STA380_James_Problems

Grayson Merritt

2024-07-31

Problem 1

We are looking for P(Yes|TC) P(Yes|RC) = .5 P(No|RC) = .5 P(Yes) = .65 P(No) = .35 P(RC) = .3

The total law of probability states that P(A) = the summation of P(A|B) * P(B) The P(Yes) comes from only two conditional probabilities: P(Yes|RC) and P(Yes|TC) So P(Yes)=P(Yes RC)P(RC)+P(Yes TC)P(TC) Using some algebra I can arrange this to (P(Yes) - P(Yes|RC)P(RC)) / P(TC) = P(Yes TC) Thus, P(Yes|TC) is **71.43%**

Part B

Sensitivity = P(P|D) = .993 Specificity = P(N|ND) = .9999 Disease = P(D) = .000025 The question we are solving is: What is the P(D|P)? Bayes Theorem is P(D|P) = (P(P|D)*P(D)) / P(P) We have P(P|D) and P(D), so we need to find P(P). This will require using the rule of total probability So P(P) = P(P|D) P(D) + P(P|ND) * P(ND) So we need P(ND) and P(P|ND) No disease = P(ND) = 1 - P(D) = .999975 P(P|ND) = 1 - P(N|ND) = .0001 (This is the False Positive case) P(P) = .00012 After calculating all of my needed info and applying Bayes Theorem, I get that the Probability of a person having the disease given a positive test is *19.88%*

```
p_yes_rc = .5
p_no_rc = .5
p_yes = .65
p_no = .35
p_rc = .3
p_tc = .7
p_yes_tc = (p_yes-p_yes_rc*p_rc) / p_tc
print(p_yes_tc)
```

```
## [1] 0.7142857
```

```
# Part B
sensitivity = .993
specificity = .9999
disease = .000025
no_disease = 1- disease
no_disease
```

```
## [1] 0.999975
```

```r
false_postive = 1 - specificity
false_postive
```

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

```r
positive = sensitivity * disease + false_postive * no_disease
positive
```

```
## [1] 0.0001248225
```

```r
disease_given_positive = (sensitivity * disease)/ positive
print(disease_given_positive)
```

```
## [1] 0.1988824
```

Question 2

Part A

This table shows the top ten most popular songs since 1958 based on how long they were on the billboard 100. Most of these songs were produced in the last 21 years. I find it interesting that there are no repeats of performers on this top ten list.

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## New names:
## Rows: 327895 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): url, week_id, song, performer, song_id
## dbl (8): ...1, week_position, instance, previous_week_position, peak_positio...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## `summarise()` has grouped output by 'performer'. You can override using the `.groups` argument.
```

```
## # A tibble: 29,389 x 3
## # Groups:   performer [10,061]
##    performer                                song                          count
##    <chr>                                    <chr>                         <int>
##  1 Imagine Dragons                          Radioactive                      87
```

```
##  2 AWOLNATION                                  Sail                           79
##  3 Jason Mraz                                  I'm Yours                      76
##  4 The Weeknd                                  Blinding Lights                76
##  5 LeAnn Rimes                                 How Do I Live                  69
##  6 LMFAO Featuring Lauren Bennett & GoonRock   Party Rock Anthem              68
##  7 OneRepublic                                 Counting Stars                 68
##  8 Adele                                       Rolling In The Deep            65
##  9 Jewel                                       Foolish Games/You Were Meant~  65
## 10 Carrie Underwood                            Before He Cheats               64
## # i 29,379 more rows
```

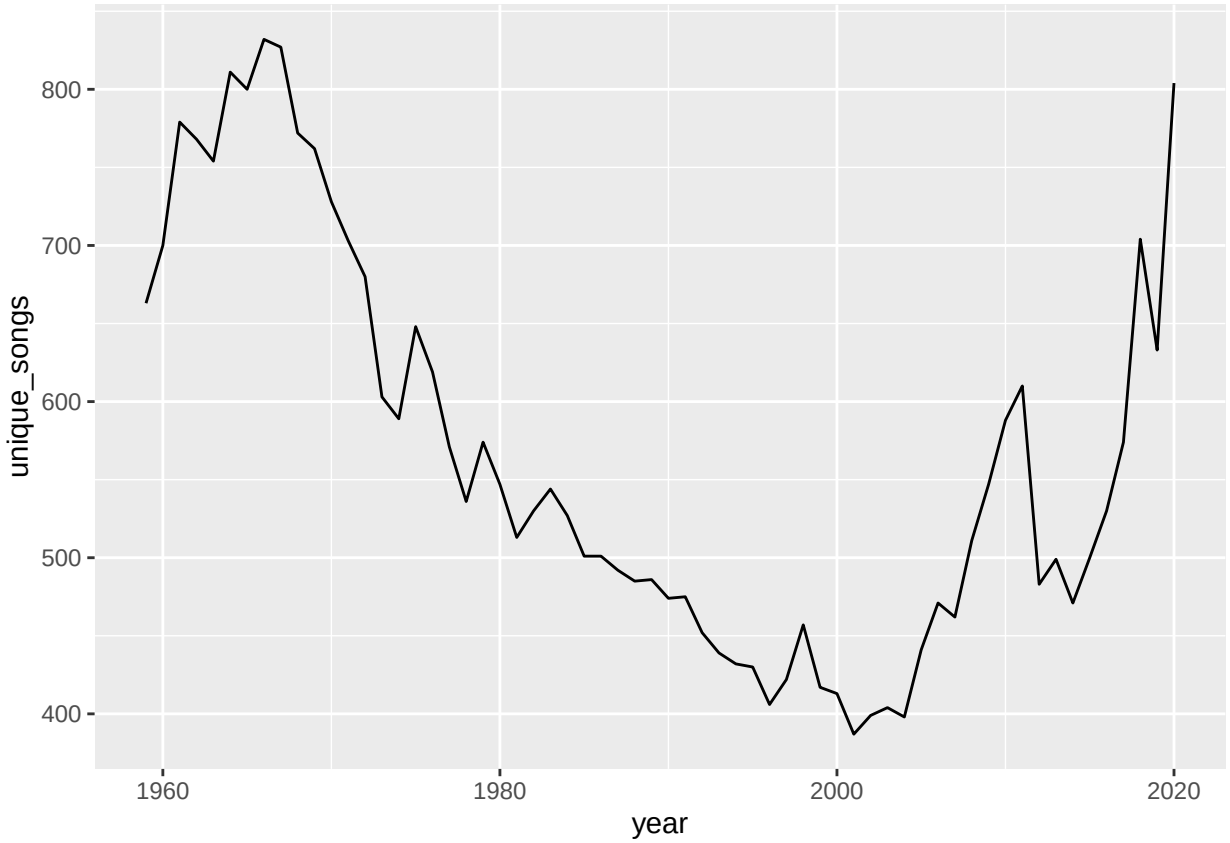
Part B

This plot shows the total number of unique songs that charted the Billboard 100 per year. I guess my parents were correct when they said music was better in the 80's! The number of unique songs that chart peaks at around 1967 and then rapidly declines until around 2002, where more unique songs started to chart. This could potentially be due to the rise if iTunes. There was a decline around 2011 followed by rapid unique song growth.

```
#b
billboard_cutoff = billboard %>% filter(year != 1958 & year != 2021)
table_with_counts = billboard_cutoff %>% group_by(performer,song,year) %>%
  summarize(total_count = n())
```

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

```
unique_song_count = table_with_counts %>% group_by(year) %>%
  summarize(unique_songs = n())
ggplot(unique_song_count) + geom_line(aes(x=year,y=unique_songs))
```

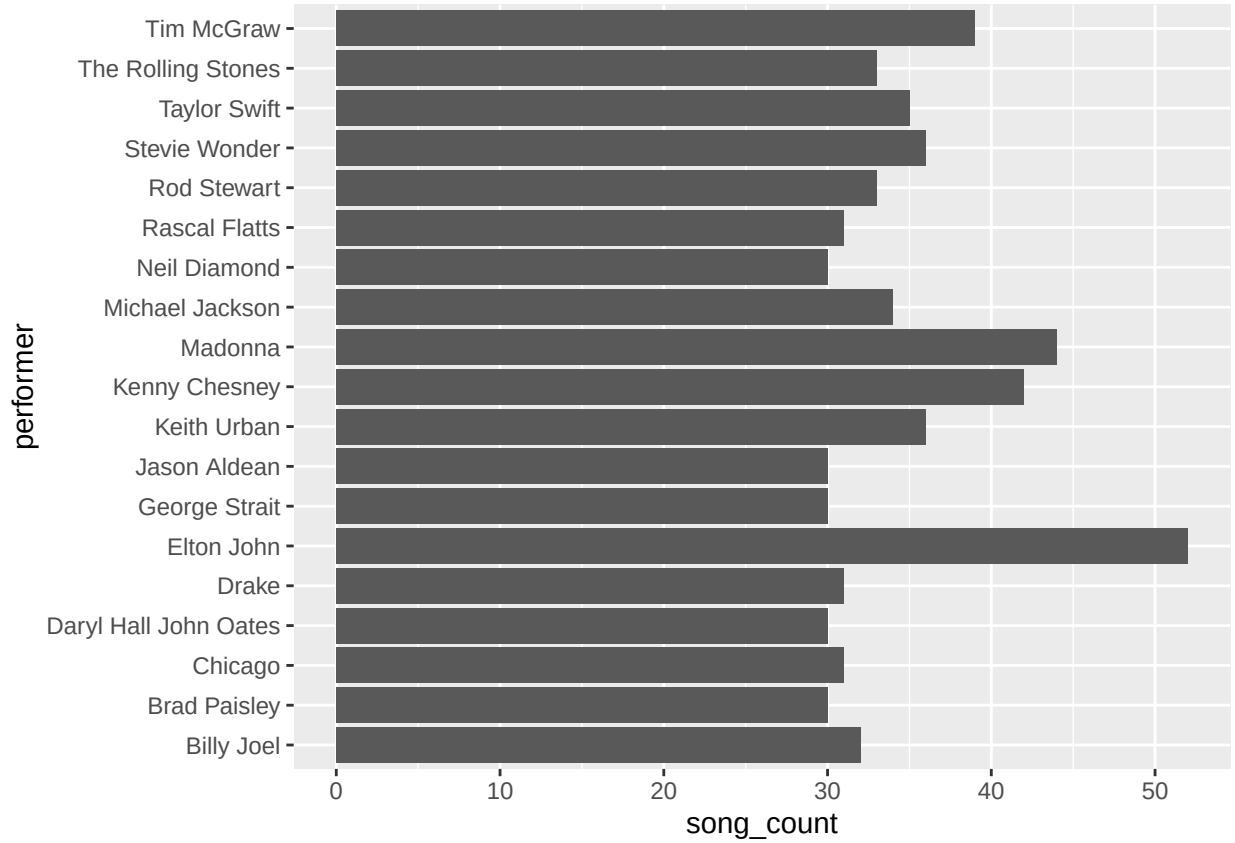


Part C This plot shows artists who have had 30 songs chart for at least ten weeks. Elton John has the highest number of songs with 52 songs. I find it interesting that there are a good amount of country artists filled. I would have thought this list would have been mainly filled with pop and rock artists

