STA 380 Part 2: Exercises

Aidan Cremins, Peyton Lewis, Joe Morris, Amrit Sandhu

2022-07-29

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(forcats)
library(reshape2)
library(knitr)
```

Probability Practice

Part a.

P(Y) = 0.65 P(N) = 0.35 P(RC) = 0.3 P(TC) = 0.7 (P(RC)-1) P(Y|RC) = 0.5 P(N|RC) = 0.5

These probabilities are summarized in the table below:

We're looking for P(Y|TC) so we can use the rule of total probability:

P(Y) = P(Y, TC) + P(Y, RC) = P(TC) * P(Y|TC) + P(RC) * P(Y|RC)

We know all of these inputs to the equation except for P(Y|TC), so we want to solve for that unknown.

0.65 = 0.7 * P(Y|TC) + 0.3 * 0.5

From the above equation, we find that P(Y|TC) $\approx$ 0.714286. This means that truthful clickers answer yes to the question about 71.43% of the time.

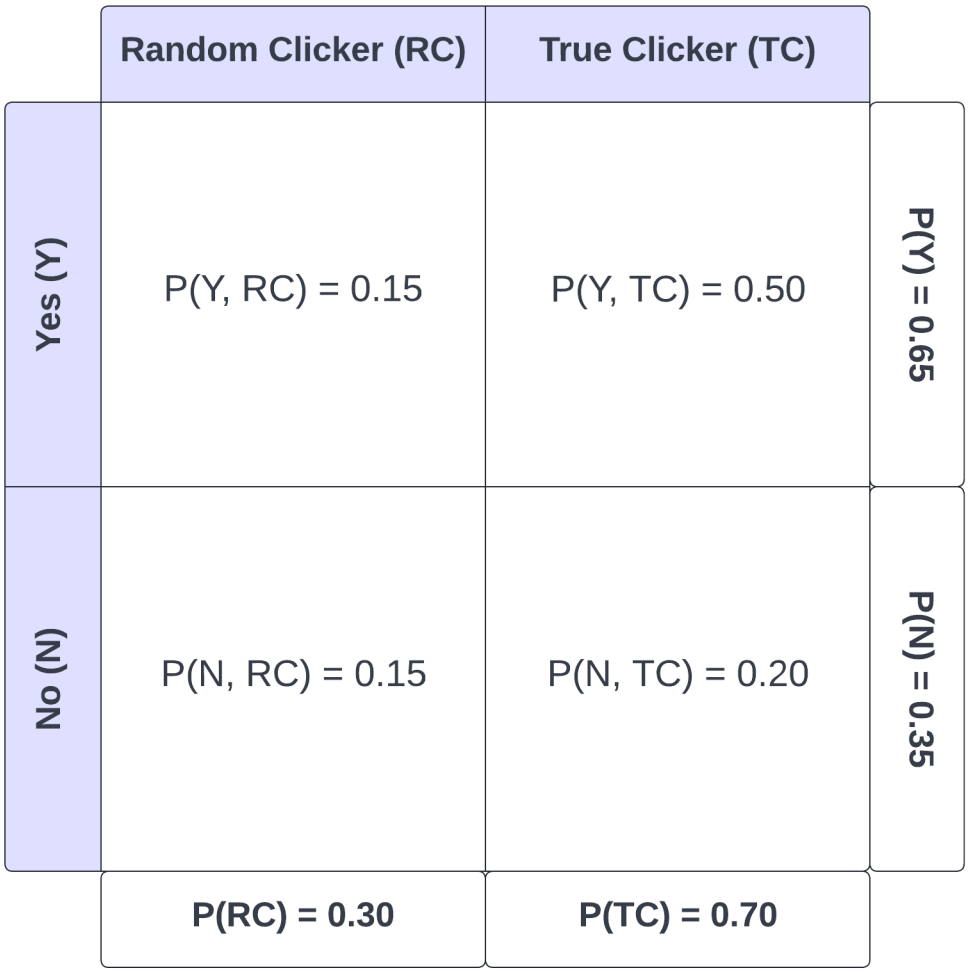

Random Clicker (RC)
True Clicker (TC)
Yes (Y)
P(Y, RC) = 0.15
P(Y, TC) = 0.50
P(Y) = 0.65
No (N)
P(N, RC) = 0.15
P(N, TC) = 0.20
P(N) = 0.35
P(RC) = 0.30
P(TC) = 0.70

Figure 1: alt

Part b.

P(Disease) = 0.000025 P(No Disease) = 0.999975 (1-0.000025) P(Positive|Disease) = .993 P(Negative|No Disease) = 0.9999

The probabilities above are summarized in the tree diagram below:

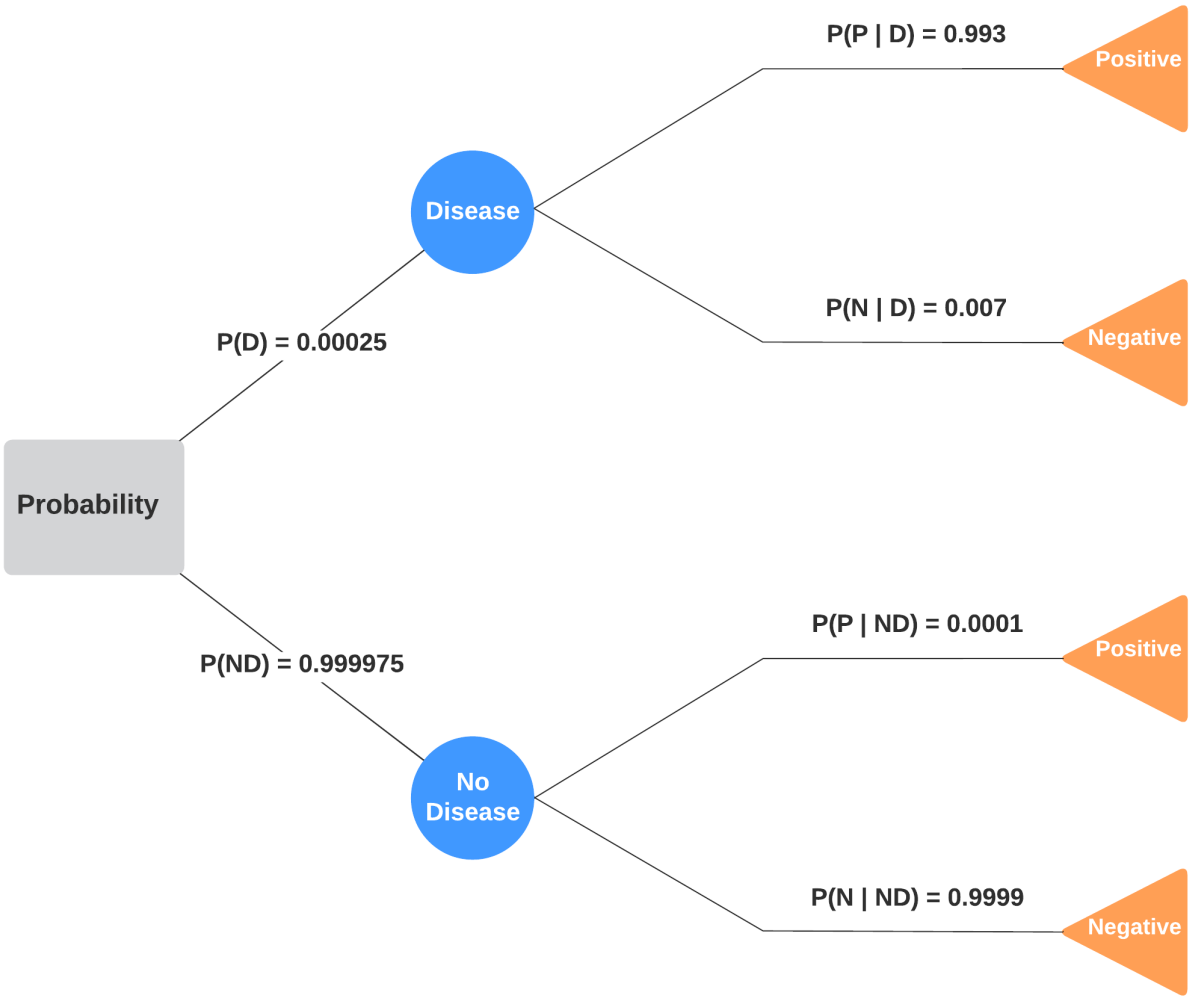


Figure 2: alt

We're looking for P(Disease|Positive) so we can use Baye's Law:

$$\frac{P(Disease)*P(Positive|Disease)}{P(Disease)*P(Positive|Disease)+P(NoDisease)*P(Positive|NoDisease)}$$

We have almost all of the inputs that we need, however, we're missing P(Positive|No Disease). These are false positives. We can find the missing probability by taking 1 - true negatives, or 1 - 0.9999 to get P(Positive|No Disease) as 0.0001. Now we can solve for P(Disease|Positive).

$\frac{0.000025*0.993}{0.000025*0.993+0.999975*0.0001} \approx .198882$. Thus, if someone tests positive, they have about a 19.89% chance of actually having the disease.

Wrangling the Billboard Top 100

```
billboard = read.csv("data/billboard.csv")
```

#Need a caption - probably something about how most are recent songs

Part a.

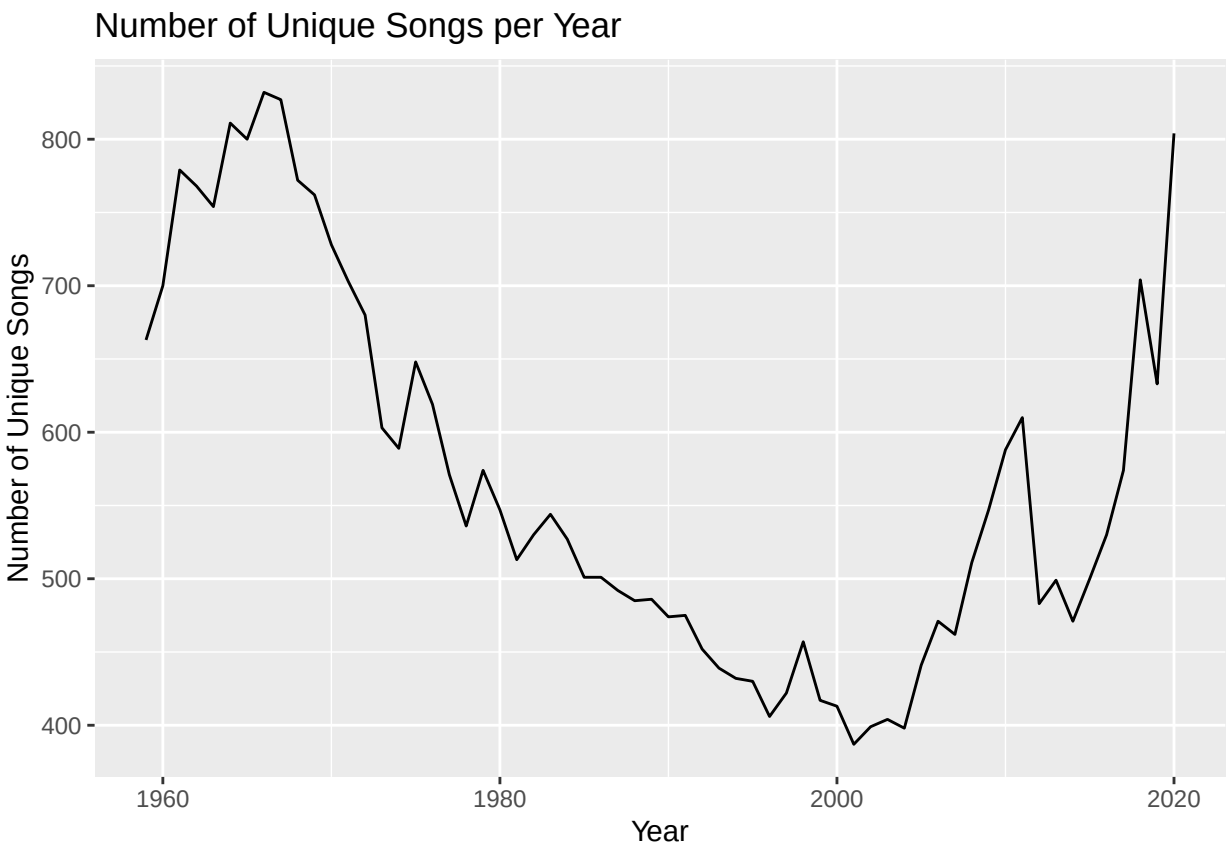
The table above shows the top 10 longest lasting songs on the Billboard 100. It reveals that a majority of these long-lasting songs are more recent songs.

Part b.

