuted_indices]
  # sample the 2 groups (weekday vs. weekend) using permuted indices (under assumption of null hypothes
  weekend <- temp[1:threshold]
  weekday <- temp[(threshold +1) : length(temp)]
  # difference in means between 2 groups that are randomly sampled
  mean_1 <- mean(weekend)
  mean_2 <- mean(weekday)
  diff_in_means = mean_1 - mean_2
  # store sample mean in array
  mean_diff_vec[i] <- diff_in_means
}

# plot histogram
g3 <- ggplot(as.data.frame(mean_diff_vec), aes(x = mean_diff_vec)) + geom_histogram(color="darkblue", fi
g3 + xlab('Mean Difference') + ggtitle('Permutation null distribution of mean difference \n between weel
  geom_vline(xintercept = diff_means, colour = 'darkblue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




The *original* difference in mean arrival volumes , which is ~9.69, is labelled with the blue line in the histogram. Finding the 2-sided p-value is equivalent to finding the proportion of the 10,000 mean difference values (from random permutation) that are more extreme than the observed difference in means.

```
#  2-sided p-value
num_less <- length(mean_diff_vec[mean_diff_vec < - diff_means])
num_more <- length(mean_diff_vec[mean_diff_vec > diff_means])
p_val = (num_less + num_more) / length(mean_diff_vec)
p_val
```

```
## [1] 8e-04
```

The two-sided p-value from the permutation null distribution is 0.0008 which is even lower than the two-sided t-test p-value. This implies there is a mere 0.08% chance of seeing values more extreme than the observed difference in mean (blue line in hisotgram) when the null hypothesis of no difference in patient volumes between weekdays and weekeends holds .In other words, we can safely reject the null hypothesis, and conclude there is indeed a statistically significant difference in the average ED patient arrival volumes betwen weekends and weekdays.

## Exercise 6

Following the tutorial on calculating the census volumes we obtain the dataset below, with the timestamps arranged by chronological order, and `counter` column keeping track of the number of new patients that arrived or departed at each hour on that date. The `volume` column shows the cumulative number of patients (i.e. census volume) at that hour on that date. But since we want to find the *average* census volumes at each hour, we need to aggregate the data further.

```
ed_data <- ed_data[complete.cases(ed_data[ , 4:5]), ] # remove NA values for arrival and departure time
```

```r
# round arrival and departure times to nearest hour
floor_arrive <- floor_date(ed_data$ed_start_time, unit = 'hour')
floor_depart <- floor_date(ed_data$ed_end_time, unit = 'hour')
ed_data <- ed_data %>%
  mutate(ed_start_time = floor_arrive, ed_end_time = floor_depart)

# set counter 1 for each arrival timestamp
arrive <- ed_data %>%
  select(time_mark = ed_start_time ) %>%
  mutate(counter = 1)
# set counter -1 for each departure timestamp
depart <- ed_data %>%
  select(time_mark = ed_end_time) %>%
  mutate(counter = -1)

# census volumes by hour by date
volumes <- arrive %>%
bind_rows(depart) %>%  # combine arrive/depart timestamps and counters (+1 or -1)
  arrange(time_mark, counter) %>%  # sort by ascending timestamp, and counters
  slice(-1) %>% # remove 1st row (year 2018 record - data quality issue)
  mutate(volume = cumsum(counter)) # cumulative sum each counter to get patient volume at given time

# sequence of consecutive dates by hour
min_time <- min(volumes$time_mark)
max_time <- max(volumes$time_mark)
seq_time_mark <- tibble(time_mark = seq(min_time, max_time, by = 'hour'))

# carry down census volume for timestamps with no arrival or departure
volumes <- volumes %>%
  right_join(seq_time_mark,
             by = 'time_mark') %>%
  arrange(time_mark) %>%
  fill(volume, .direction = 'down')

volumes
```

```
## # A tibble: 158,684 x 3
##    time_mark           counter volume
##    <dttm>                <dbl>  <dbl>
##  1 2019-01-01 06:00:00      -1     -1
##  2 2019-01-01 06:00:00       1      0
##  3 2019-01-01 06:00:00       1      1
##  4 2019-01-01 06:00:00       1      2
##  5 2019-01-01 06:00:00       1      3
##  6 2019-01-01 06:00:00       1      4
##  7 2019-01-01 06:00:00       1      5
##  8 2019-01-01 06:00:00       1      6
##  9 2019-01-01 06:00:00       1      7
## 10 2019-01-01 06:00:00       1      8
## # ... with 158,674 more rows
```

```r
# volumes %>% knitr::kable()
```

To aggregate the data further, we sum up the `counter` column for each hour and for each date to get the hourly change in volume for that date in that hourly timeframe. Then, we find the cumulative sum of the

hourly change in volume, which gives the ED volume at that hour on a specific date. To find the average volume by hour, we would aggregate by hour and calculate the mean for each hour. This is shown below. *Note: since we do not have records prior to January 2019, we would assume an initial ED volume of 0 here.*

```
counter_sum <- volumes$counter %>%
               aggregate(by = list(Time_mark = volumes$time_mark), FUN = sum) %>% # sum counters
               rename(hourly_change = x) %>% # rename column
               mutate(hourly_volume = cumsum(hourly_change)) # cumulative sum

counter_sum <- counter_sum %>%
             mutate(Hour = format(ymd_hms(counter_sum$Time_mark), '%H:%M:%S' )) # extract hour


# calculate average hourly volume
hourly_volume <- counter_sum$hourly_volume %>%
                 aggregate( by = list(Hour = counter_sum$Hour), FUN = mean, na.rm = TRUE ) %>% # avg. vo
                 rename(average_volume = x) %>% # rename
                 mutate(average_volume = floor(average_volume)) %>% # round down to nearest integer
                 arrange(desc(average_volume)) %>% # sort volume by descending order %>%
                 knitr::kable()
hourly_volume
```

| Hour | average_volume |
|---|---|
| 20:00:00 | 74 |
| 21:00:00 | 74 |
| 22:00:00 | 74 |
| 23:00:00 | 72 |
| 19:00:00 | 70 |
| 00:00:00 | 69 |
| 01:00:00 | 68 |
| 02:00:00 | 67 |
| 18:00:00 | 66 |
| 03:00:00 | 64 |
| 17:00:00 | 62 |
| 04:00:00 | 61 |
| 05:00:00 | 57 |
| 16:00:00 | 55 |
| 06:00:00 | 51 |
| 07:00:00 | 48 |
| 15:00:00 | 47 |
| 08:00:00 | 45 |
| 09:00:00 | 44 |
| 10:00:00 | 43 |
| 14:00:00 | 42 |
| 11:00:00 | 41 |
| 12:00:00 | 40 |
| 13:00:00 | 40 |

The data frame above shows average hourly census in the ED from January 2019 to January 2020. It is important to note that the `Hour` column actually indicates *hourly* timeframes. For example, the `20:00:00` Hour denotes the time frame from 8pm to 9pm, rather than exactly at 8pm. **From the output we see the highest average census occurs from 8pm to 11pm, with there being an average of 74 patients in the ED per hour.**