```
#C
billboard_ten_week = billboard %>% group_by(performer,song) %>%
  summarize(count = n()) %>%
  filter(count >=10)
```

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

```
billboard_19_artists = billboard_ten_week %>% group_by(performer) %>%
  summarize(song_count =n()) %>% filter(song_count >=30)
ggplot(billboard_19_artists) + geom_col(aes(x=performer, y=song_count)) +
  coord_flip()
```



Problem 3 In order to agree with the stats guru's conclusion that building a green building makes sense, we first have to do our own analysis of the data and see what findings we can come up with. We first took a look at the median rent of green buildings vs non green buildings and confirmed that on average green buildings earn about $2.6 more per sq ft than non green buildings. Our general strategy is to see what factors could affect rent and see if these factors are over or under represented in the green buildings.

```
library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
```

```
## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

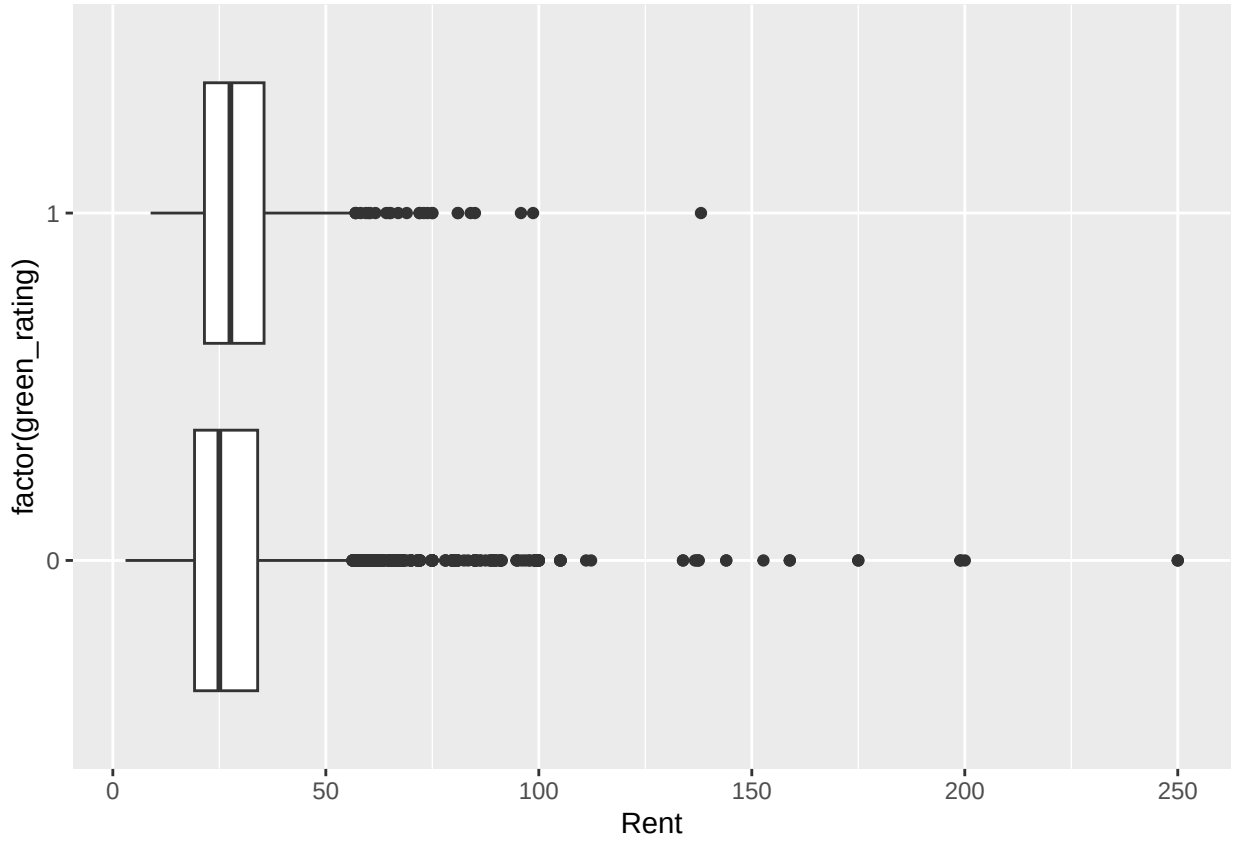
```
greenbuildings = read_csv("greenbuildings.csv")
```

```
## Rows: 7894 Columns: 23

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (23): CS_PropertyID, cluster, size, empl_gr, Rent, leasing_rate, stories...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# also remove the occupancy rate of less than 10%
greenbuildings = greenbuildings %>% filter('leasing_rate' >= 10)

median(Rent ~ green_rating, data=greenbuildings)
```

```
##    0    1
## 25.0 27.6
```

```
ggplot(greenbuildings) +
  geom_boxplot(aes(x=factor(green_rating), y=Rent)) +
  coord_flip()
```



We then looked to see if green buildings tended to be more of Class A. We found that around 80% of green rated buildings are class A, versus only 36% of non green buildings. This may be a potential confounder, as class A buildings will command more rent.

```
# Look at which buildings are "more desirable" (Class A)
xtabs(~ class_a + green_rating, data=greenbuildings) %>%
  prop.table(margin=2)
```

```
##         green_rating
## class_a         0         1
##       0 0.6378138 0.2029197
##       1 0.3621862 0.7970803
```

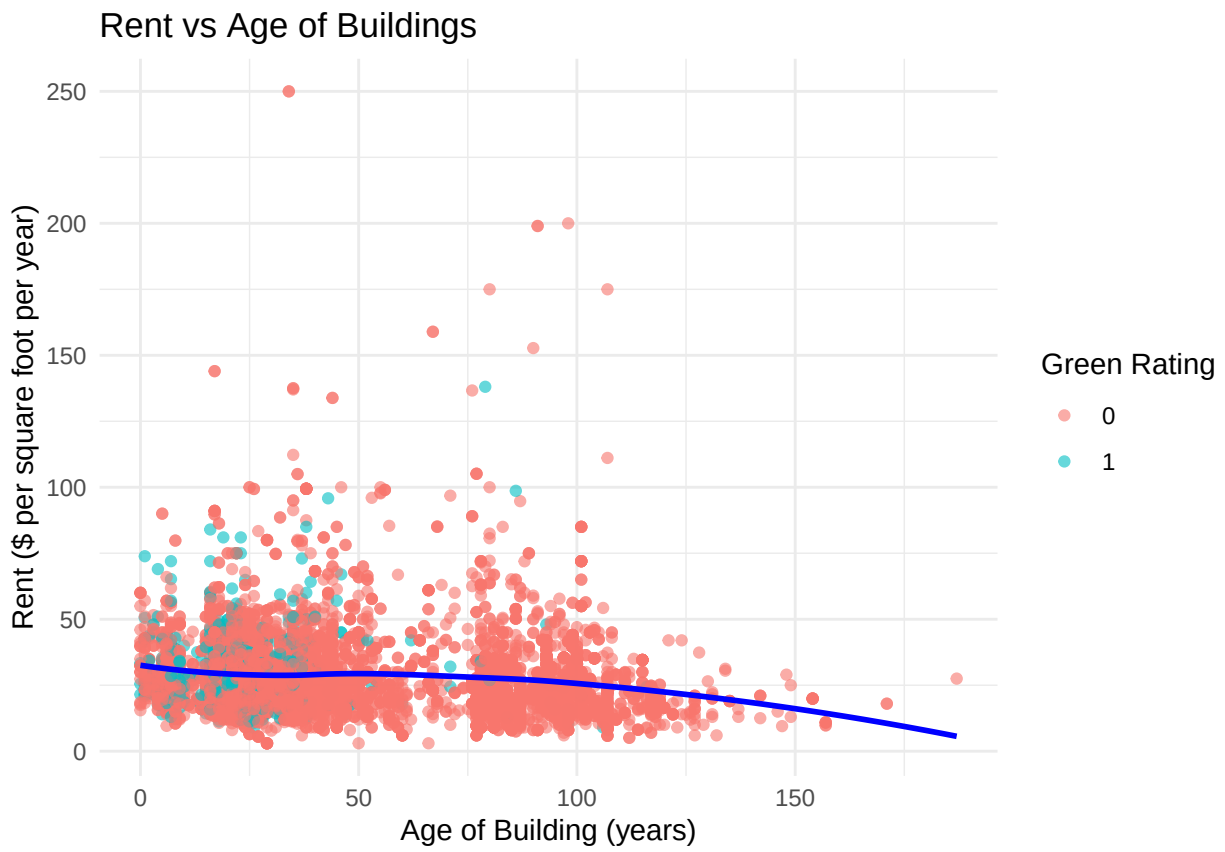
We plotted Rent vs age and found that an older building commands less rent. However, the non green buildings are almost 50 years old on average vs 24 years for green buildings! This could certainly be a confounder.

```
# Look at how age affects rent
mean(age ~ green_rating, data=greenbuildings)
```

```
##        0        1
## 49.46733 23.84526
```

```
ggplot(greenbuildings, aes(x = age, y = Rent)) +
  geom_point(aes(color = as.factor(green_rating)), alpha = 0.6) +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(title = "Rent vs Age of Buildings",
       x = "Age of Building (years)",
       y = "Rent ($ per square foot per year)",
       color = "Green Rating") +
  theme_minimal()
```