```
musical_diversity = billboard %>%
  filter(year != 1958 & year != 2021) %>%
  group_by(year) %>%
  summarize(song_performer = paste(performer, song),unique_songs_per_year=length(unique(song_performer))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
ggplot(musical_diversity) + geom_line(aes(x = year, y = unique_songs_per_year)) + xlab("Year") + ylab("
```



As seen in the plot, the number of unique songs steadily decreased from roughly 1965 to a overall minimum of just below 400 songs around 2000. Since then, the trend has been increasing and in the last year of data (2020), there were about 800 unique songs. It looks like it's on trend to surpass its historical max in the mid 1960s (about 830).

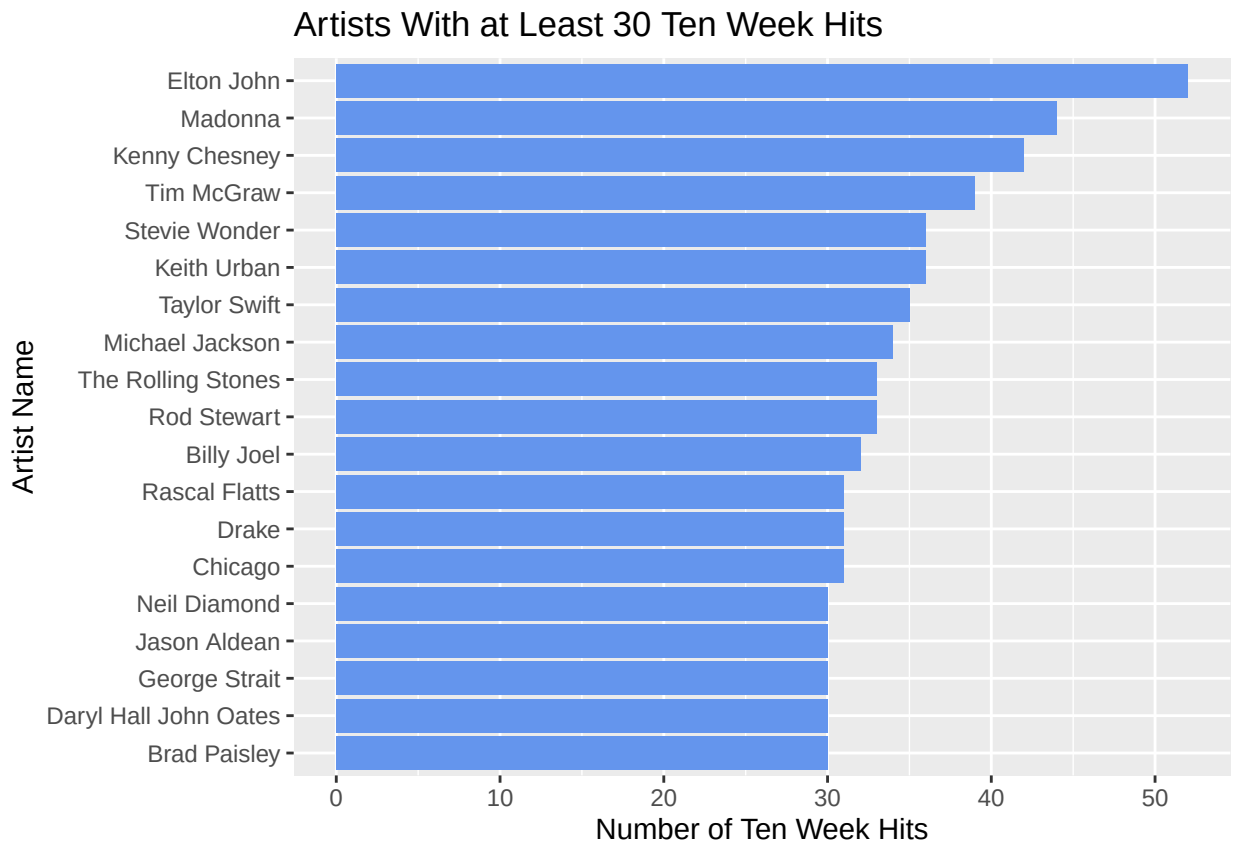
Part c.

```
ten_week_hit_songs <- billboard %>%
  group_by(performer,song) %>%
  summarize(ten_week_hit = ifelse(n()>=10,"Yes","No")) %>%
  filter(ten_week_hit == "Yes")
```

```
## `summarise()` has grouped output by 'performer'. You can override using the
## `.groups` argument.
```

```
top_artists <- ten_week_hit_songs %>%
  group_by(performer) %>%
  summarize(num_ten_week_hit = n()) %>%
  filter(num_ten_week_hit>=30)

