Deliverables Part 1 - Code

Linus Jen

8/5/2020

Note: This portion of the code will cover the deliverables found in the Word document. Specifically, this will cover the following concepts:

Do weekend games, on average, have higher attendance than weekday games?

Identify + rank the top 4 opponents with the highest average number of attendances.

Identify + rank the top 10 sections that are, on average, the most filled to their capacity.

Is there a correlation between opponents having higher Vegas Odds Score and higher attendance?

On average, which 15-minute period before/after the start of game has the highest number of people scanning in?

The questions above can be, for the most part, answered with only 1 dataset - the ticket_scan_data dataset. Thus, while I did include the other datasets, they may not be used often, if at all.

Packages

```
library(readxl) # reading excel files
library(dplyr) # basic tidyr functions
library(ggplot2) # Graphing the data, as needed
library(stringr) # Simple string manipulation
library(lubridate) # Working with dates
library(ggthemes) # To give the graphs some pop
library(corrplot) # A nice way to visualize correlations
library(car) # Used for powerTransform
```

Data

```
team_data = read_excel('team_data.xlsx')
game_data = read_excel('game_data.xlsx')
seating_chart = read_excel('seating_chart.xlsx')
ticket_price_data = read_excel('ticket_price_data.xlsx')
ticket_scan_data = read_excel('ticket_scan_data.xlsx')

# Looking through our data, though this was already done in the data cleanup file
#glimpse(team_data)
#glimpse(game_data)
#glimpse(seating_chart)
#glimpse(ticket_price_data)
#glimpse(ticket_scan_data)
```

We need to fix the dates, which are stored as UTC times instead of Pacific time

```
# Fix the GameDate column of game_data
game_data$GameDate = with_tz(game_data$GameDate, tz = "US/Pacific")

# Fix the event_date and scan_datetime columns of ticket_scan_data
ticket_scan_data$event_datetime = with_tz(ticket_scan_data$event_datetime,
                                          tz = "US/Pacific")
ticket_scan_data$scan_datetime = with_tz(ticket_scan_data$scan_datetime,
                                         tz = "US/Pacific")
```

Weekends vs. Weekdays

We must first determine, from our data, which games are on the weekend, and which games are during the weekdays