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



Rent is LOWER for renovated buildings. This is not what we expected. Renovated buildings command less rent by $3.79! Most green buildings (around 79%), however, are NOT renovated! This means that they are drawing higher prices due to not being renovated.

```
# look to see if renovated affects rent
mean(Rent ~ renovated, data = greenbuildings)
```

```
##        0        1
## 29.85776 26.06570
```

```
xtabs(~ renovated + green_rating, data=greenbuildings) %>%
  prop.table(margin=2)
```

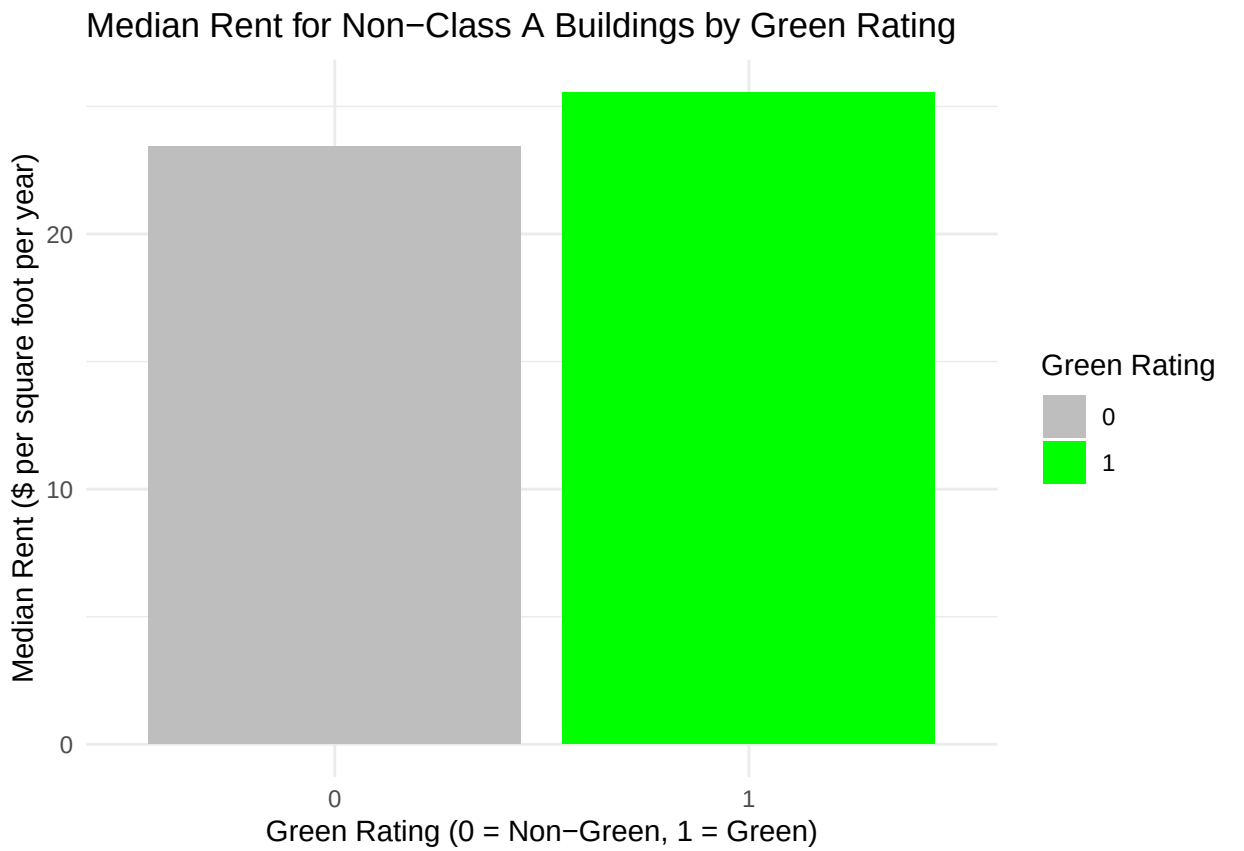
```
##          green_rating
## renovated         0         1
##         0 0.6046608 0.7868613
##         1 0.3953392 0.2131387
```

So far we have found 3 plausible explanations for why green buildings demand higher rent. The class A variable is likely one of the biggest confounders, so we want to see the median rent within non class A buildings faceted by green rating. We found that after adjusting for class A buildings the premium is only $2.12 now.

```
# Look at non class A buildings
median_rent_non_class_a <- greenbuildings %>%
  filter(class_a == 0) %>%  # Exclude Class A buildings
  group_by(green_rating) %>%
  summarise(median_rent = median(Rent))

print(median_rent_non_class_a)
```

```
## # A tibble: 2 x 2
##   green_rating median_rent
##          <dbl>       <dbl>
## 1            0        23.4
## 2            1        25.6
```

```
ggplot(median_rent_non_class_a, aes(x = factor(green_rating), y = median_rent, fill = factor(green_rati
  geom_bar(stat = "identity") +
  labs(title = "Median Rent for Non-Class A Buildings by Green Rating",
       x = "Green Rating (0 = Non-Green, 1 = Green)",
       y = "Median Rent ($ per square foot per year)",
       fill = "Green Rating") +
  scale_fill_manual(values = c("0" = "grey", "1" = "green")) +
  theme_minimal()
```



We think this trend will continue for most of these confounders. We think the best way to adjust for these confounders is to use a technique called matching. Matching relies on a simple principle: compare like with like. In this example, that means if we have a 25-year-old, Class A building that is renovated with a green rating, we try to find another 25-year old, Class A renovated building without a green rating to compare it to. Matching constructs a balanced data set from an unbalanced one. This matched data can them be compared by their rents to see if green buildings truly cause a higher premium.

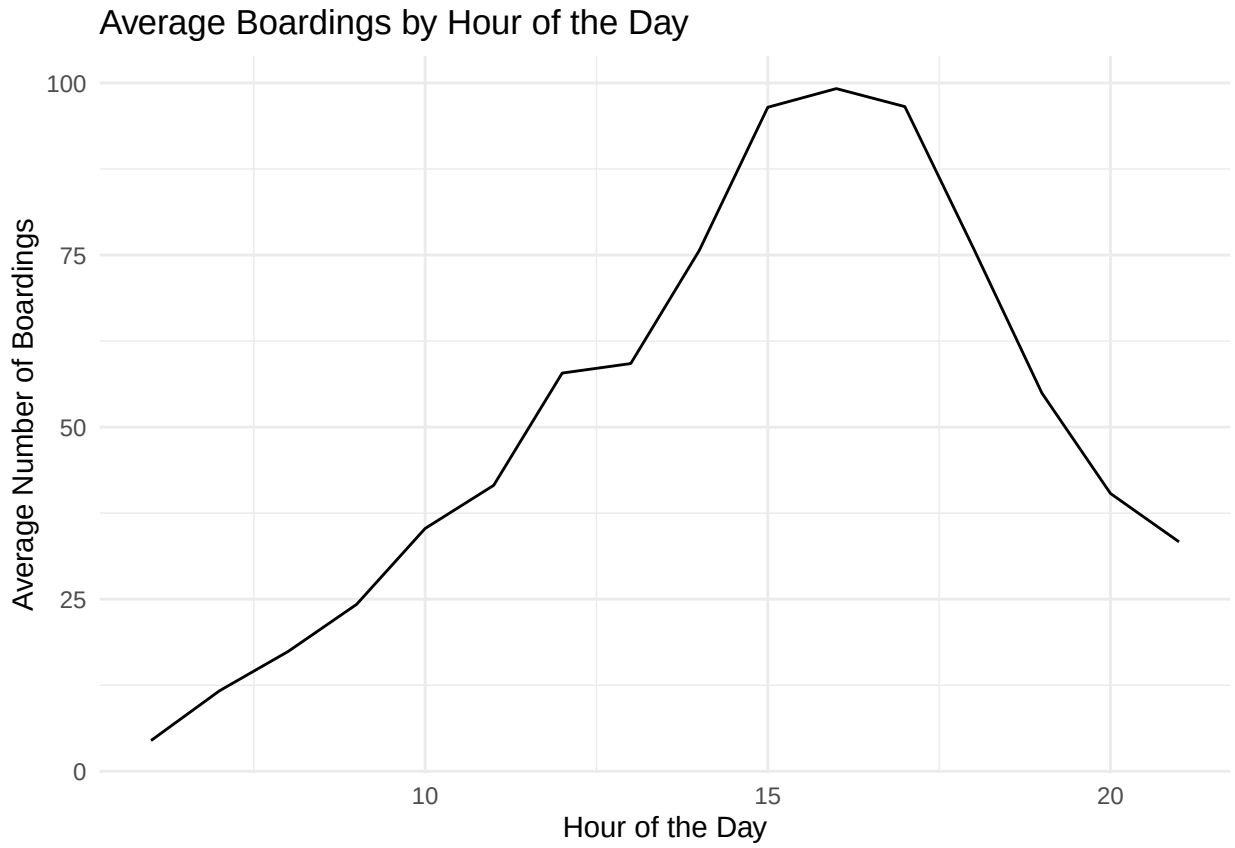
We think the stats guru did not take into account any confounders in his model. While green building may demand higher rent, we think there any more variables at play here.