ggplot(top_artists) + geom_bar(aes(x = fct_reorder(performer,num_ten_week_hit), y = num_ten_week_hit),s
```



Given the bar plot above, it shows us that Elton John had the most 10 week hits with just over 50. The other 18 artists in the plot seem to be a mix of generations and genres, but they all are relatively well known which makes sense given that they have so many hits.

#Visual story telling part 1: green buildings

To begin this problem, we can look at the distribution of rents for green buildings vs. non-green buildings and we see that indeed green buildings have a slightly higher median rent than non-green buildings. We're already skeptical of the stats guru, because while yes, there is a difference in rents, it's really not very significant.

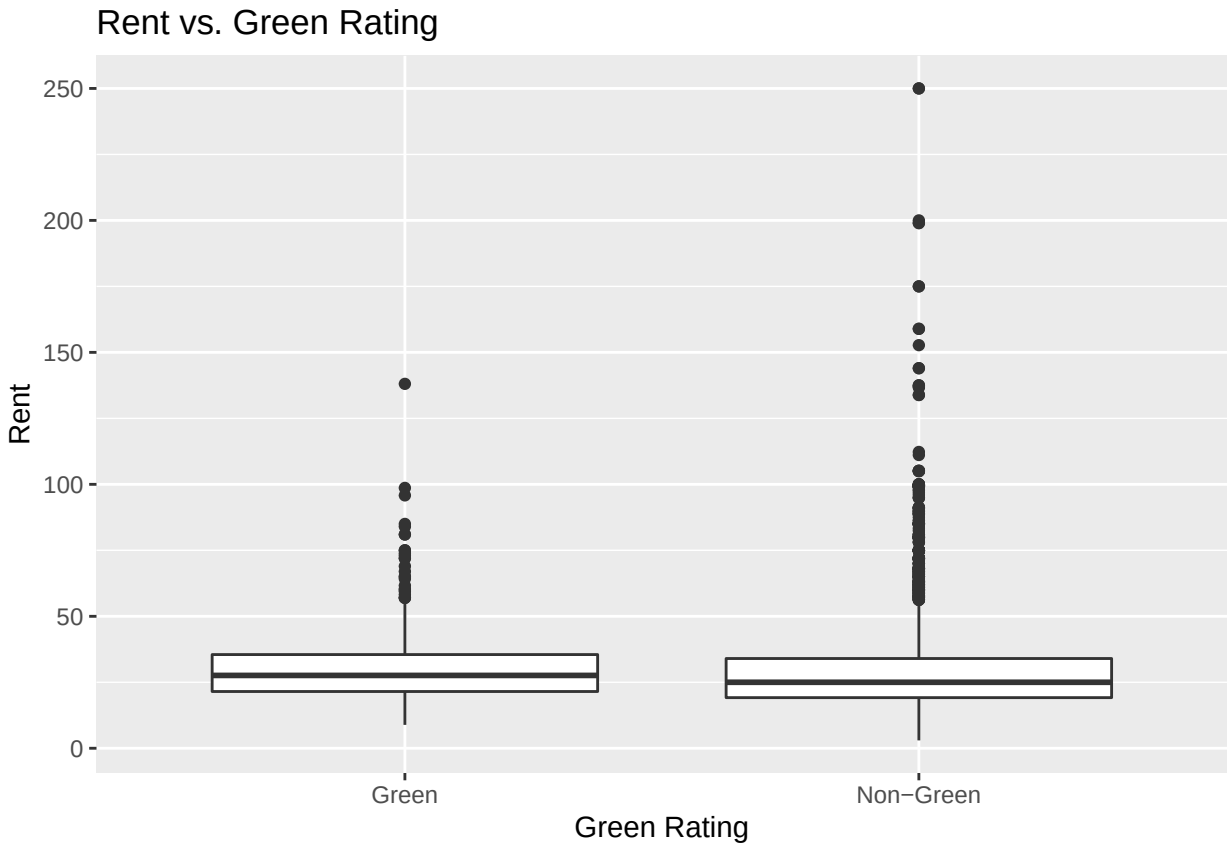


Figure 1

Even though the rent premium wasn't that large, if the leasing rate in green buildings was substantially higher, the larger volume of tenants to charge rent to might still justify undergoing the investment. However, the boxplot below suggests that there's not a huge difference in the median leasing rates between green and non-green buildings.

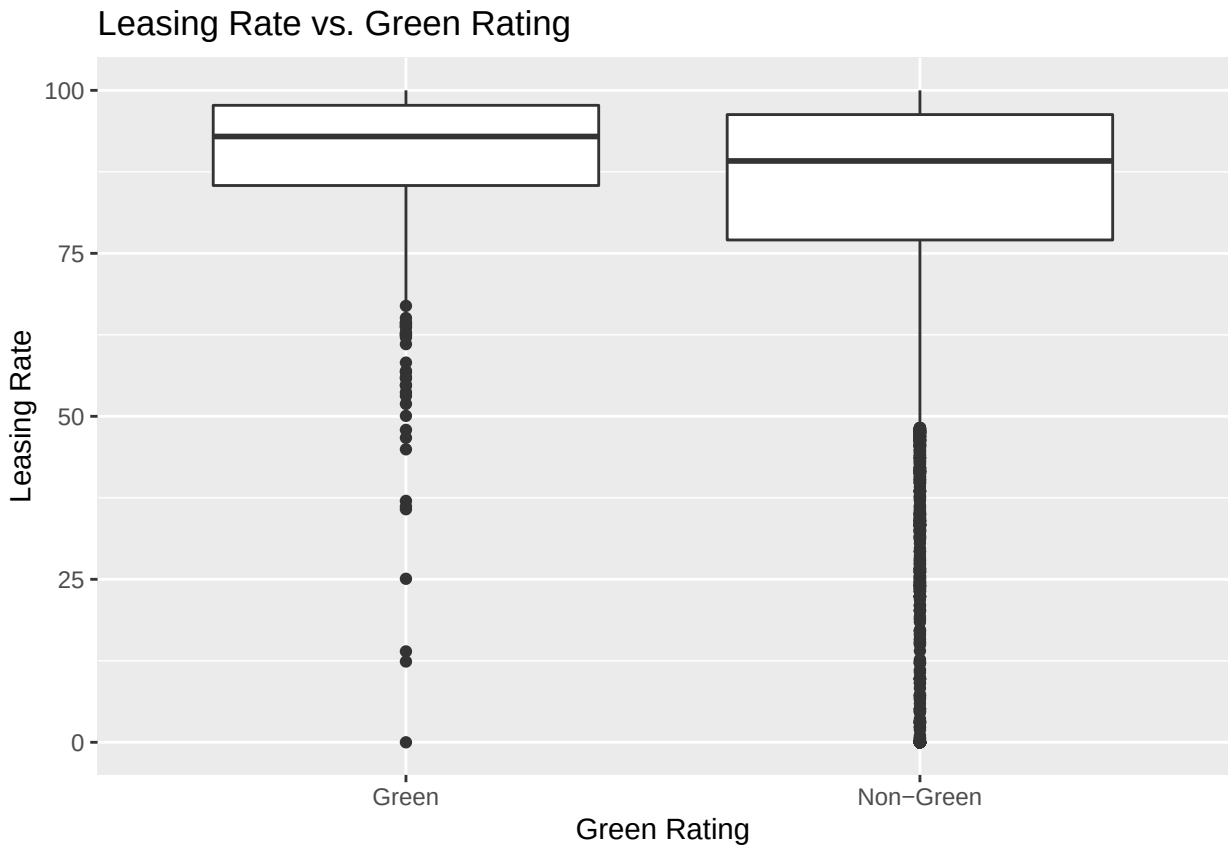
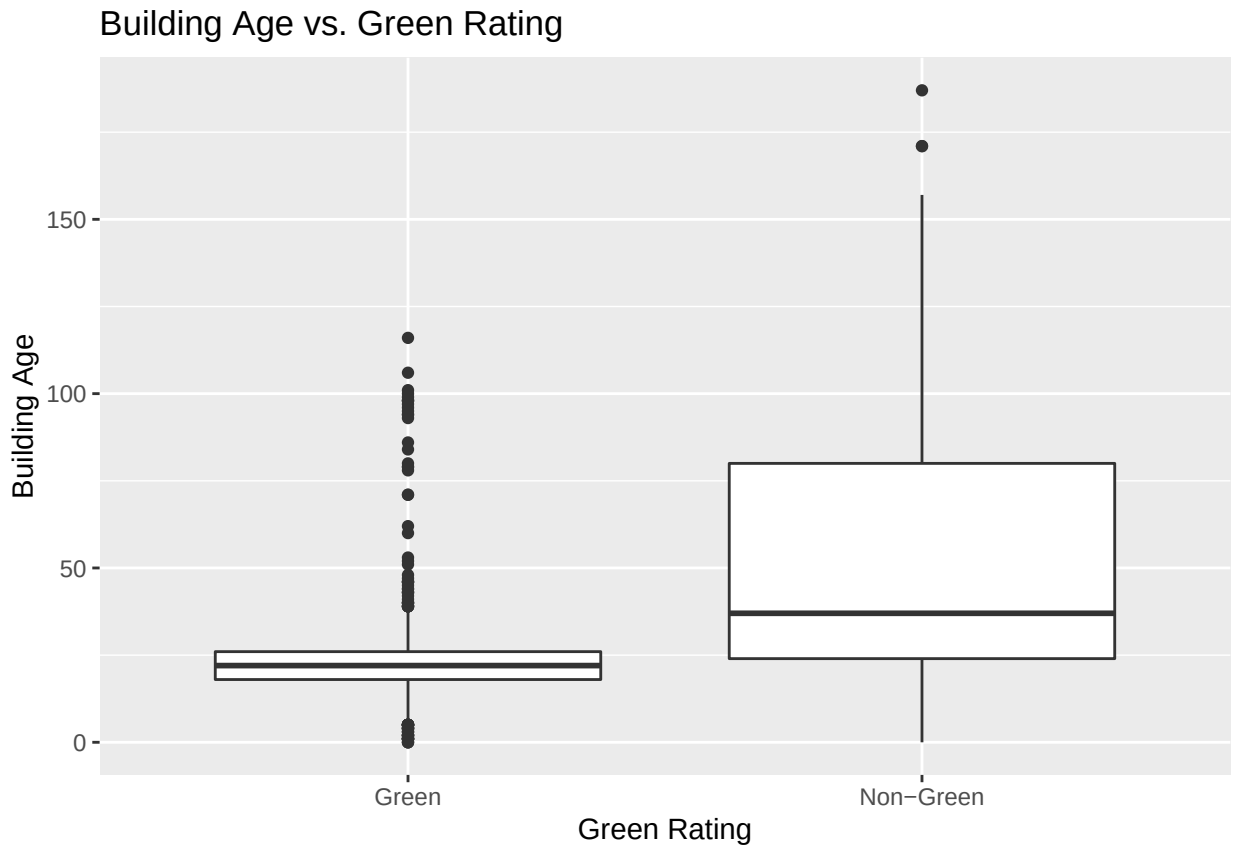


Figure 2

We also wanted to explore some confounding variables that may be influencing the relationship between rent and green rating. The plot below reveals that green buildings tend to be significantly newer compared to non-green buildings. This could be a problem because perhaps the premium of green buildings is just because they're newer buildings.

Figure 3

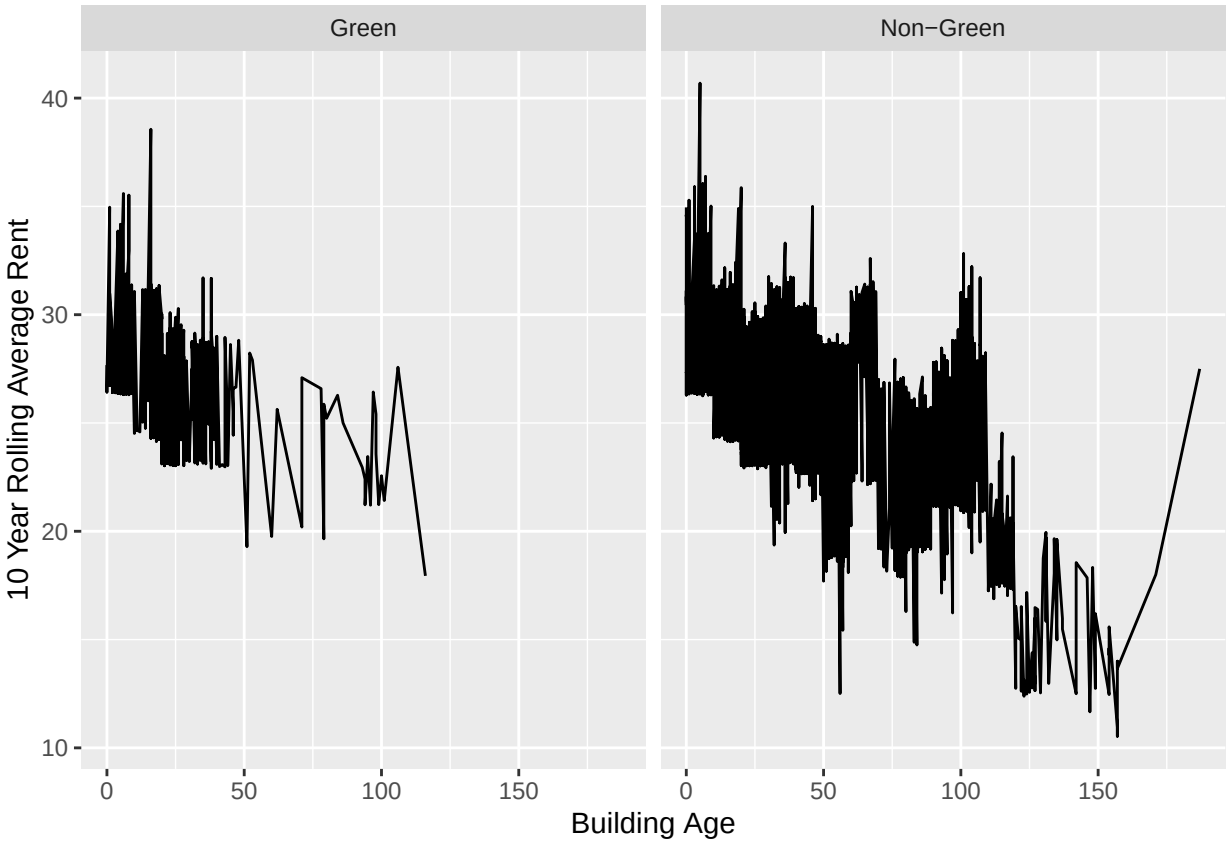
```
ggplot(green_buildings, aes(x = green_rating, y = age, group = green_rating)) + geom_boxplot() + xlab("
```



The plot below "adjusts" for age by plotting the 10 year rolling average rent vs. age and then faceted by green vs. non-green buildings. It shows that the average rents between green vs. non-green buildings doesn't appear to be significantly different in the range of 0-100 years of age. Interestingly, we found that there are some very old (100-150 years) non-green buildings that have very low 10 year rolling average rents whereas there are no green buildings in this age range. It seems as though the very old non-green buildings may be the reason for the median rent discrepancy shown in Figure 1.

Figure 4