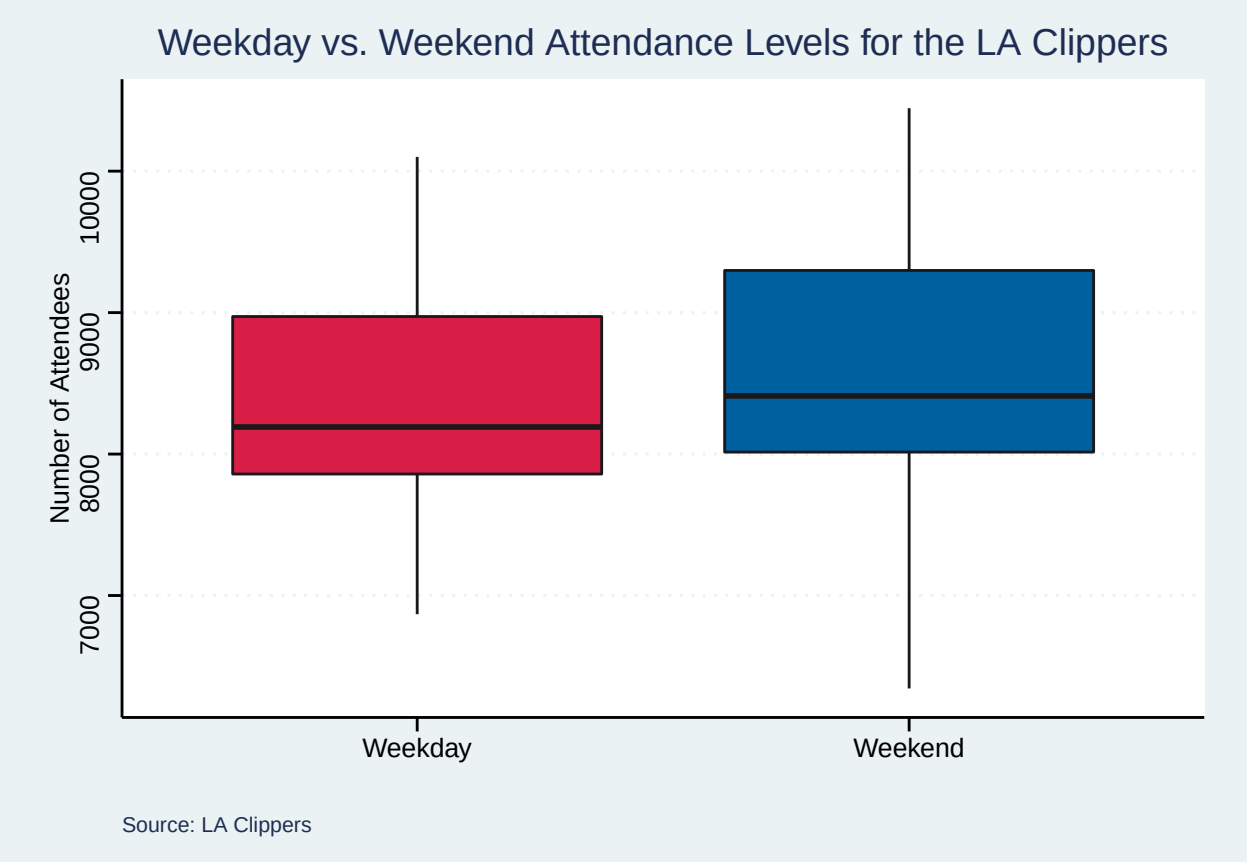
```
# Create a new column in our data frame that tells us the day of the week,
# and another column telling us if it is a weekday or not
wkday_comparison = ticket_scan_data %>%
   mutate(day_of_week = as.character(wday(event_datetime)),
          weekday = ifelse(day_of_week <= 4, "Weekday", "Weekend"))

# Create a new summary table that shows the number of attendees per game
attendees_per_game = wkday_comparison %>%
  group_by(event_name, weekday) %>%
  summarise(attendees = n())

# From the summary table above, now group by weekday to see if there is
# an obvious difference is average attendees per game
(attendees_avg_wkday = attendees_per_game %>%
    group_by(weekday) %>%
    summarise(average_attendees = mean(attendees)))
```

```
## # A tibble: 2 x 2
##   weekday average_attendees
##   <chr>               <dbl>
## 1 Weekday             8323.
## 2 Weekend             8606.
```

```
# Notice from the summary table above, there seems to be a small difference
# of 300 in average number of attendees for weekday and weekend games

# Let's run a t-test to see if this is significant, though it most likely is not
# The first argument gives the number of attendees when it is a weekday game
# The second argument gives the number of attendees when it is a weekend game
# The last argument asks if we are looking for a 1 or 2 sided t-test.
# Note that because we are checking if weekend games have a higher average
# attendance level, it will be a one-sided t-test

t.test(attendees_per_game$attendees[attendees_per_game$weekday == "Weekday"],
       attendees_per_game$attendees[attendees_per_game$weekday == "Weekend"],
       alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  attendees_per_game$attendees[attendees_per_game$weekday == "Weekday"] and attendees_per_game$a
## t = -0.83173, df = 24.461, p-value = 0.2068
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##      -Inf 298.7471
## sample estimates:
## mean of x mean of y
##  8322.615  8605.667
```

```r
# From the t-test above, while weekday games seem to have a lower number of attendees,
# the t-test tells us that this difference is NOT significant,
# and thus no conclusion of differences can be made regarding it

# Lastly, let's create a boxplot that compares weekday and weekend attendance levels
ggplot(attendees_per_game, aes(x = weekday, y = attendees, fill = weekday)) +
  geom_boxplot(color = "#1A1919") +
  labs(
    x = "",
    y = "Number of Attendees",
    title = "Weekday vs. Weekend Attendance Levels for the LA Clippers",
    caption = "Source: LA Clippers") +
  theme_stata() +
  theme(
    legend.position = 'none',
    panel.grid.major.y = element_line(linetype = "dotted")) +
  scale_fill_manual(values = c("#D81D47", "#0061A1"))
```



Ranking Opponents by Average Number of Attendances

For this section, we'll be grouping and ranking opponents based on the average number of attendees. Keep in mind that the number of times the Clippers play each team is not the same, as you do play teams within your division more than those of other conferences.

Note that in our data cleanup, we created a new column that gave the team they would be playing against. I had originally joined the ticket_scan_data with the game data, using the game_number as the joiner between the two. However, I found that the game_number from the event_name column of the ticket_scan_data actually referred to what number home game they were playing and NOT the actual game number for the year.

```
# First, create a summary table of the attendances per game
attendees_by_game = ticket_scan_data %>%
  group_by(game_number, Opponent, event_datetime) %>%
  summarise(attendees = n()) %>%
  ungroup()

# Create a summary table to find the average number of fans in attendance per game
# Using attendees_by_game above, which includes the Opponent and attendees
avg_attendees_by_team = attendees_by_game %>%
  group_by(Opponent) %>%
  summarise(avg_attendance = mean(attendees)) %>%
  arrange(desc(avg_attendance))
```

```r
# Show the entire data frame
print(avg_attendees_by_team, n = nrow(avg_attendees_by_team))
```

```
## # A tibble: 29 x 2
##    Opponent      avg_attendance
##    <chr>                  <dbl>
##  1 Boston                10101
##  2 Golden State           9798.
##  3 Los Angeles            9477
##  4 New York               9432
##  5 Sacramento             9360
##  6 San Antonio            9324
##  7 Milwaukee              9159
##  8 Dallas                 9034.
##  9 Philadelphia           8904
## 10 Toronto                8885
## 11 Brooklyn               8609
## 12 Chicago                8411
## 13 Miami                  8408
## 14 Orlando                8291
## 15 Phoenix                8200.
## 16 Houston                8110.
## 17 Cleveland              8032
## 18 Denver                 8028.
## 19 Washington             8019
## 20 Detroit                7996
## 21 Minnesota              7985
## 22 Utah                   7932.
## 23 Atlanta                7924
## 24 Oklahoma City          7890
## 25 Portland               7550.
## 26 Memphis                7438.
## 27 New Orleans            7156
## 28 Indiana                7012
## 29 Charlotte              6873
```

```r
# Look at the top 4 teams
head(avg_attendees_by_team, 4)
```

```
## # A tibble: 4 x 2
##   Opponent     avg_attendance
##   <chr>                 <dbl>
## 1 Boston               10101
## 2 Golden State          9798.
## 3 Los Angeles           9477
## 4 New York              9432
```