Problem 4

After changing our timestamp to a datetime data type, we plotted the average boardings by a few different variables.

```
file_path <- "capmetro_UT.csv"
capmetro_UT <- read.csv(file_path)

# Convert timestamp to datetime
capmetro_UT$timestamp <- ymd_hms(capmetro_UT$timestamp)

# average boardings by hour of the day
hour_summary <- capmetro_UT %>%
  group_by(hour_of_day) %>%
  summarize(mean_boardings = mean(boarding))
```

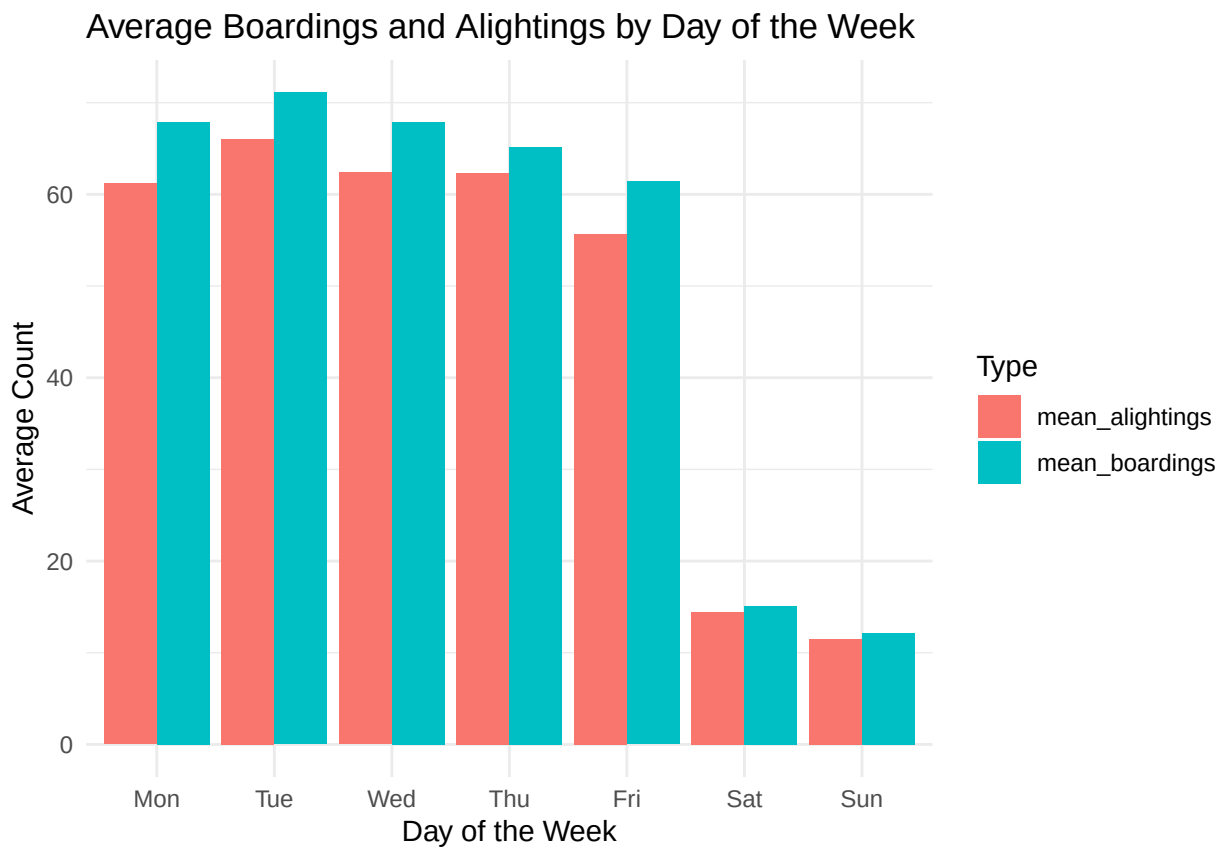
```r
ggplot(hour_summary) +
  geom_line(aes(x = hour_of_day, y = mean_boardings)) +
  labs(title = "Average Boardings by Hour of the Day",
       x = "Hour of the Day",
       y = "Average Number of Boardings") +
  theme_minimal()
```



The relationship between boardings and hour of the day is obviously nonlinear. There is a peak in the late afternoon, followed by a lull overnight and in the early morning, when fewer people ride the bus.

```r
# Convert day_of_week to a factor with levels in the correct order
capmetro_UT$day_of_week <- factor(capmetro_UT$day_of_week,
                                  levels = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))

# Summary of mean boardings and alightings by day of the week
day_summary <- capmetro_UT %>%
  group_by(day_of_week) %>%
  summarise(mean_boardings = mean(boarding), mean_alightings = mean(alighting)) %>%
  pivot_longer(cols = c(mean_boardings, mean_alightings), names_to = "type", values_to = "count")

# Plot average boardings and alightings by day of the week
ggplot(day_summary, aes(x = day_of_week, y = count, fill = type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average Boardings and Alightings by Day of the Week",
       x = "Day of the Week",
       y = "Average Count",
```

```
       fill = "Type") +
  theme_minimal()
```



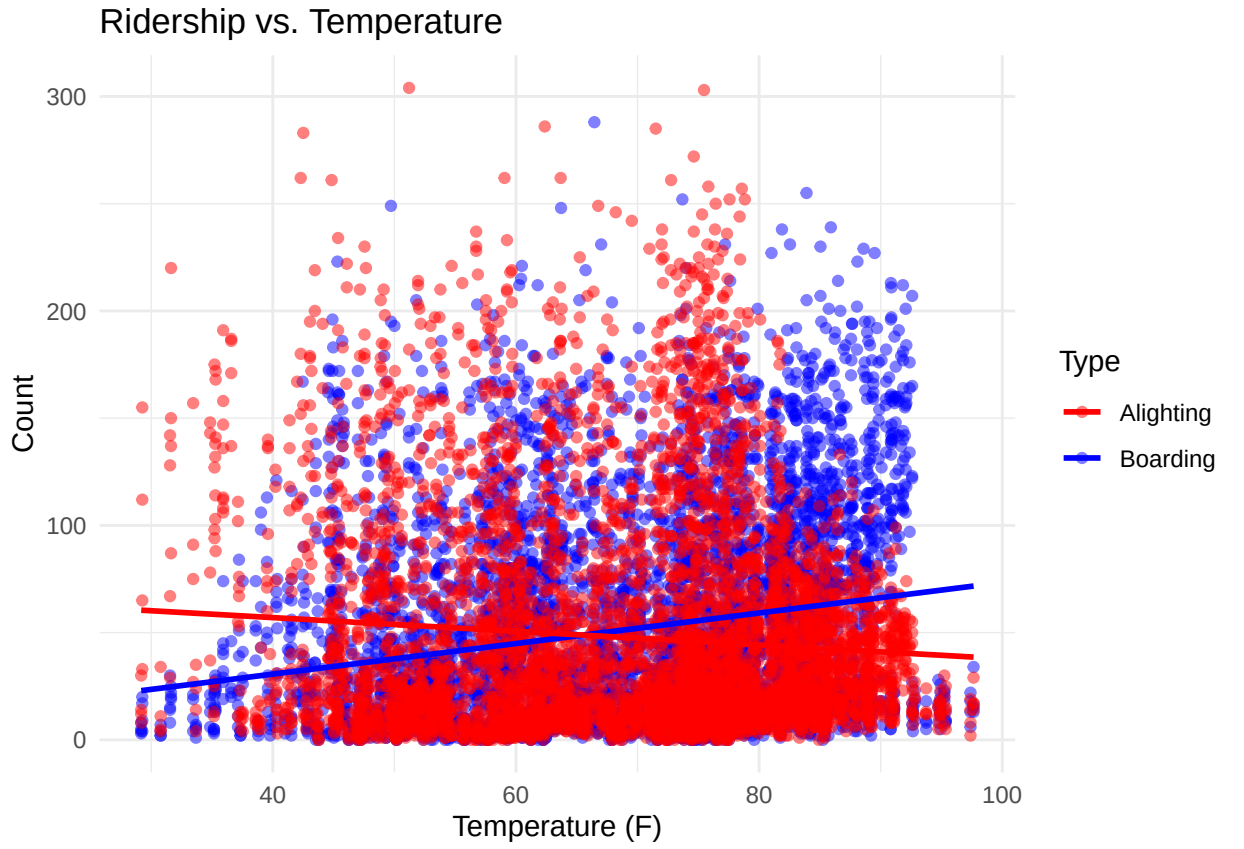
There are more boardings than alightings for every day. It would make sense that boardings would be more accurately tracked than alightings, so we hypothesize that there is a proportion of alightings that is not captured each day. Another hypthesis is that this is data only for UT bus stops. That would mean that more people leave campus on a metro than arrive to campus on a metro. Saturday and Sunday have less boardings and alightings due to classes not being on those days. Boardings and alightings peak on Tuesday and then tail off towards the end of the week. There are less classes on Friday typically, so this drop makes sense.

```
coldest_temperatures <- capmetro_UT %>%
  arrange(temperature) %>%
  select(timestamp, temperature) %>%
  head()

# Print the coldest few values
print(coldest_temperatures)
```

```
##             timestamp temperature
## 1 2018-11-14 06:00:00       29.18
## 2 2018-11-14 06:15:00       29.18
## 3 2018-11-14 06:30:00       29.18
## 4 2018-11-14 06:45:00       29.18
## 5 2018-11-14 07:00:00       29.27
## 6 2018-11-14 07:15:00       29.27
```

```
ggplot() +
  geom_point(data = capmetro_UT, aes(x = temperature, y = boarding, color = "Boarding"), alpha = 0.5) +
  geom_point(data = capmetro_UT, aes(x = temperature, y = alighting, color = "Alighting"), alpha = 0.5)
  geom_smooth(data = capmetro_UT, aes(x = temperature, y = boarding, color = "Boarding"), method = "lm"
  geom_smooth(data = capmetro_UT, aes(x = temperature, y = alighting, color = "Alighting"), method = "l
  labs(title = "Ridership vs. Temperature",
       x = "Temperature (F)",
       y = "Count",
       color = "Type") +
  scale_color_manual(values = c("Boarding" = "blue", "Alighting" = "red")) +
  theme_minimal()
```