```
#Get rolling average of rents for each decade
green_buildings$decade <- (green_buildings$age%/%10)*10
decade_data <- green_buildings %>%
  group_by(decade) %>%
  mutate(rec = 1) %>%
  mutate(rollavg = cumsum(Rent)/cumsum(rec)) %>%
  select(-rec)
labels = c("Green","Not Green")
ggplot(decade_data, aes(x = age, y = rollavg)) + geom_line() + facet_grid(. ~ green_rating) + xlab("Bui
```



Another confounding variable that we noticed was building class. There are many more buildings of class A (higher quality) than class B for green buildings and the opposite for non-green. Thus, the rent discrepancy in Figure 1 could just be because of quality differences.

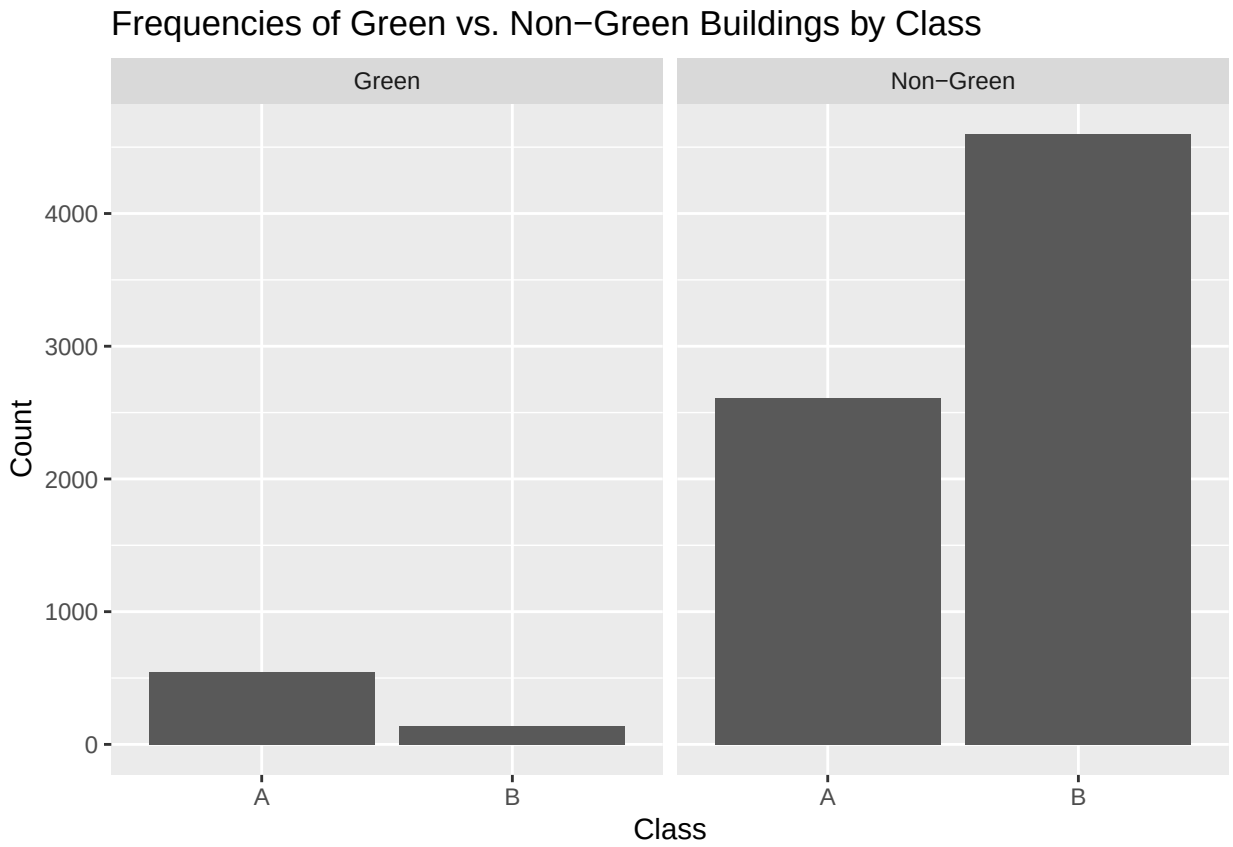


Figure 5

Figures 6 and 7 below "adjust" for class by having a pair of box plots showing rent vs. green rating for class A buildings and then another set of box plots for class B buildings. After controlling for building class, in both plots the median rent for green and non-green buildings were not significantly different.

Figure 6

```
classa_subset <- subset(green_buildings, green_buildings$class=="A")
classb_subset <- subset(green_buildings,green_buildings$class=="B")
ggplot(classa_subset, aes(x = green_rating, y = Rent ,group = green_rating)) + geom_boxplot() + xlab("G
```

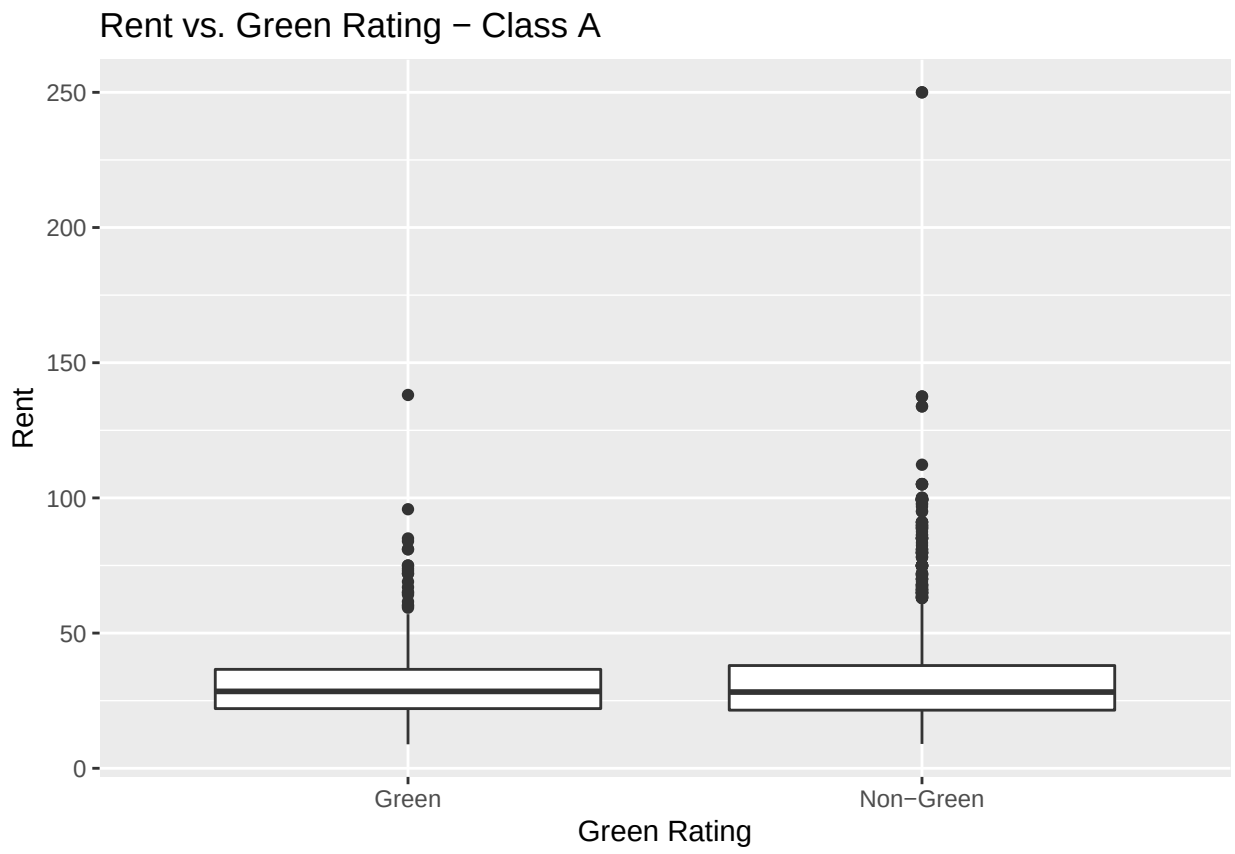
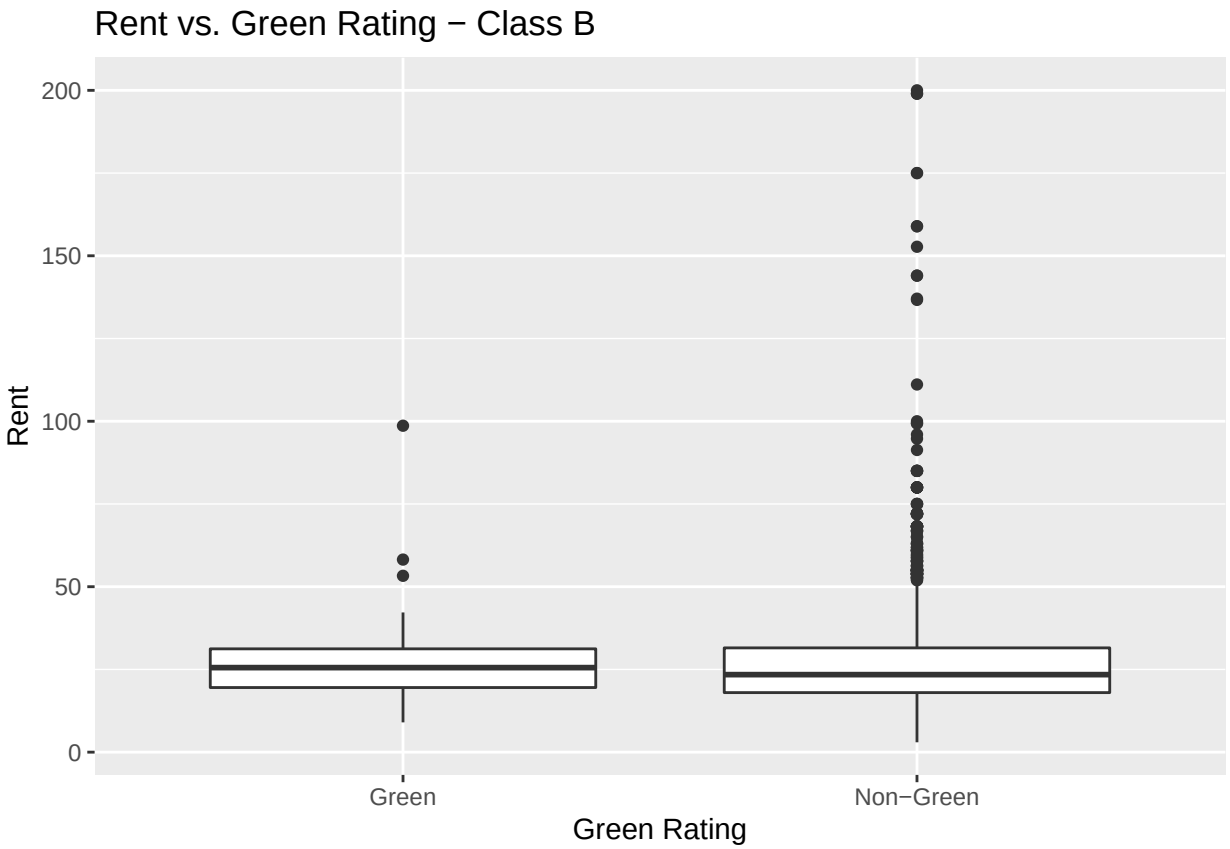


Figure 7