From our summary table above, we see that the top four teams with the highest avergae number of fans attending were the Boston Celics, Golden State Warriors, Los Angeles Lakers, and New York Knicks. These teams all have the longest history in the NBA, and are some of the largest organizations in the NBA, located in big, metropolitan hubs.

Hot Seat (or Hot Sections)

Now, we'll be looking at the top 10 sections that are most filled. To do this, we will be using the seating_chart and ticket_scan_data data sets, joining them together, and then grouping by section_name. To find most filled, we will divide them by the total number of seats per each group, and then find the average per game. Notice that rows don't matter, and court info is not given here, so we can simply not include those seat totals to our joined data. It is important to also note that not only were court seats not included in our ticket_scan_data, but premium seats, specifically 101 and 102 (and potentially others) were not included in our ticket_scans_data data set.

```
# First, we group by each section and game number,
# and find the total number of fans per each section
attendance_by_section = ticket_scan_data %>%
  group_by(game_number, section_name) %>%
  summarize(fans_per_section = n())

# Change the type of section_name to be character
attendance_by_section$section_name = as.character(attendance_by_section$section_name)

# Summarise our seating_chart so that we only have 2 columns:
# one with each section, and another column with the total seats per sections
# We will be keeping the court (CT) seats and regular seats separately
seating_chart_summarised = seating_chart %>%
  group_by(section_name) %>%
  summarise(seat_total = sum(seat_count))

# Now, we join this with our seating_chart data based off section_name so that
# we include the total seat_count, and include the percentage of seats filled per section
joined_seating_by_section = attendance_by_section %>%
  left_join(seating_chart_summarised, by = "section_name") %>%
  mutate(fill_percent = fans_per_section / seat_total)

# This next section was done to ensure that we don't have any errors,
# where maybe an entire section is filled
# (including court seats, which were not included when we joined the data together)
which(joined_seating_by_section$fans_per_section > joined_seating_by_section$seat_total)
```

```
## integer(0)
```

```
# Now, we create a column of percentage filled per game,
# summarise the average fill, sort the table, and determine
# which 10 sections had the highest average number of people attend the game
sorted_fill_percent_by_section = joined_seating_by_section %>%
  group_by(section_name) %>%
  summarise(avg_fill_percent = mean(fill_percent)) %>%
  arrange(desc(avg_fill_percent))

# Now, display the head
head(sorted_fill_percent_by_section, 10)
```

```
## # A tibble: 10 x 2
##    section_name avg_fill_percent
##    <chr>                   <dbl>
```

```
##  1 207                    0.794
##  2 309                    0.790
##  3 106                    0.786
##  4 118                    0.782
##  5 215                    0.774
##  6 116                    0.773
##  7 206                    0.766
##  8 105                    0.764
##  9 310                    0.758
## 10 217                    0.753
```

Notice that the seats right behind the opponents backboard seems to garner the highest fill percentage, as seen with sections 207, 309, and 106 making the top 3 filled sections. And in general, the sections behind the backboard are quite highly filled, as 9 of the 10 sections (all but 118) on the list are somewhere behind the backboard.

From my work with the UCLA Athletics marketing department, I know that these seats tend to be the least popularity, as the backboards obstruct the view for fans. However, the Clippers do a great job selling off these seats to fans.

Correlation Between Vegas Odds and Attendance

Here, we'll be looking into if there is a correlation between Vegas Odds at winning the championship, and attendance per game. To do this, I will join a prior dataset, attendees_by_game (which gives the game number, opponent, and number of attendees) with the teams_data dataset, which includes the Vegas Odds values. I will be transforming the data to see if there could potentially be a hidden function behind this using a power transformation, and this will be noted in the report.

```
# Use the attendees_by_game dataset here, as it already lists the Opponent
# and number of attendees per game.
# We will join this with the team_data to get the Vegas Odds for the year
# First, we must turn them into character strings to join
attendees_by_game$join_date = as.character(attendees_by_game$event_datetime)
game_data$join_date = as.character(game_data$GameDate)

# Now, join the data together by date, and select the Opponent.y column, attendees,
# and Vegas Odds for 1819
avg_attendees_by_team = attendees_by_game %>%
  left_join(game_data, by = "join_date") %>%
  select(Opponent.y, attendees)

# Rename the columns for better organization
names(avg_attendees_by_team)[1] = "Opponent"

# Join the previous dataset with the team_data dataset, which includes Vegas Odds
(attendance_odds_per_game = avg_attendees_by_team %>%
  left_join(team_data, by = c("Opponent" = "Team Full Name")) %>%
  select(Opponent, attendees, `Vegas Odds for 1819`))
```

```
## # A tibble: 41 x 3
##    Opponent               attendees `Vegas Odds for 1819`
##    <chr>                      <int>                 <dbl>
##  1 Denver Nuggets              8173                  47.5
```

```
##  2 Oklahoma City Thunder       6343                 50.5
##  3 Houston Rockets             8009                 54.5
##  4 Washington Wizards          8019                 44.5
##  5 Minnesota Timberwolves      7985                 44.5
##  6 Milwaukee Bucks             9159                 46.5
##  7 Golden State Warriors       9150                 62.5
##  8 San Antonio Spurs           8559                 43.5
##  9 Memphis Grizzlies           7019                 34.5
## 10 Phoenix Suns                7648                 28.5
## # ... with 31 more rows
```