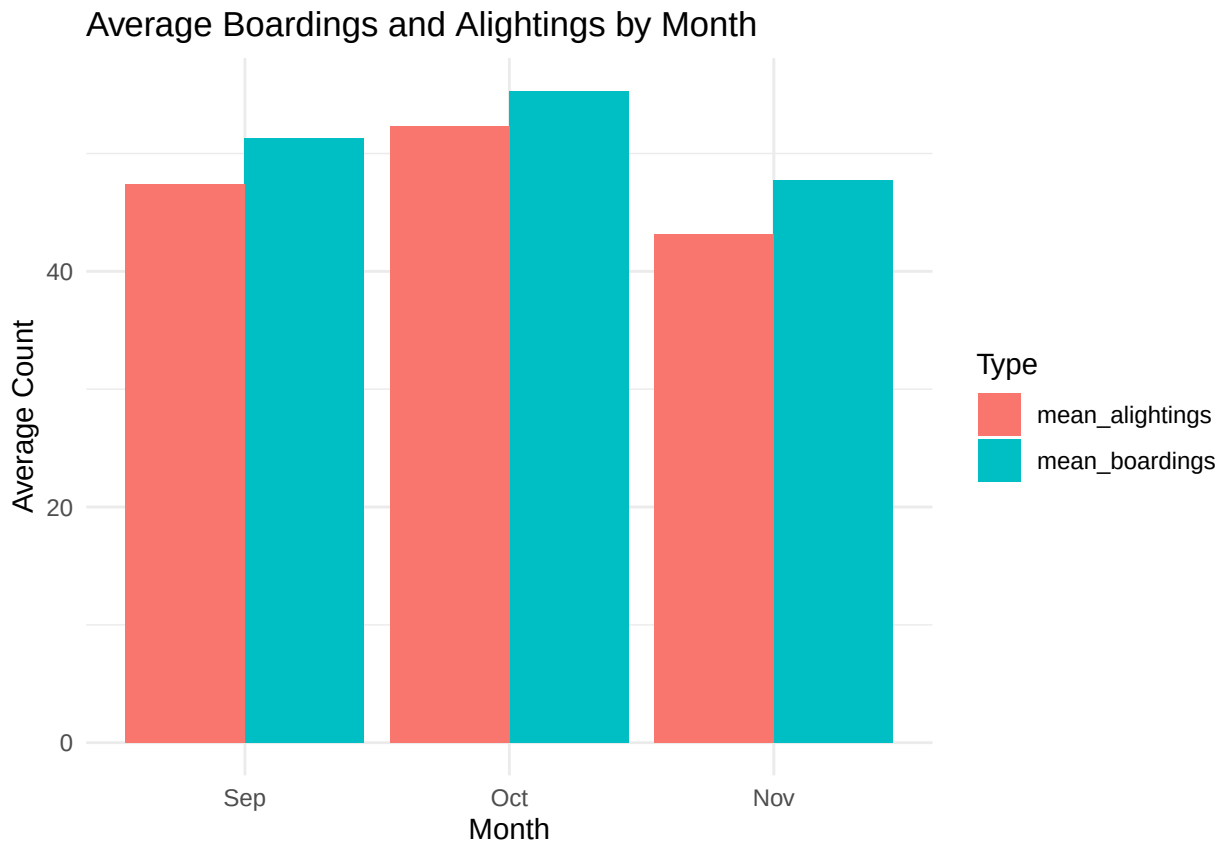
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



This plot is very messy. Of note is that the coldest temperature that Fall 2018 Semester is around 29 degrees. It appears that there is a positive relationship between boardings and temperature but a negative relationship between temperature and alighting! The trendlines are likely influenced by outliers though.

```
# Convert month to a factor with levels in the correct order
capmetro_UT$month <- factor(capmetro_UT$month,
                            levels = c("Sep", "Oct", "Nov"))

# Summary of mean boardings and alightings by month
month_summary <- capmetro_UT %>%
```

```
  group_by(month) %>%
  summarise(mean_boardings = mean(boarding, na.rm = TRUE),
            mean_alightings = mean(alighting, na.rm = TRUE)) %>%
  pivot_longer(cols = c(mean_boardings, mean_alightings), names_to = "type", values_to = "count")

# Plot average boardings and alightings by month
ggplot(month_summary, aes(x = month, y = count, fill = type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average Boardings and Alightings by Month",
       x = "Month",
       y = "Average Count",
       fill = "Type") +
    theme_minimal()
```



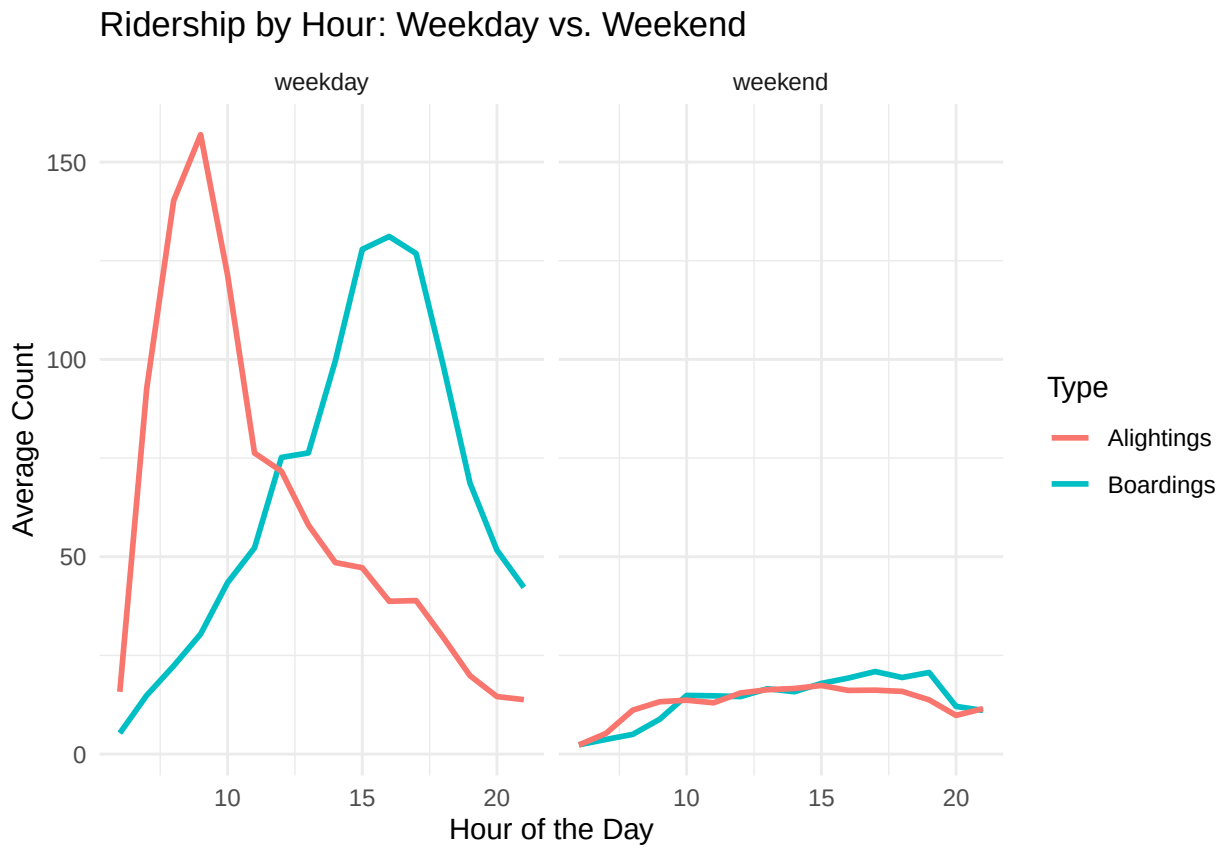
There are more boardings and alightings in October than September. This is likely due to there being no holidays in October, making it a busier month. Football games are in full force and Halloween is a big deal. Conversely, November has less people on the metro. We think this is due to school burnout causing less people to go to school plus the Thanksgiving break that affects at least a week of data.

```
# ridership by weekend status
weekend_summary <- capmetro_UT %>%
  group_by(weekend, hour_of_day) %>%
  summarise(mean_boardings = mean(boarding), mean_alightings = mean(alighting))
```