```
ggplot(classb_subset, aes(x = green_rating, y = Rent ,group = green_rating)) + geom_boxplot() + xlab("G
```



Overall, there doesn't seem to be a significant difference in median rents between green and non-green buildings which was true even after controlling for building age and building class. If we were to get a different sample of green vs. non-green buildings, it's possible that the median rent for green buildings could be lower than non-green buildings because the medians in our sample were so close. Ultimately, it doesn't seem to be wise to undergo additional costs to make the building green when there doesn't appear to be a significantly higher median rent that can be charged to offset the cost. The analyst failed to consider the sample variability, confounding variables, or the time value of money (i.e. $650,000 9 years from now isn't same as $650,000 today).

Visual story telling part 2: Cap Metro data

To begin, we analyzed the total volume of ridership (Boarding and Alighting) across three distinct periods of the day (Morning, Afternoon, and Evening) on both weekdays and weekends. We noticed that on weekdays, Morning and Afternoon were by far the most active times. Meanwhile, on weekends, afternoon was the most popular time followed by evening. However, the total volume of ridership was much lower on weekends as expected since people are using the Metro for leisure rather than commuting.

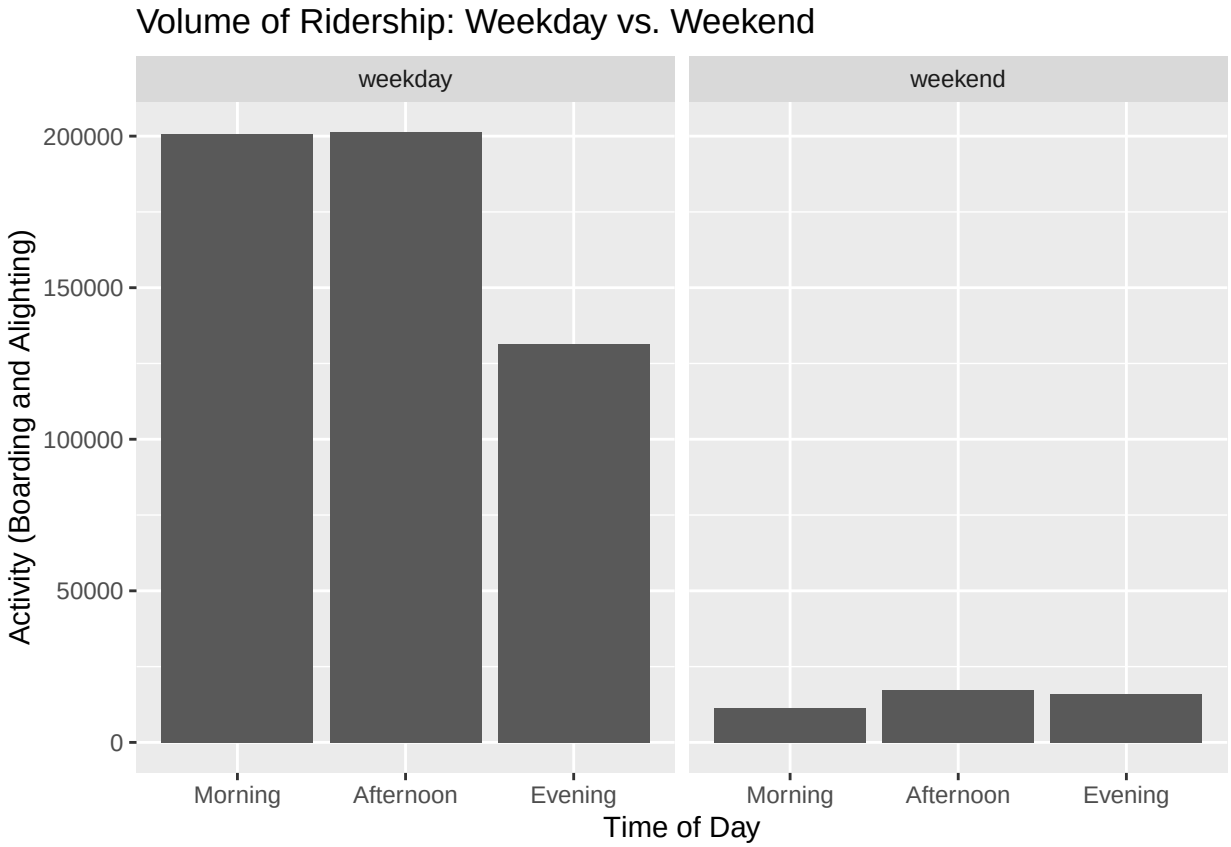
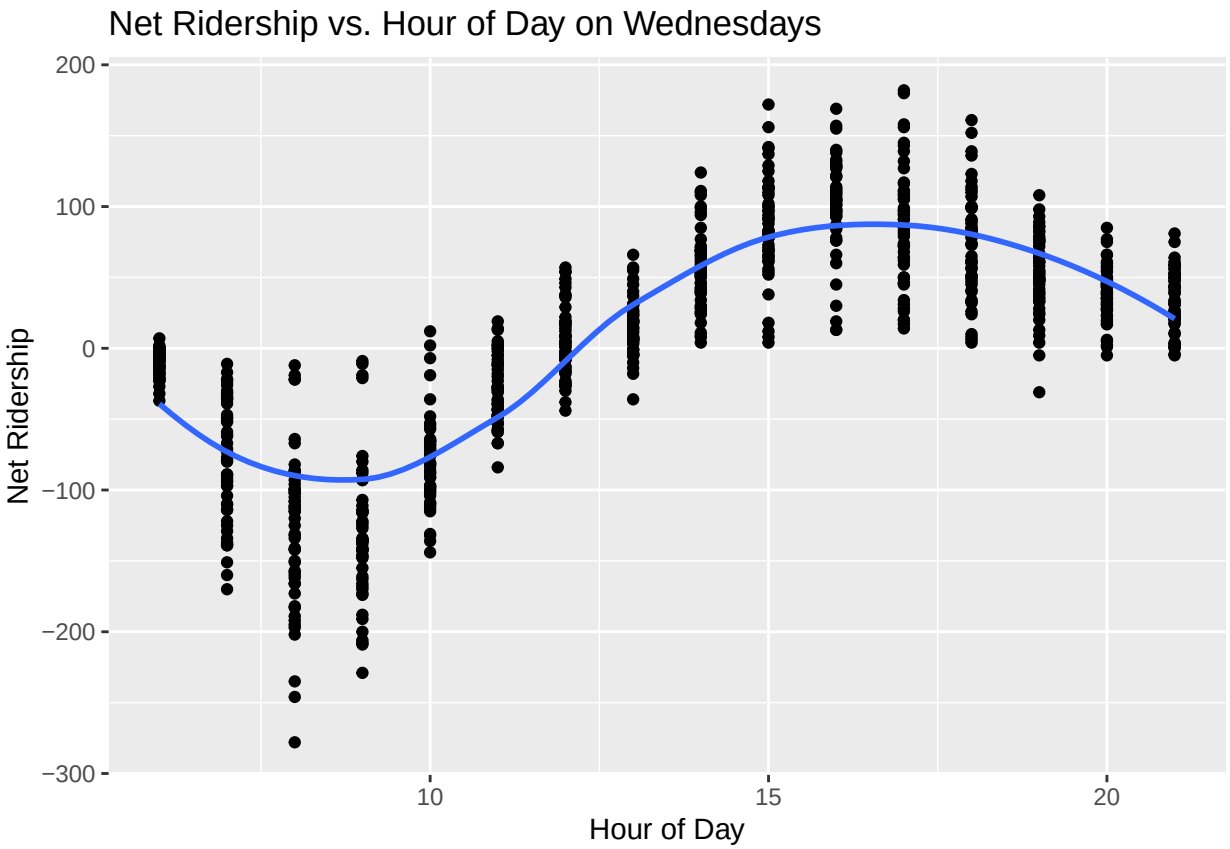


Figure 1

Following our analysis above, we decided to hone in on a specific weekday (Wednesday) and a specific weekend day (Sunday) to get a better understanding of the hourly trends of net ridership. What we discovered was net ridership was most negative (i.e. more people getting off the bus) in the morning hours of 8-9am. From here, average net ridership increases and becomes positive at around noon. It reaches its max between 4-5pm (more people getting on bus to go home).

Figure 2

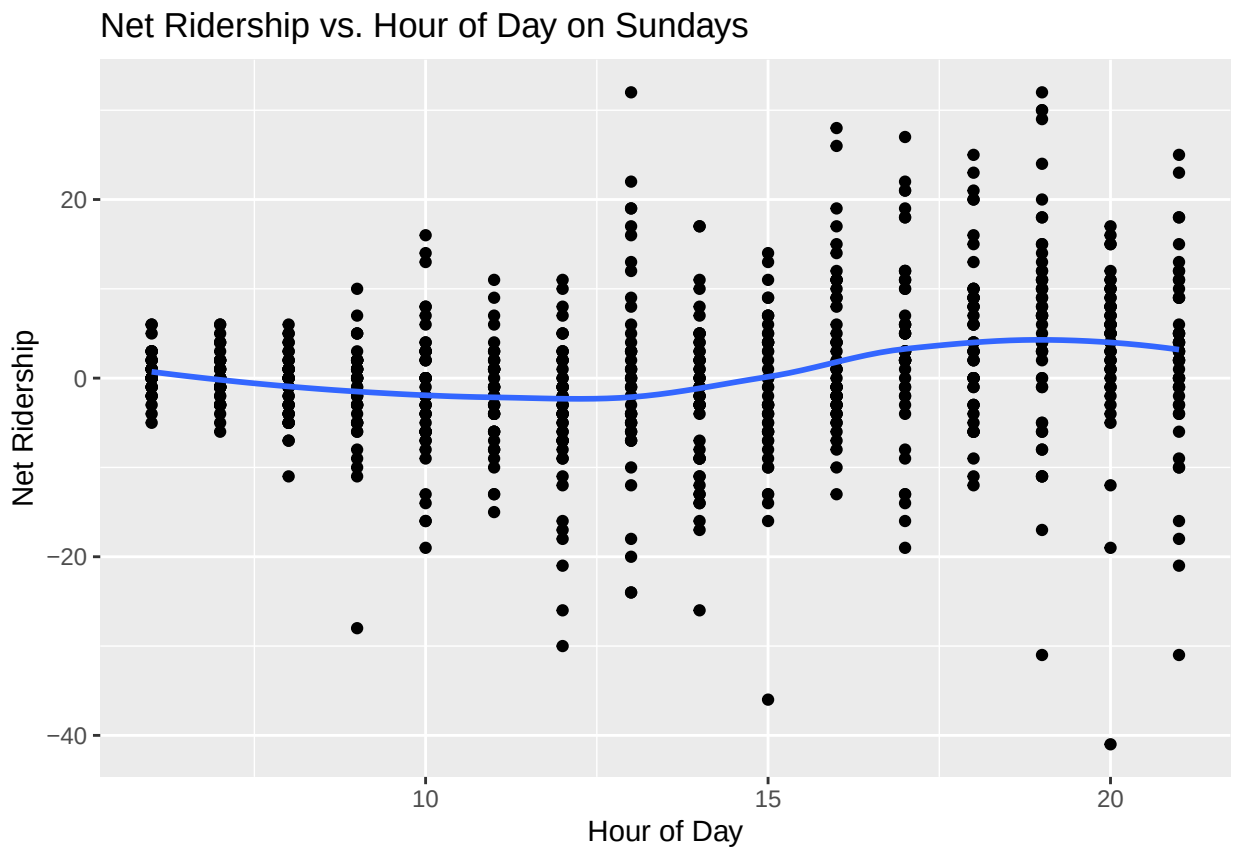
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



On Sundays, the trend of average net ridership is much flatter and less volatile. It appears to reach a minimum at noon, hits 0 at 3pm, and reaches its max at around 5-6. Since people aren't typically commuting to work/school on Sundays, net ridership doesn't show nearly as clear of a pattern compared to Wednesday.

Figure 3

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

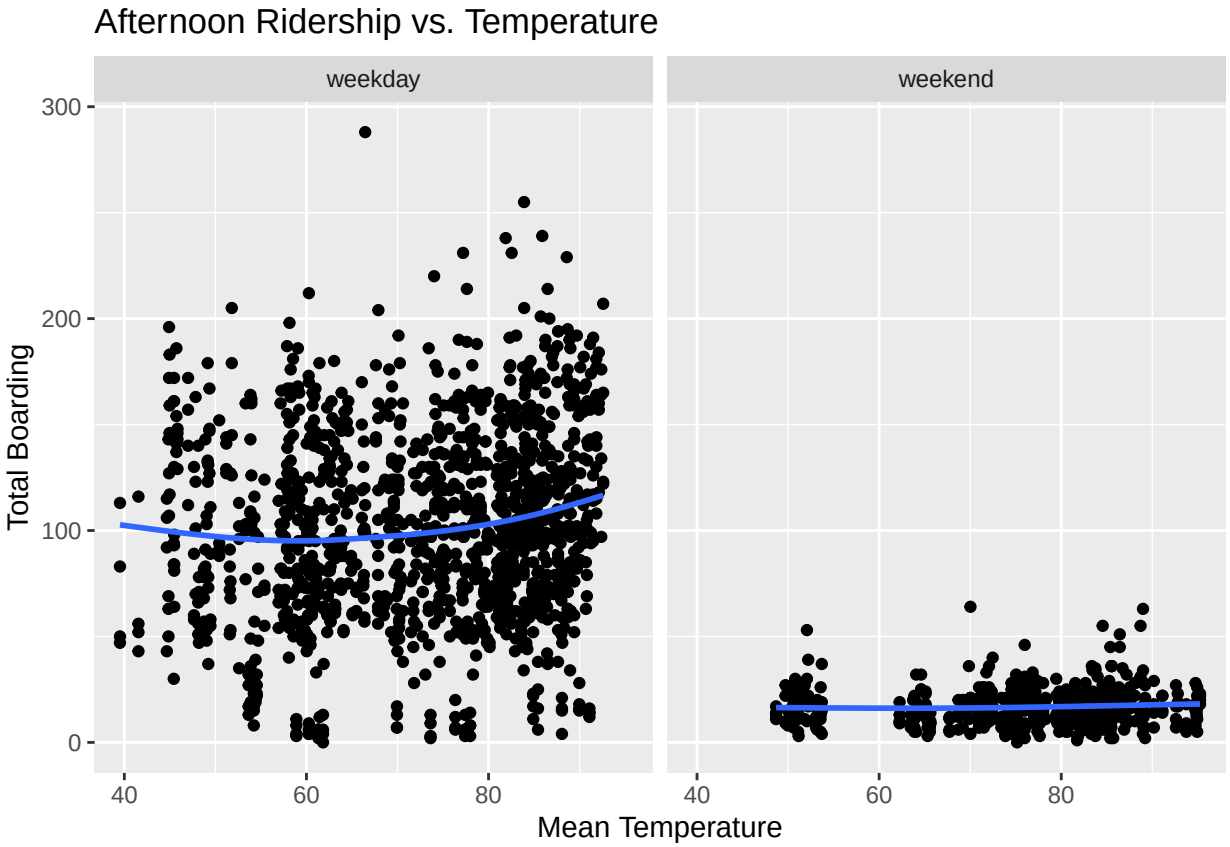


After looking at Figures 1-3, it's clear that afternoons are the most popular boarding time for both weekdays and weekends (positive net ridership). We wanted to see specifically for afternoons, what impact temperature had on curbing or enhancing Metro ridership.

Figure 4

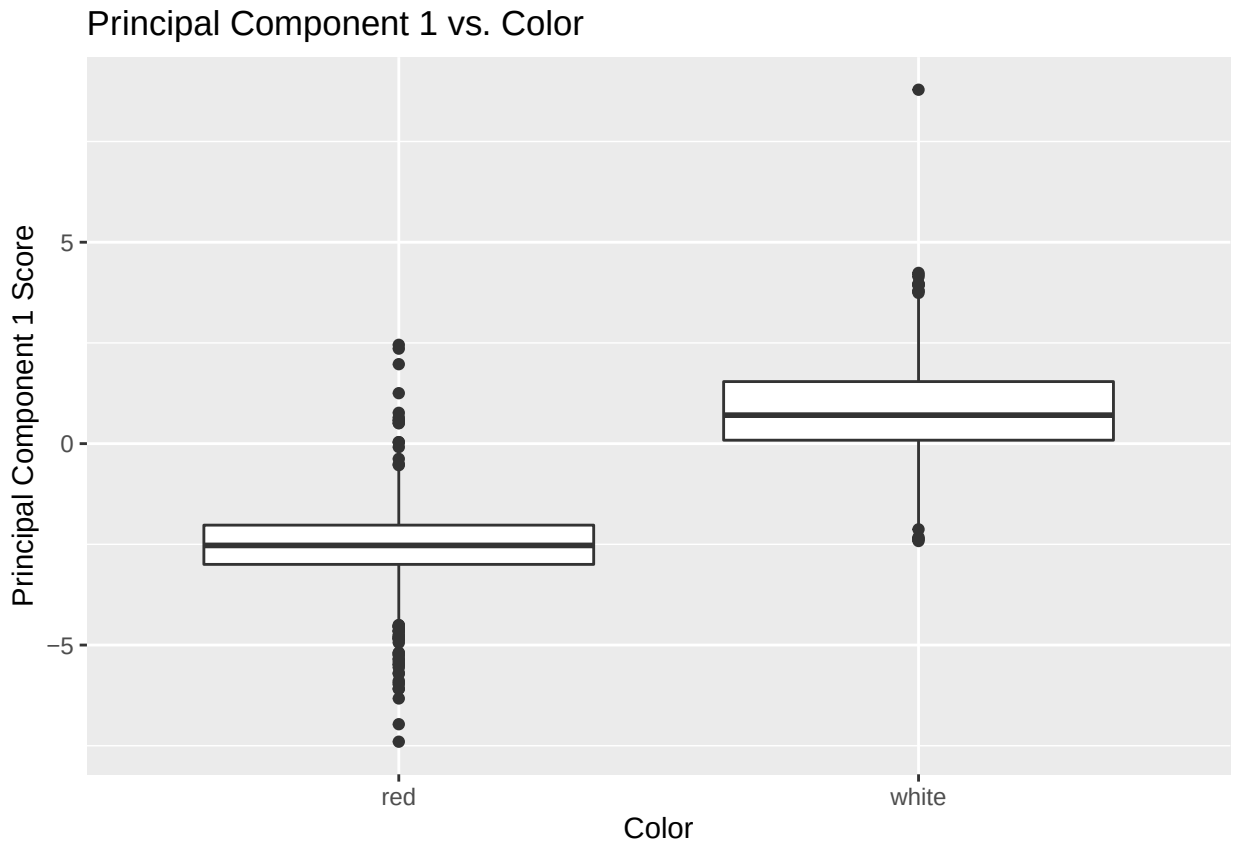
```
afternoon <- subset(cap_metro, cap_metro$time_of_day=="Afternoon")
riders_temp = afternoon %>%
  group_by(timestamp) %>%
  summarize(total_riders = sum(boarding), mean_temp = mean(temperature), weekend = weekend, time_of_day=
ggplot(riders_temp, aes(x = mean_temp, y = total_riders)) + geom_point()+facet_grid(.~weekend)+geom_smoo
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The boarding trend is pretty flat on weekends, but shows an upward curve for more extreme temperatures on weekdays. The heightened sensitivity to temperature on weekday ridership is likely due to people opting for a temperature-controlled bus ride rather than being outdoors. It's also possible that the increased boarding for higher temperatures corresponds to the peak boarding times on weekdays of 3-4pm which also happens to be the hottest part of the day generally.