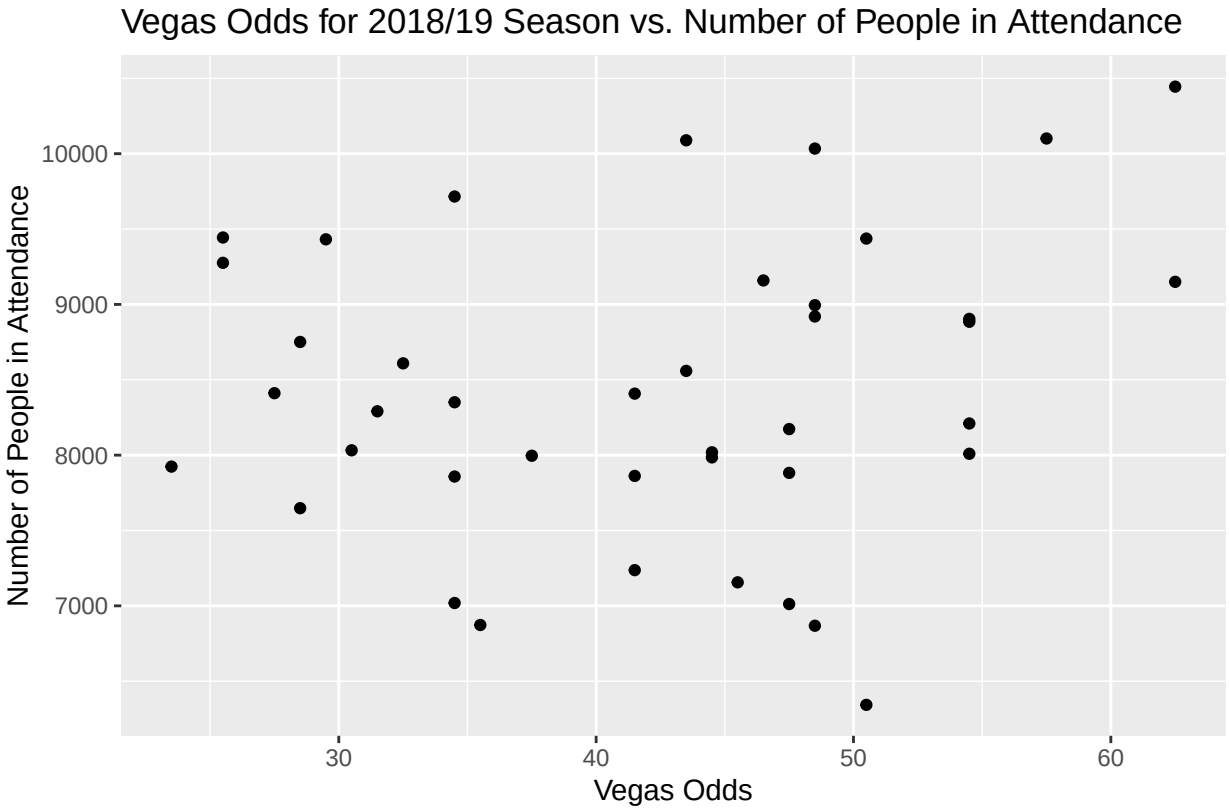
```r
# Now, find the correlation between Vegas Odds and number of attendees
corrmat = cor(attendance_odds_per_game$`Vegas Odds for 1819`,
              attendance_odds_per_game$attendees)
corrmat
```

```
## [1] 0.1606018
```

From the output above, we can see that there is a weak correlation between the Vegas Odds for winning a championship, getting an R value of .16. This was surprising to me, as I would've thought that people would want to watch the Clippers play the best teams in the league.

However, there is a chance that applying some transformations to the data may improve the correlation. We will be using the powerTransform() function to turn the variables to be more normal, and the invResPlot() to make the distribution of one variable closer to that of the dependent variable.