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

```
ggplot(weekend_summary) +
  geom_line(aes(x = hour_of_day, y = mean_boardings, color = "Boardings"), size = 1) +
  geom_line(aes(x = hour_of_day, y = mean_alightings, color = "Alightings"), size = 1) +
  facet_wrap(~weekend) +
  labs(title = "Ridership by Hour: Weekday vs. Weekend",
       x = "Hour of the Day",
       y = "Average Count",
       color = "Type") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



The graphs between weekday and weekend are strikingly different! Weekends see a lot less traffic as classes are not in session. Weekdays have a peak alighting at around 8-9AM as students and faculty head to class. There is more boardings beyond 12PM with a peak at around 6PM as students and faculty leave campus for the day.

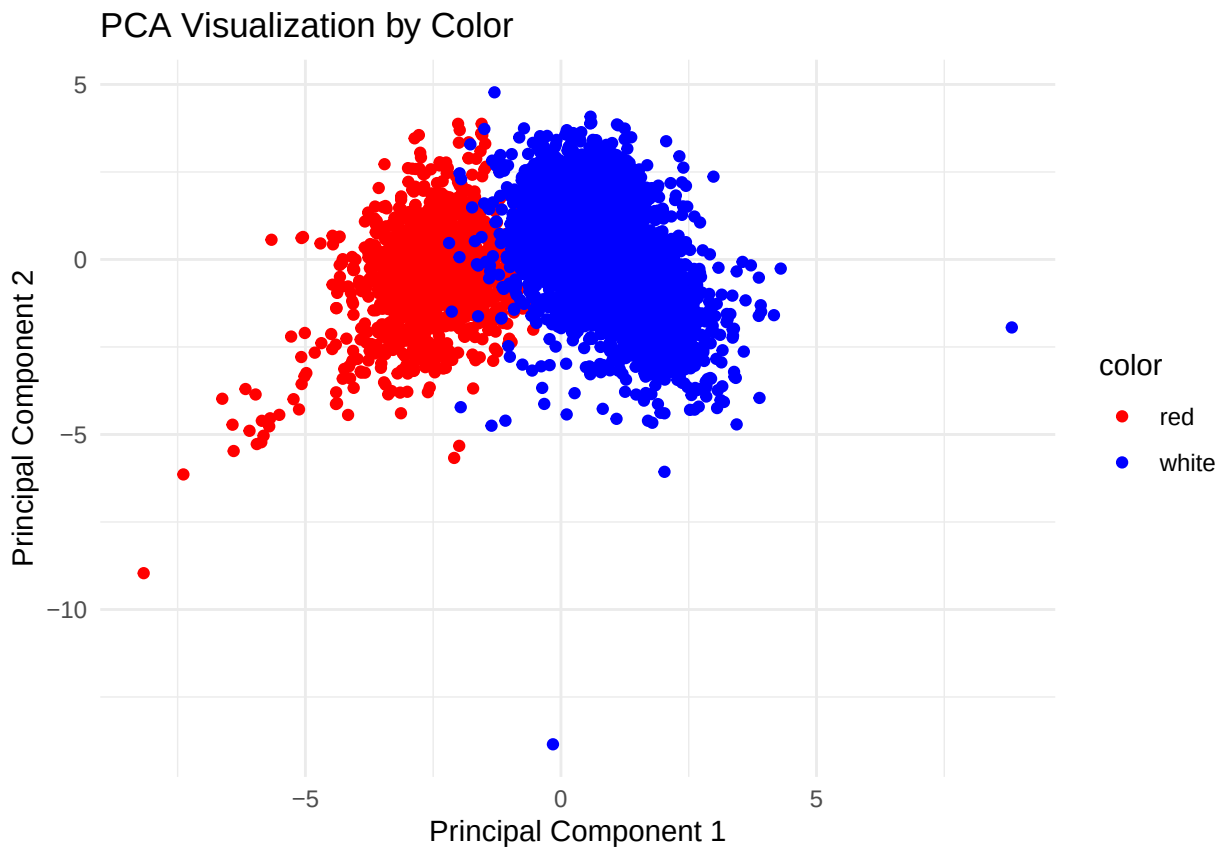
Problem 5

First we had to load our data and engineer it into a usable format for our clustering algorithms. We narrowed the data down to only the 11 chemical properties and then scaled the data while also removing the data.

```r
library(Rtsne)

# Keep just the chemicals, scale the data, and then remove duplicates
set.seed(19)
wine = read.csv('wine.csv')
wine = unique(wine)
wineSubset = select(wine, -color, -quality)
wineSubsetScaled = scale(wineSubset)
duplicate_rows = duplicated(wineSubsetScaled) | duplicated(wineSubsetScaled, fromLast = TRUE)
wineSubsetScaled = wineSubsetScaled[!duplicate_rows, ]
wine = wine[!duplicate_rows, ]
```

Then we run PCA on the data. About half the data can be captured in two dimensions. The dataframe shows how much data is captured by each PCA component. The graph shows that PCA does a decent job separating the wines into color based on the chemical properties. The clusters are in close vicinity after reducing the dimensions from 11 to 2. There is some overlap in the middle between the two clusters.

```r
# scale allows us to make sure each variable contributes equally
pcaResult = prcomp(wineSubsetScaled, scale = TRUE)

# get the first and second principal component
pcaDataColor = data.frame(PC1 = pcaResult$x[, 1], PC2 = pcaResult$x[, 2], color = wine$color)

ggplot(pcaDataColor, aes(x = PC1, y = PC2, color = color)) +
  geom_point() +
  labs(title = "PCA Visualization by Color", x = "Principal Component 1", y = "Principal Component 2") +
  scale_color_manual(values = c("white" = "blue", "red" = "red"))+
  theme_minimal()
```



We then used K means clustering to try and see how accurate PCA is. The clustering shows a majority of reds in one cluster and a majority of whites in the other. This is good news. The accuracy was 98.16%.

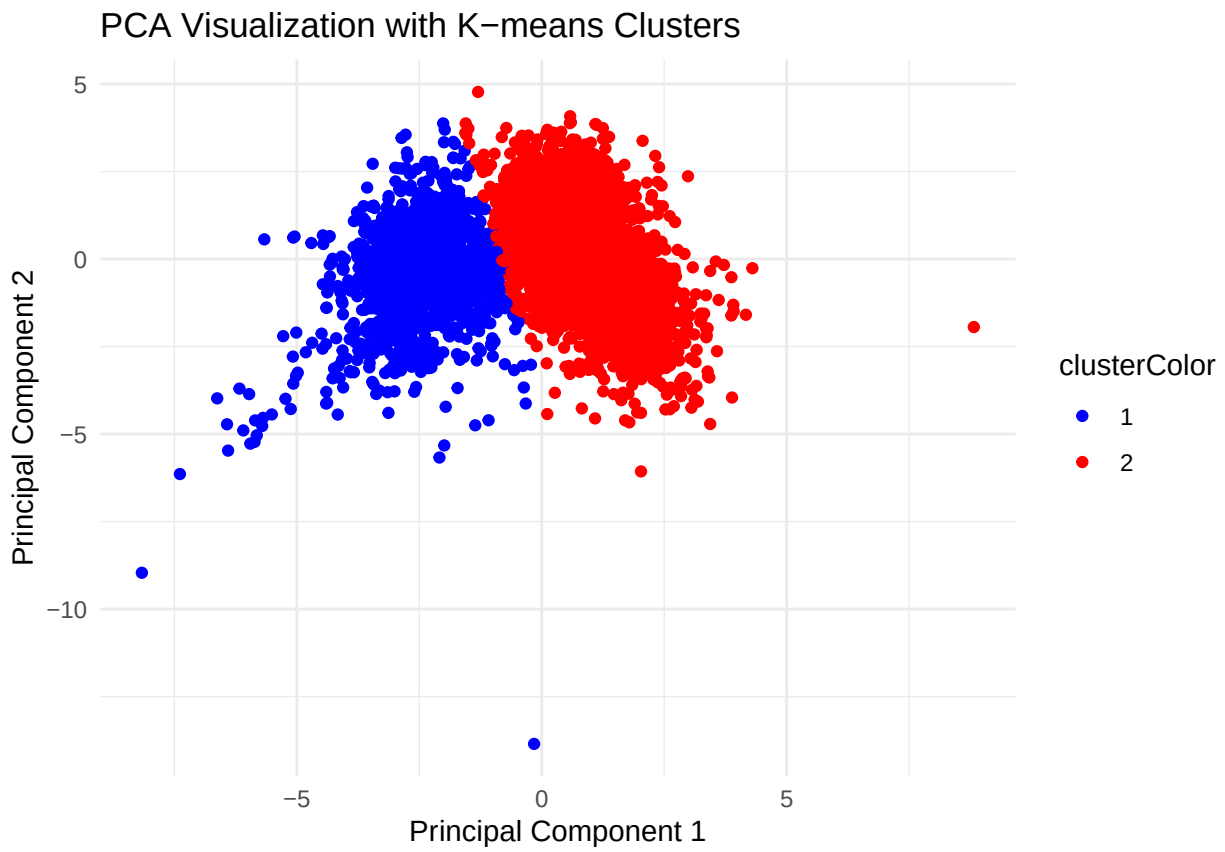
```
# Perform K-means clustering on the PCA-transformed data
kmeansPCAResultColor = kmeans(pcaDataColor[, 1:2], centers = 2, nstart = 20)
pcaDataColor$clusterColor = as.factor(kmeansPCAResultColor$cluster)