Clustering and PCA



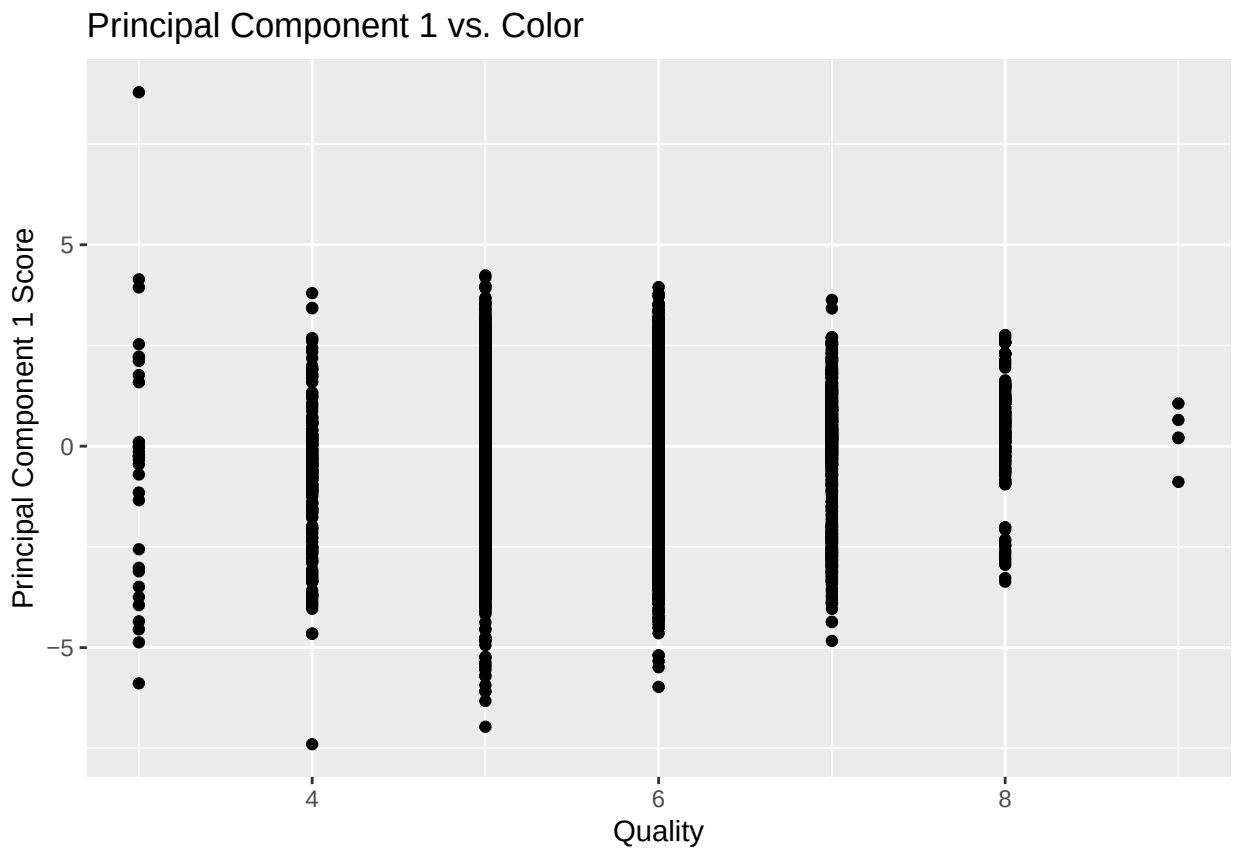

```
cor_df <- melt(as.data.frame(cor(wine_copy$pc1,wine_quant)))
```

```
## No id variables; using all as measure variables
```