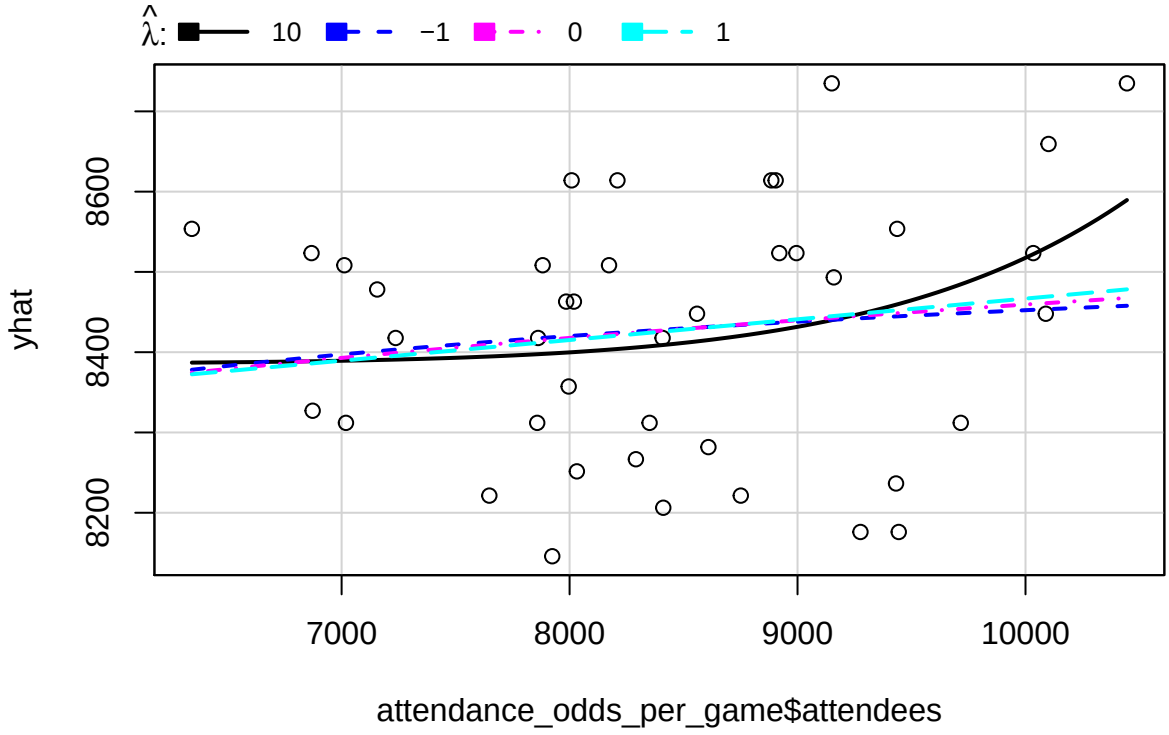
```r
# Simple plot to check this
ggplot(attendance_odds_per_game, aes(x = `Vegas Odds for 1819` , y = attendees)) +
  geom_point() +
  labs(
    x = "Vegas Odds",
    y = "Number of People in Attendance",
    title = "Vegas Odds for 2018/19 Season vs. Number of People in Attendance",
    caption = "Source: LA Clippers")
```




```
# Let's see if a transformation can improve the correlation

# Method 1: powerTranformation to normalize the data
summary(powerTransform(cbind(attendance_odds_per_game$attendees, attendance_odds_per_game$`Vegas Odds fo
```

```
## bcPower Transformations to Multinormality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.9923           1      -1.4526       3.4372
## Y2    0.9465           1      -0.3261       2.2192
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                              LRT df    pval
## LR test, lambda = (0 0) 2.691752  2 0.26031
##
## Likelihood ratio test that no transformations are needed
##                                 LRT df    pval
## LR test, lambda = (1 1) 0.006789848  2 0.99661
```

```
# The output above returns lambda values of 1, meaning that it does not recommend any changes to make th

# Lastly, we'll use an inverse response plot to try and make attendees have
# a similar distribution as the Vegas Odds, and vice versa
invResPlot(lm(attendance_odds_per_game$attendees ~
                attendance_odds_per_game$`Vegas Odds for 1819`))
```