# Create a confusion matrix to compare the actual color with the cluster assignments
confusion_matrix_pca <- table(pcaDataColor$color, pcaDataColor$clusterColor)
print(confusion_matrix_pca)
```

```
##
##              1    2
##   red     1329   28
##   white     70 3889
```

```
# Calculate the accuracy
correct_labels_pca <- sum(diag(confusion_matrix_pca))  # Sum of diagonal elements
total_labels_pca <- sum(confusion_matrix_pca)  # Sum of all elements
accuracy_pca <- correct_labels_pca / total_labels_pca  # Accuracy

# Print the accuracy
print(paste("Accuracy: ", round(accuracy_pca, 4)))
```

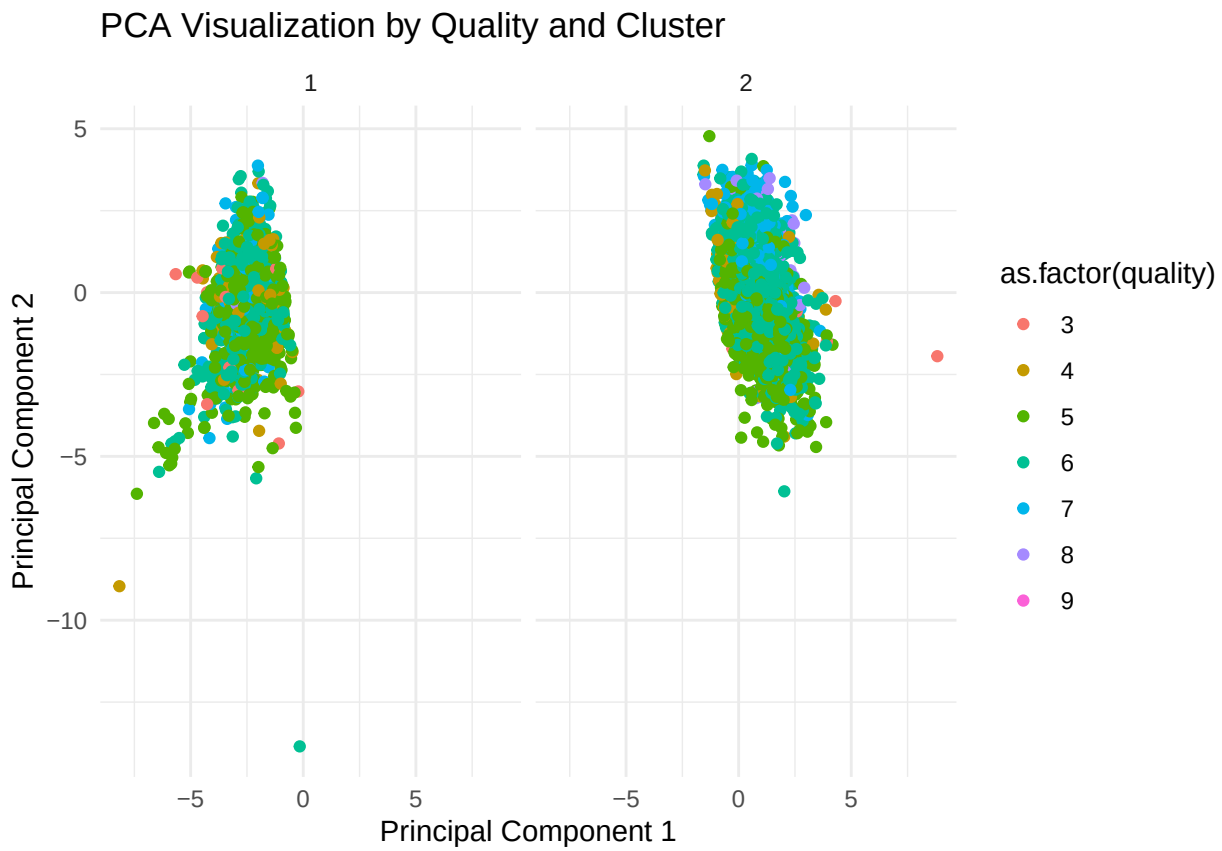
```
## [1] "Accuracy:  0.9816"
```

```r
# Visualization of the PCA with clusters
ggplot(pcaDataColor, aes(x = PC1, y = PC2, color = clusterColor)) +
  geom_point() +
  labs(title = "PCA Visualization with K-means Clusters", x = "Principal Component 1", y = "Principal C
  scale_color_manual(values = c("1" = "blue", "2" = "red")) +
  theme_minimal()
```



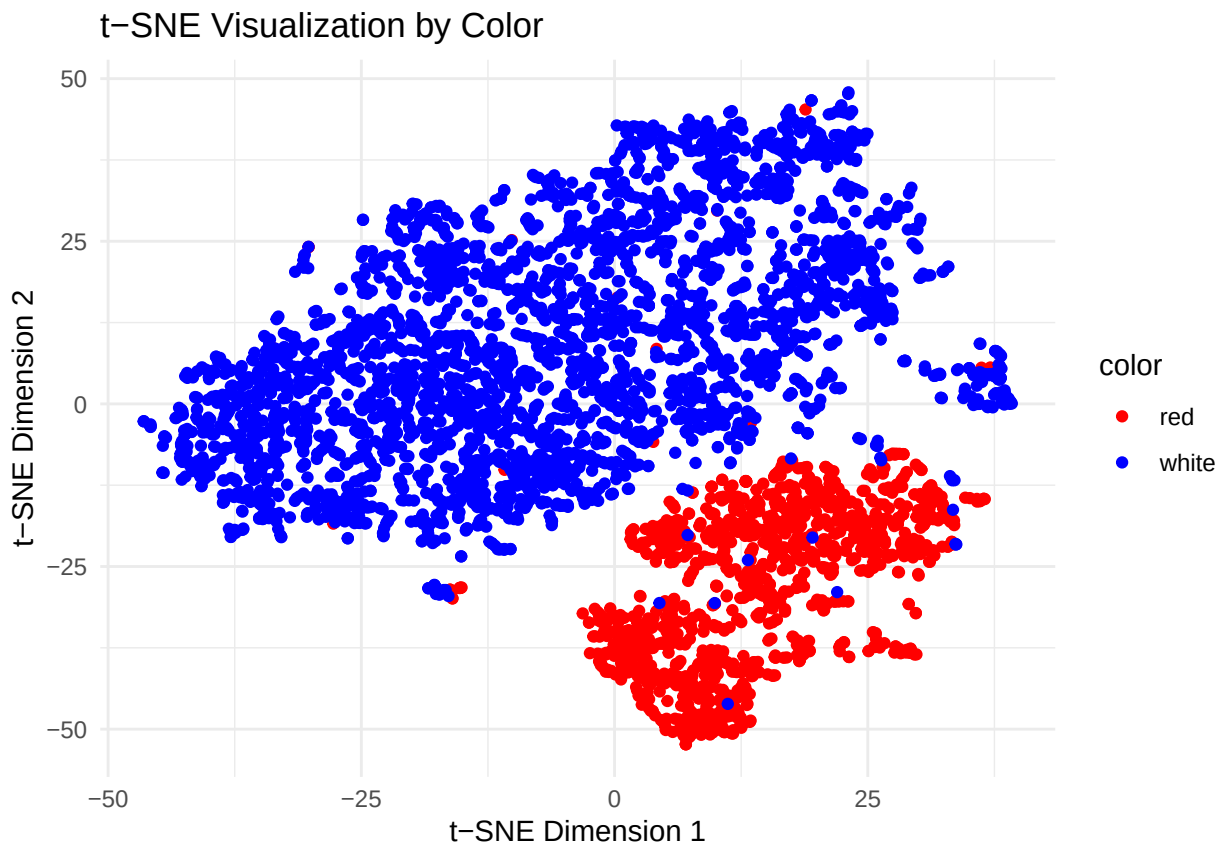
We then tried to see if PCA can predict the wine quality correctly. Judging by our graph, it does a pretty poor job on both clusters. Wine quality is likely much more subjective than if the wine is red vs white based on its chemical components!

```r
# Add quality to the data frame for visualization
pcaDataQuality = data.frame(PC1 = pcaResult$x[, 1], PC2 = pcaResult$x[, 2],
                            quality = wine$quality, cluster = pcaDataColor$clusterColor)

ggplot(pcaDataQuality, aes(x = PC1, y = PC2, color = as.factor(quality))) +
  geom_point() +
  facet_wrap(~ cluster) +
  labs(title = "PCA Visualization by Quality and Cluster", x = "Principal Component 1", y = "Principal
  theme_minimal()
```



Next we ran tSNE. Perhaps the linear PCA summary is not very good and is misleading us, and tSNE can help us out here with its nonlinear dimensonality reduction. We decided to do this in R to keep this problem all on the same file (hopefully this is a good idea!). There is more of a defined boundary in TSNE than PCA. There are a few more whites that are miscalssified as red than reds misclassified as white.