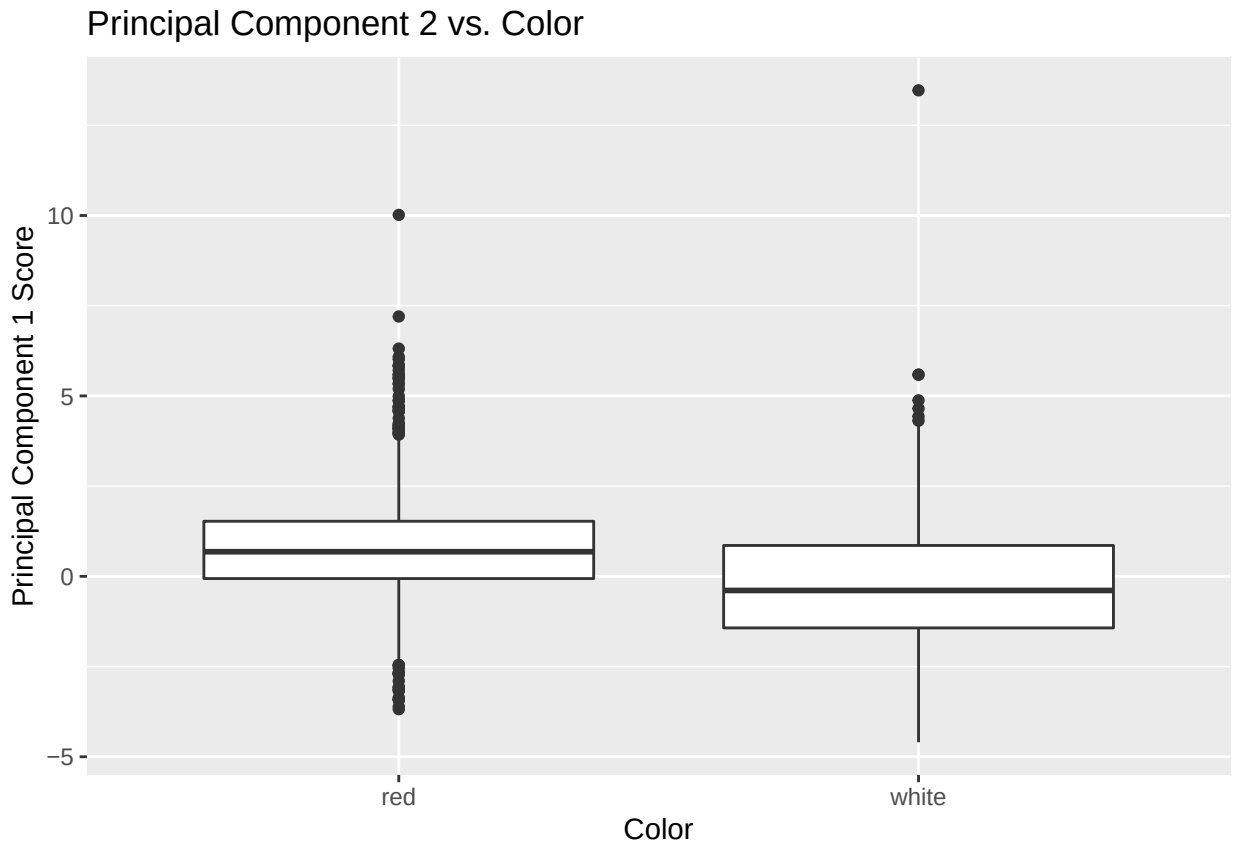
```
colnames(cor_df) <- c("Variable","Correlation with PC1")
kable(cor_df)
```

Variable	Correlation with PC1
fixed.acidity	-0.4156657
volatile.acidity	-0.6627662
citric.acid	0.2652552
residual.sugar	0.6021261
chlorides	-0.5049850
free.sulfur.dioxide	0.7500712
total.sulfur.dioxide	0.8484251
density	-0.0782190
pH	-0.3806569
sulphates	-0.5119869
alcohol	-0.1852700

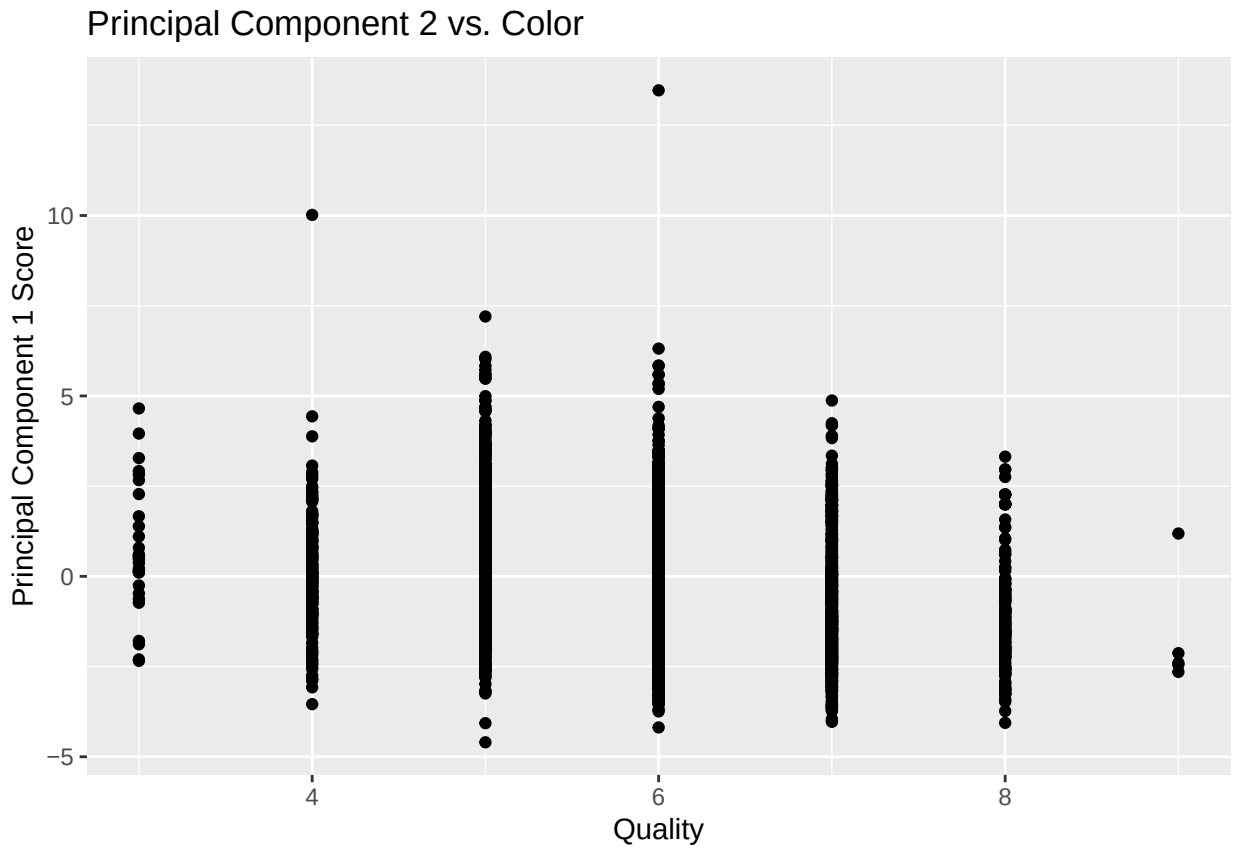
```
ggplot(wine_copy, aes(x = quality, y = pc1,group=color)) +geom_point() + xlab("Quality") + ylab("Princi
```




```
wine_copy$pc2 <- wine_pca$x[,2]
ggplot(wine_copy, aes(x = color, y = pc2,group=color)) +geom_boxplot() + xlab("Color") + ylab("Principal
```




```
wine_copy$pc2 <- wine_pca$x[,2]
ggplot(wine_copy, aes(x = quality, y = pc2,group=color)) +geom_point() + xlab("Quality") + ylab("Princi
```



Cluster 1 is mostly red wines, whereas Cluster 2 is mostly white wines. Even just making two clusters distinguishes between the two wine colors very well (98.58% of wines were grouped correctly).

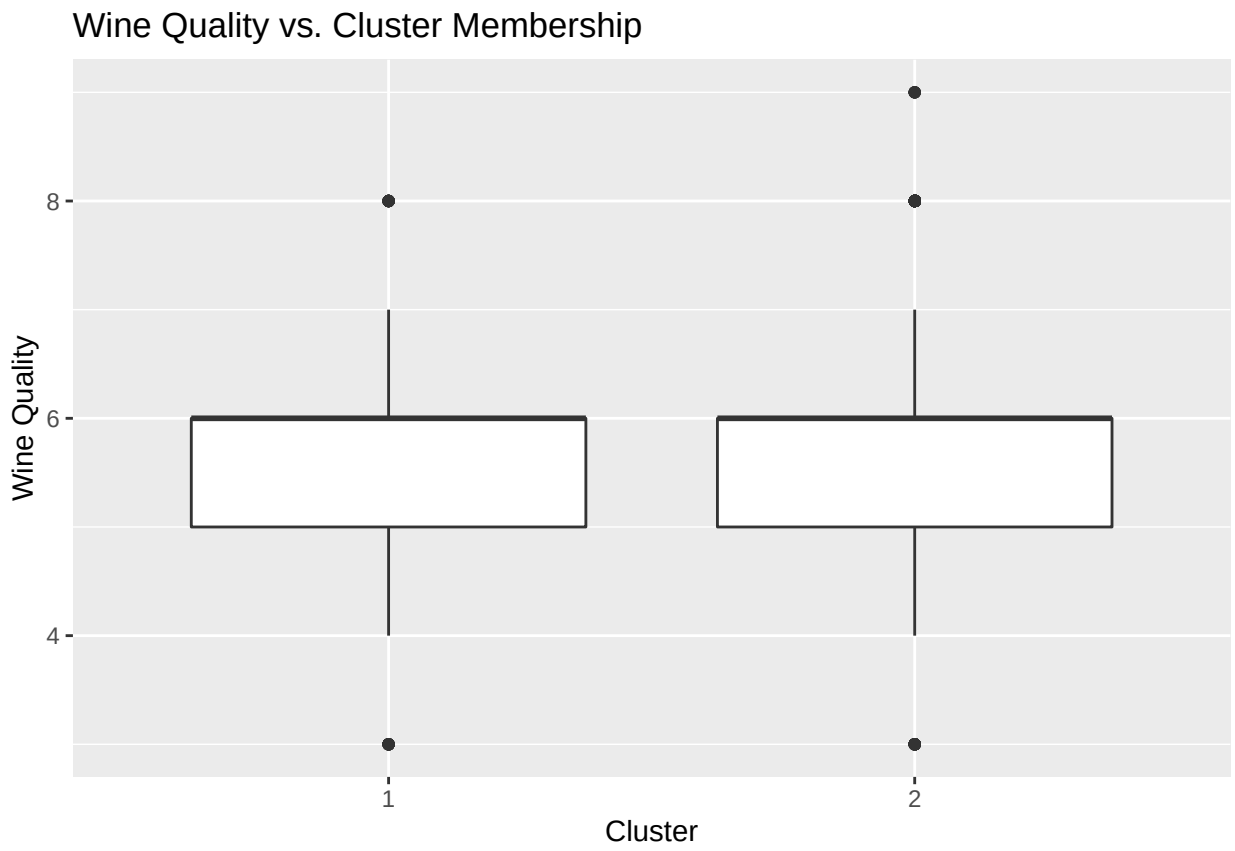
Figure 1

	red	white
Cluster 1	1575	68
Cluster 2	24	4830

While the 2 clusters separated out the two wine colors well, they don't seem to distinguish between wine quality because the median quality is essentially the same for both clusters. Even if we increase the number of clusters pretty dramatically up to 10, there still doesn't appear to be major quality differences between the box plots.