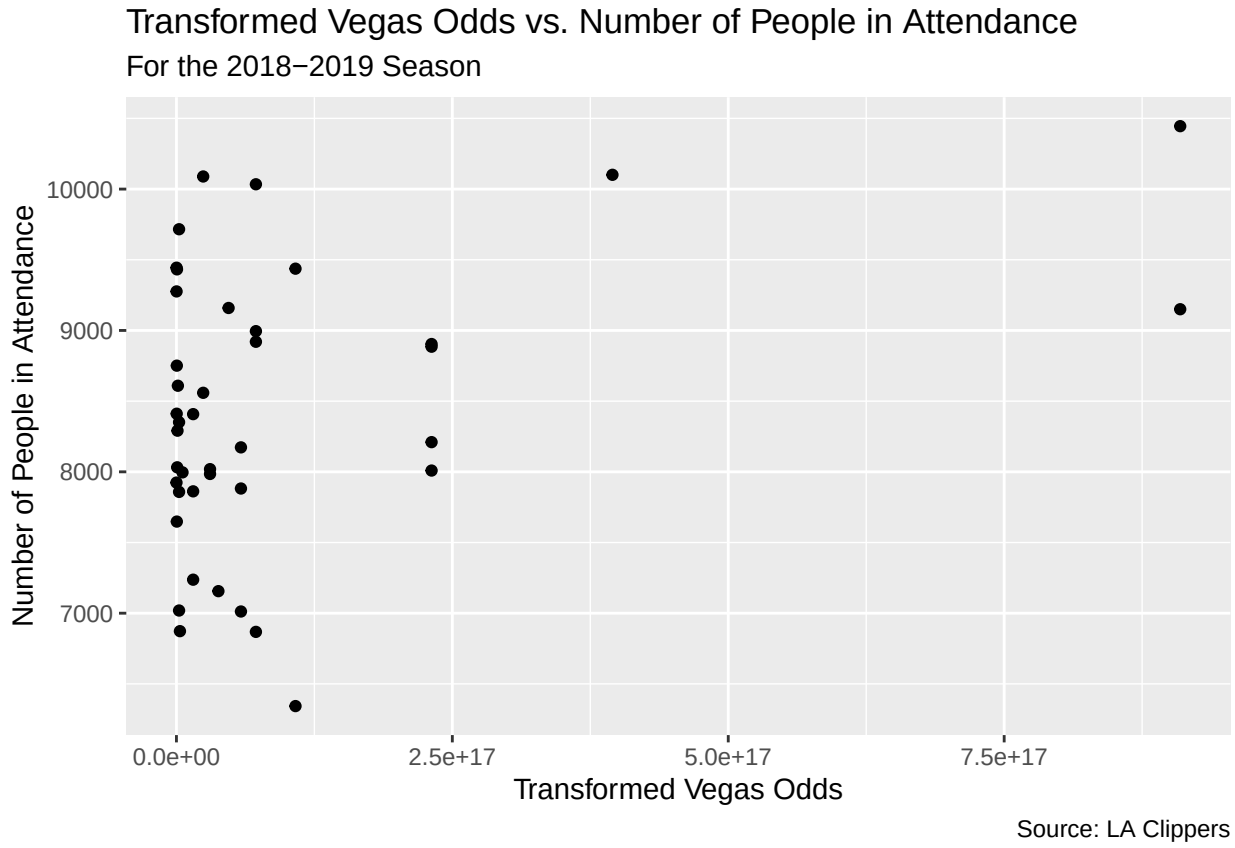

```
##      lambda      RSS
## 1  9.999926 918128.9
## 2 -1.000000 989569.9
## 3  0.000000 983913.3
## 4  1.000000 977527.4
```

```
# Apply a lambda of 10 to the Vegas Odds to see its impact
cor(attendance_odds_per_game$`Vegas Odds for 1819`^10,
    attendance_odds_per_game$attendees)
```

```
## [1] 0.374807
```

```
# We see a huge increase in correlation after applying this change!

# Let's make a very quick graph to show this
# First, create a new column of the transformed data
attendance_odds_per_game$Vegas_Odds_trans = attendance_odds_per_game$`Vegas Odds for 1819`^10

# Now, a quick and simple graph
ggplot(attendance_odds_per_game, aes(x = Vegas_Odds_trans , y = attendees)) +
  geom_point() +
  labs(
    x = "Transformed Vegas Odds",
    y = "Number of People in Attendance",
    title = "Transformed Vegas Odds vs. Number of People in Attendance",
```

```
    subtitle = "For the 2018-2019 Season",
    caption = "Source: LA Clippers")
```



From our tests and studies above, we see that by using an inverse response plot, we can greatly improve our R value to .375, which sees over a .2 increase in the R value. A positive trend does seem to appear here.

Busiest Scan Period

Here, we'll be finding which 15 minute period has the most people scanning in to enter the STAPLES center.

```
# Create a new column called time_diff that finds the difference between
# the scan time and event start time
time_difference = ticket_scan_data %>%
  mutate(time_diff = as.numeric(difftime(scan_datetime, event_datetime,
                                         units = "mins")),
    # And create a new column called interval that determines which
    # 15 minute interval the time_diff falls in, and rounds the value down
                           interval = floor(time_diff/15))

# Now, we can group by the 15 minute interval and game,
# and count the number of observations
# Notice that positive values mean that they scanned in AFTER the game
# has started, and negative values mean that they arrived early
(interval_grouped_by_game = time_difference %>%
  group_by(interval, game_number) %>%
```

```
  summarise(count = n()) %>%
  ungroup() %>%
  arrange(game_number))
```

```
## # A tibble: 788 x 3
##    interval game_number count
##       <dbl>       <dbl> <int>
##  1      -13           1     1
##  2       -8           1    82
##  3       -7           1    92
##  4       -6           1   315
##  5       -5           1   265
##  6       -4           1   443
##  7       -3           1   827
##  8       -2           1  1156
##  9       -1           1  1566
## 10        0           1  1530
## # ... with 778 more rows
```

```
# Now, there are two ways of answering this question

# Part 1: Which period has the highest average NUMBER of people scanning in?
# To do this, we will need to now group it by interval, but find the average
# number of attendees for each section per game
(interval_grouped_avg = interval_grouped_by_game %>%
  group_by(interval) %>%
  summarise(average_scans = mean(count)) %>%
  ungroup() %>%
  arrange(desc(average_scans)))
```

```
## # A tibble: 28 x 2
##    interval average_scans
##       <dbl>         <dbl>
##  1       -1         1507.
##  2        0         1340.
##  3       -2         1253.
##  4       -3          909.
##  5        1          716.
##  6       -4          634.
##  7       -6          577.
##  8       -5          468.
##  9        2          338.
## 10       -8          188.
## # ... with 18 more rows
```