```
tsneResult = Rtsne(wineSubsetScaled, dims = 2)
tsneData = data.frame(TSNE_1 = tsneResult$Y[, 1], TSNE_2 = tsneResult$Y[, 2], color = wine$color)

ggplot(tsneData, aes(x = TSNE_1, y = TSNE_2, color = color)) +
  geom_point() +
  labs(title = "t-SNE Visualization by Color", x = "t-SNE Dimension 1", y = "t-SNE Dimension 2") +
  scale_color_manual(values = c("white" = "blue", "red" = "red"))+
  theme_minimal()
```



We than ran K means on TSNE and calculate the confusion matrix. It seems to do a good with the reds, but there quite a few misclassified whites! At first glance it looks like PCA does a much better job with accuracy, as the calculated tSNE accruacy is 5.44%

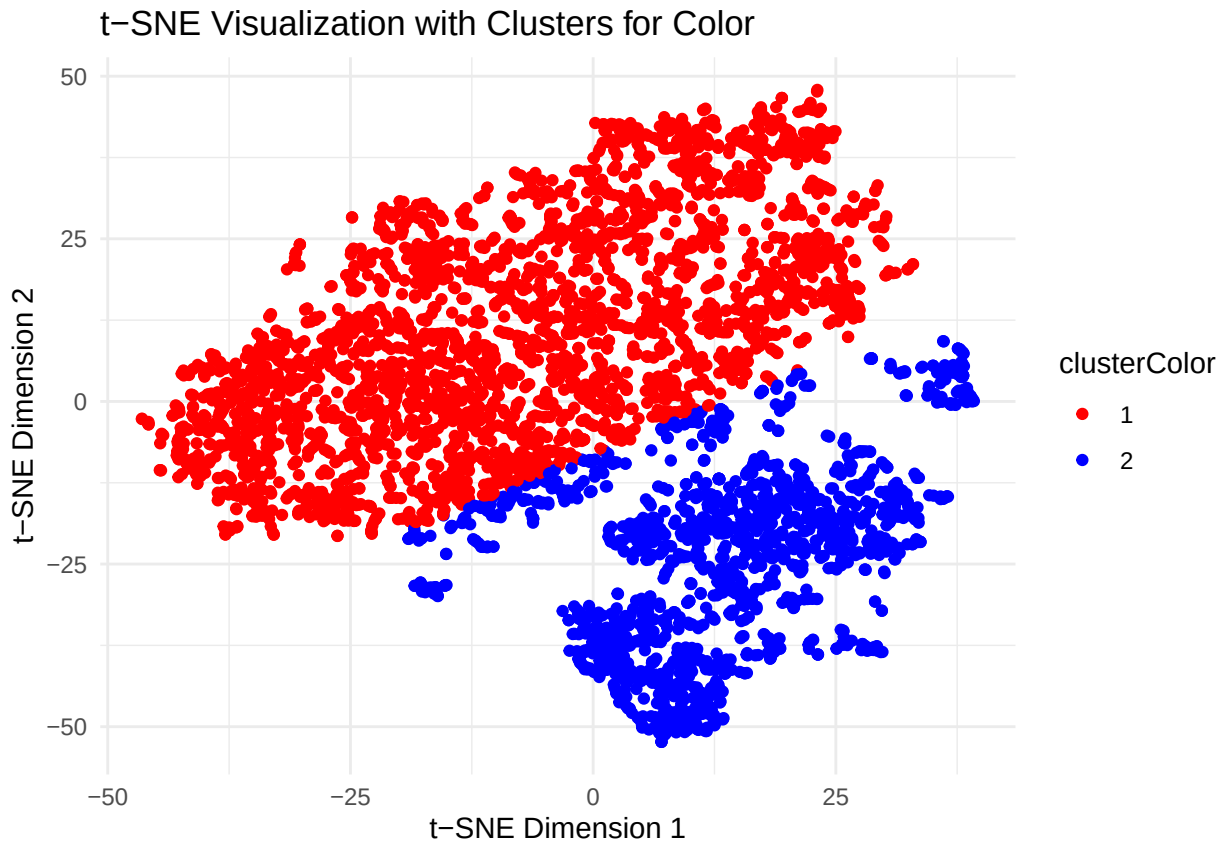
```
K = 2
kmeansTSNEResultColor = kmeans(tsneData[, 1:2], centers = K, nstart = 20)
tsneData$clusterColor = as.factor(kmeansTSNEResultColor$cluster)

# Create a confusion matrix
confusion_matrix_tsne <- table(tsneData$color, tsneData$clusterColor)
print(confusion_matrix_tsne)
```

```
##
##              1    2
##   red       13 1344
##   white   3577  382
```

```
# Calculate the accuracy
correct_labels_tsne <- sum(diag(confusion_matrix_tsne))  # Sum of the diagonal elements
total_labels_tsne <- sum(confusion_matrix_tsne)  # Sum of all elements
accuracy_tsne <- correct_labels_tsne / total_labels_tsne  # Accuracy

# Print the accuracy
print(paste("Accuracy: ", round(accuracy_tsne, 4)))
```

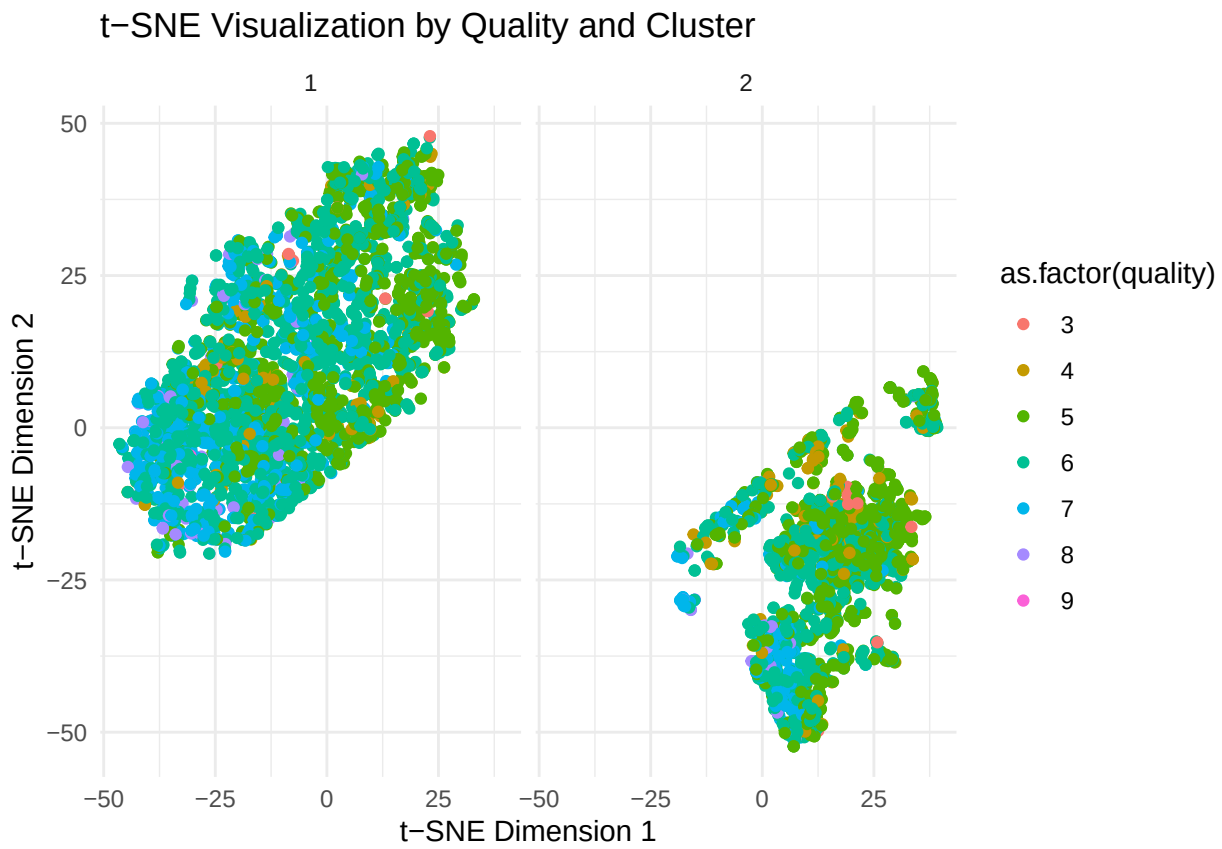
```
## [1] "Accuracy:  0.0743"
```

```r
# Visualize t-SNE with K-means clusters
ggplot(tsneData, aes(x = TSNE_1, y = TSNE_2, color = clusterColor)) +
  geom_point() +
  labs(title = "t-SNE Visualization with Clusters for Color", x = "t-SNE Dimension 1", y = "t-SNE Dimens
  scale_color_manual(values = c("2" = "blue", "1" = "red"))+
  theme_minimal()
```



TSNE may do a little better job are predicting quality, but it still doesn't look great. The plot below is quite a mess, with the quality all over the place in two clusters. The boundary at the top of the second cluster is quite interesting. It looks like most predicted wine quality is 5 or 6.

```r
tsneDataQuality = data.frame(
  TSNE_1 = tsneResult$Y[, 1],
  TSNE_2 = tsneResult$Y[, 2],
  quality = wine$quality,
  cluster = tsneData$clusterColor
)

ggplot(tsneDataQuality, aes(x = TSNE_1, y = TSNE_2, color = as.factor(quality))) +
  geom_point() +
  facet_wrap(~ cluster) +
  labs(title = "t-SNE Visualization by Quality and Cluster",
       x = "t-SNE Dimension 1", y = "t-SNE Dimension 2") +
  theme_minimal()
```



Overall, I would pick PCA for clustering the wine into reds or whites. Even though PCA is linear in nature, it does a better job at clustering. It is cool that these unsupervised techniques seem to be doing a pretty good job on clustering with very few parameters.