Figure 2

```
wine$cluster = as.factor(wine_clusters$cluster)
ggplot(wine, aes(x = cluster, y = quality)) + geom_boxplot() + xlab("Cluster") + ylab("Wine Quality") +
```



Market Segmentation

We decided to define market segments for this problem as clusters identified through the k-means clustering approach. We omitted the Twitter user's randomly generated ID when creating the clusters and instead only used the scores for each Tweet interest. We settled on creating 10 market segments (clusters) as 10 seemed to be a sweet spot between capturing legitimate differences between Twitter followers while also not overloading the company with too many market segments to try to understand.

While NutrientH20 might be interested in all 10 of the market segments that we've identified, they'll likely care the most about the segments that would be most receptive to their products. Thus, we found the 3 market segments with the highest average scores for the "health_nutrition" interest given that NutrientH20 seems to be a health-oriented company. The summaries of the three segments are below:

```
##    Market Segment Interest Category Average Score
## 1               1  health_nutrition     12.541667
## 2               1  personal_fitness      6.651042
## 3               1           chatter      3.941406
## 4               1           cooking      3.425781
## 5               1          outdoors      2.876302
## 6               1     photo_sharing      2.399740
## 7               1              food      2.205729
## 8               1    current_events      1.514323
## 9               1          shopping      1.283854
## 10              1            travel      1.229167
## 11              3           cooking     11.684211
```

```
## 12              3     photo_sharing      6.088421
## 13              3           fashion      5.985263
## 14              3           chatter      4.246316
## 15              3            beauty      4.208421
## 16              3  health_nutrition      2.269474
## 17              3          shopping      1.755789
## 18              3    current_events      1.751579
## 19              3       college_uni      1.496842
## 20              3            travel      1.461053
## 21              4           chatter      4.653061
## 22              4  health_nutrition      2.795918
## 23              4     photo_sharing      2.448980
## 24              4            travel      2.244898
## 25              4          politics      2.244898
## 26              4       college_uni      1.918367
## 27              4     sports_fandom      1.897959
## 28              4    current_events      1.877551
## 29              4           cooking      1.795918
## 30              4  personal_fitness      1.755102
```

From the three most promising market segments, the first one appears to be the most appealing to NutrientH20. Members of this segment have by far the highest average scores for the "health_nutrition" interst category, and also have high average scores for "fitness" which probably is something closely related to what NutrientH20 does as well. There are 768 Twitter users in the most promising market segment of segment 1, and then 475 and 49 in segments 3 and 4, respectively. Focusing in on these market segment will hopefully yield more future customers than trying to market to all Twitter followers.

«««< HEAD

The Reuters Corpus

Figure out how to download this data - For now, just clone the Github and copy the folder over; I've added it to .gitignore so it won't be pushed to Github

Association Rule Mining

```
library(arules)
```

```
## Warning: package 'arules' was built under R version 4.1.3
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

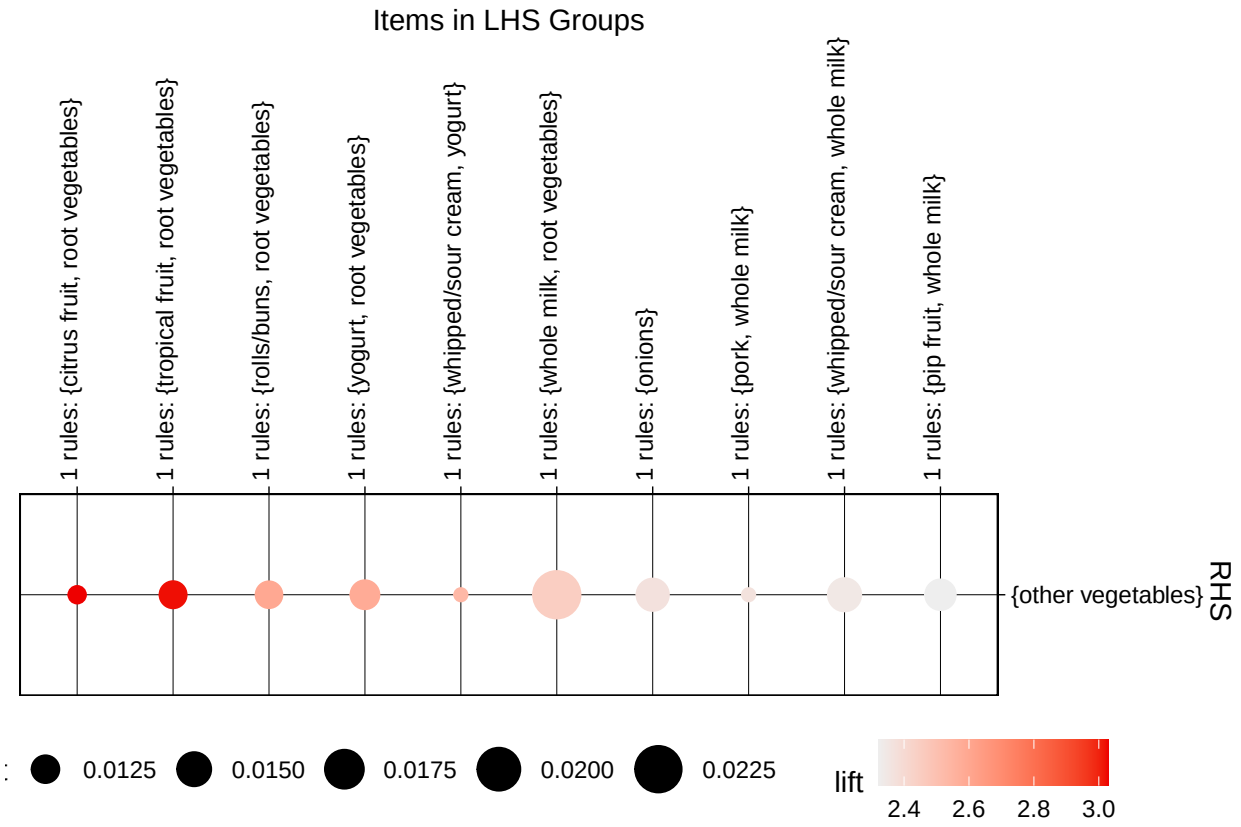
```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(reshape2)
#Read in the groceries.txt file. Find max number of objects
#in a basket so that R doesn't automatically cap the number
#of columns we can have
no_col <- max(count.fields("groceries.txt", sep = ","))
groceries <- read.table("groceries.txt",sep=",",fill=TRUE,col.names=c(1:no_col))
no_col <- max(count.fields("data/groceries.txt", sep = ","))
groceries <- read.table("data/groceries.txt",sep=",",fill=TRUE,col.names=c(1:no_col))
#Add in a column that indicates which customer corresponds to
#the basket (row number)
groceries$customer = as.factor(1:nrow(groceries))
#Get data in long format
groceries <- melt(groceries,id.vars = 'customer')
groceries <- as.data.frame(groceries)
#Drop all values that are blank
groceries <- subset(groceries, groceries$value != "")
#Group the grocery products by the customer who bought them
groceries <- split(x=groceries$value, f=groceries$customer)
#Make sure each customer is only associated with unique
#products in their basket
groceries <- lapply(groceries, unique)
```

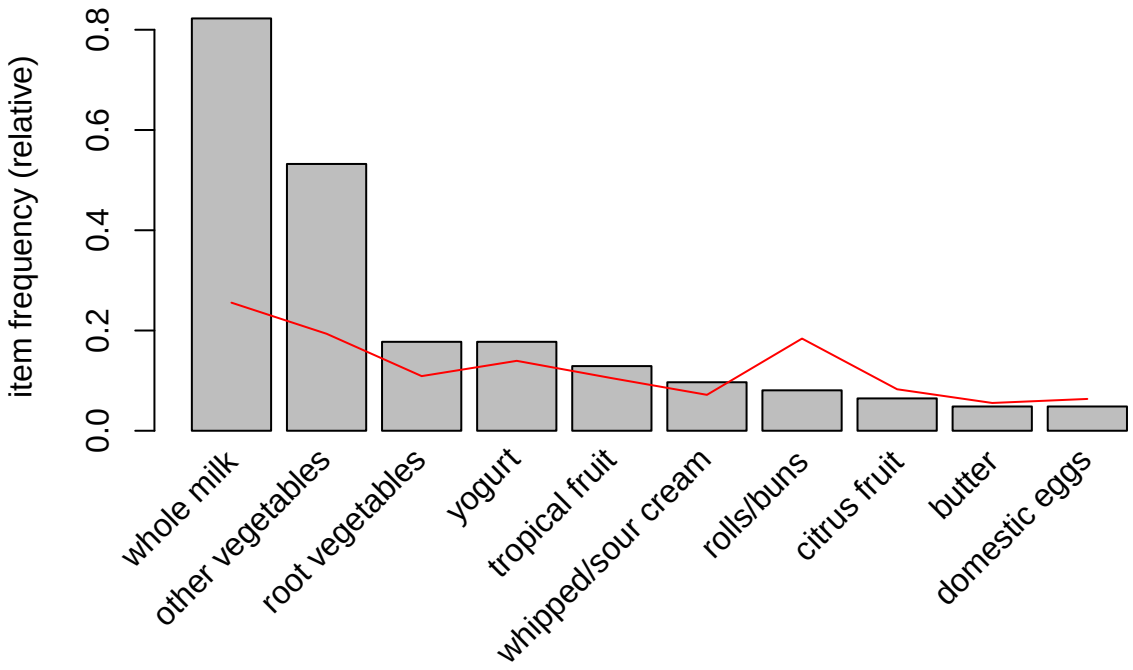
```
library(arulesViz)
```

```
## Warning: package 'arulesViz' was built under R version 4.1.3
```

```
interesting_rules <- head(sort(groceries_rules, by="lift"), 10)
plot(interesting_rules, method="grouped")
```




```
itemFrequencyPlot(items(groceries_rules),population=groceries_trans,topN=10,popCol="red")
```




```
whole_milk_rules <- apriori(groceries_trans,
    parameter=list(support=.01, confidence=.4, maxlen=4),appearance = list(default="lhs", rhs="whole mi
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##       4  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4

## Warning in apriori(groceries_trans, parameter = list(support = 0.01, confidence
## = 0.4, : Mining stopped (maxlen reached). Only patterns up to a length of 4
## returned!
```

```
##  done [0.00s].
## writing ... [43 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
length(whole_milk_rules)/length(groceries_rules)
```

```
## [1] 0.6935484
```

```
#Export to a graphml file so that we can visualize this data in Gephi
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.1.2
```

```
##
## Attaching package: 'igraph'
```

```
## The following object is masked from 'package:arules':
##
##     union
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
groceries_graph = associations2igraph(subset(groceries_rules, lift>1), associationsAsNodes = FALSE)
igraph::write_graph(groceries_graph, file='groceries.graphml', format = "graphml")
```

======= #Export to a graphml file so that we can visualize this data in Gephi library(igraph) library(arulesViz) groceries_graph = associations2igraph(subset(groceries_rules, lift>1), associationsAsNodes = FALSE) igraph::write_graph(groceries_graph, file='groceries.graphml', format = "graphml") "' »»»> 243a1b6b2111a88510e69fdd65cfde561c881fbc