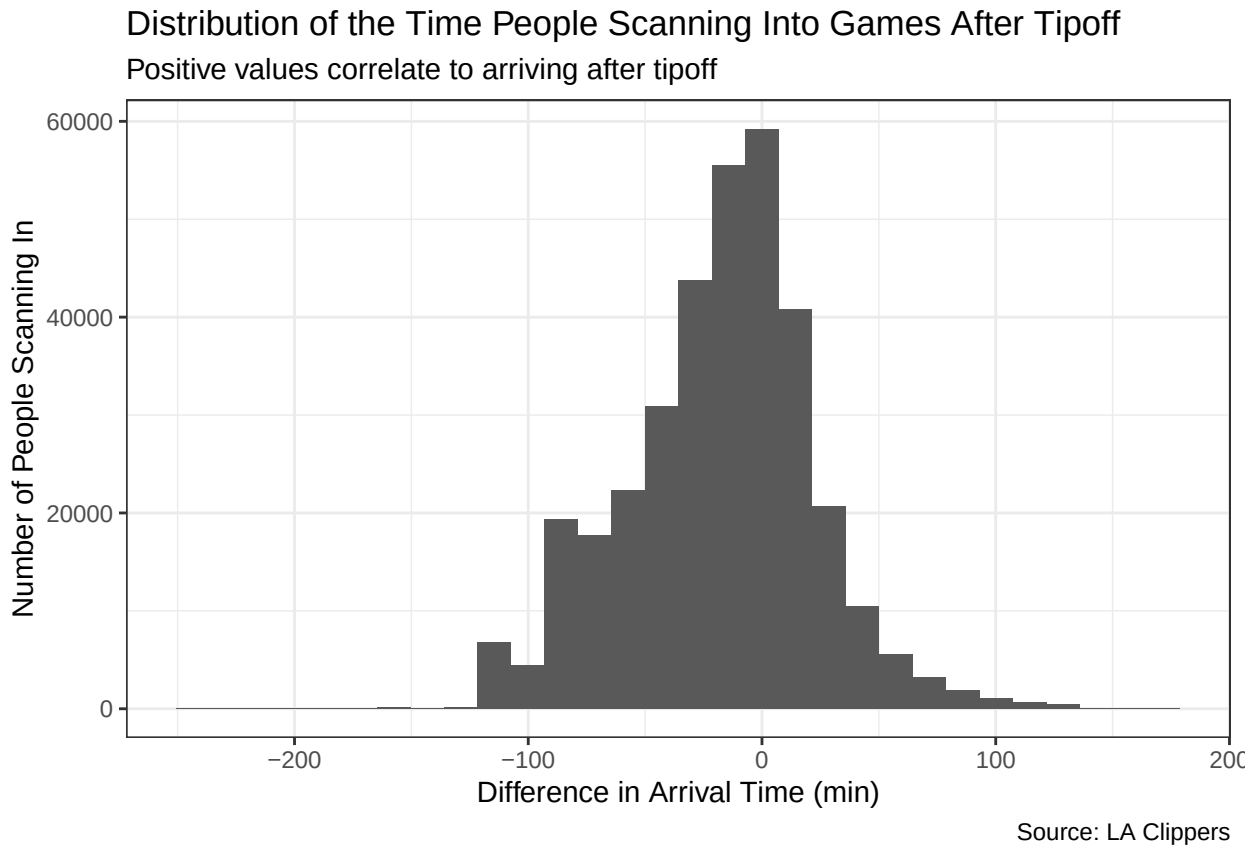
```
# From the summary table outputted above, we see that the -1 interval,
# representing 30 minutes prior to the game starting to 15 minutes before tipoff,
# has the most people, on average, scanning in, at ~1507 people

# Part 2: Which period has the highest average PERCENTAGE of people scanning in?
# To do this, we will first need to make a summary table of the total number
# of people scanning in per game
```

```r
total_scans_per_game = interval_grouped_by_game %>%
  group_by(game_number) %>%
  summarise(total_attendees = sum(count))

# Now, we join the total scans per game with our interval_grouped_by_game dataset
# So that we can create a new column of % of people entering per interval
interval_percent = interval_grouped_by_game %>%
  left_join(total_scans_per_game, by = "game_number") %>%
  mutate(percent_fill = count / total_attendees)

# Now, create a summary table to find the average amount of people scanning
# in at each interval
(interval_percent = interval_percent %>%
  group_by(interval) %>%
  summarise(avg_percent_scan = mean(percent_fill)) %>%
  arrange(desc(avg_percent_scan)))
```

```
## # A tibble: 28 x 2
##    interval avg_percent_scan
##       <dbl>            <dbl>
##  1       -1           0.180
##  2        0           0.160
##  3       -2           0.149
##  4       -3           0.107
##  5        1           0.0861
##  6       -4           0.0745
##  7       -6           0.0673
##  8       -5           0.0542
##  9        2           0.0408
## 10       -8           0.0219
## # ... with 18 more rows
```

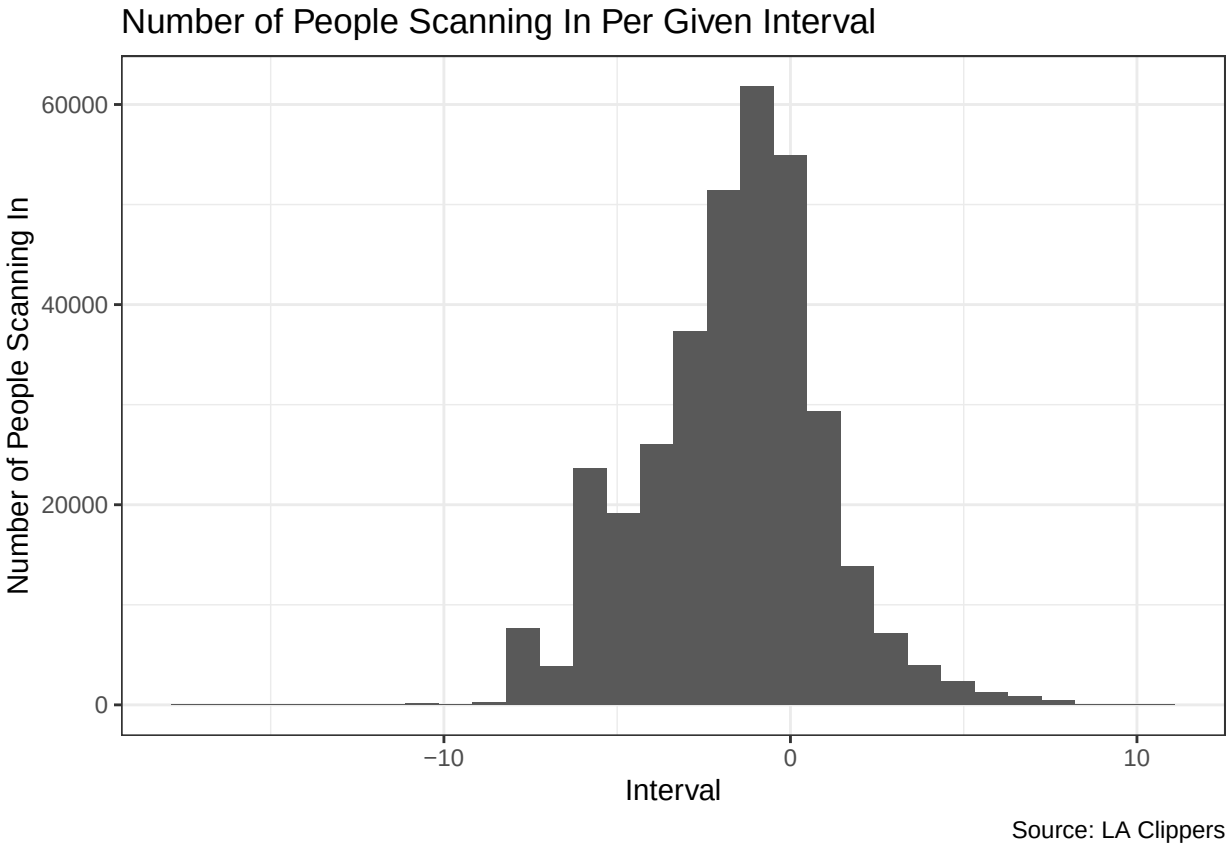
```r
# Again, we see that the -1 interval, representing 30 minutes to 15 minutes
# prior to tipoff, has the most people coming into the Staples Center
# These values match the table we found earlier


# Just a quick and dirty visualization to make sure that the data looks right
ggplot(time_difference) +
  geom_histogram(aes(x = time_diff)) +
  labs(
    x = "Difference in Arrival Time (min)",
    y = "Number of People Scanning In",
    title = "Distribution of the Time People Scanning Into Games After Tipoff",
    subtitle = "Positive values correlate to arriving after tipoff",
    caption = "Source: LA Clippers") +
  theme_bw()
```

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




```
ggplot(time_difference, aes(x = interval)) +
  geom_histogram() +
  labs(
    x = "Interval",
    y = "Number of People Scanning In",
    title = "Number of People Scanning In Per Given Interval",
    caption = "Source: LA Clippers") +
  theme_bw()
```

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



From the summary table and histograms, we see that the interval with the most scans is the interval from 30 minutes prior to game starting to 15 minutes prior to the game starting, as denoted here as interval -1. Intervals were created by dividing the difference between the scan time and the game time by 15, and then rounding the value. For example, a person scanning into the game at 7:20PM for a 7:30PM game would have difference of -10, and after dividing by 15 and rounding down, would get a value of 0, representing that they came during the interval between 15 minutes before the game and tipoff. On the other hand, if the person came at 7:20PM for a 7:00PM game, the difference in times would be 20, and when divided by 15 and rounded down, we would get a value of 1. This interval would represent that they came during the interval 15 minutes after the game had started to 30 minutes after tipoff. These are roughly supported in the crude histograms shown above, as the peak values are slightly before 0, thus showing a small, negative value for the interval.