SDS323 Exercises 1

Conner Callahan, Kate Patrick, Nick Romanow

Flights at ABIA

```
library(ggplot2)
library(mosaic)
```

```
## Loading required package: dplyr

##
## 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

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggstance

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix
```

```
## 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.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##
## 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: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
```

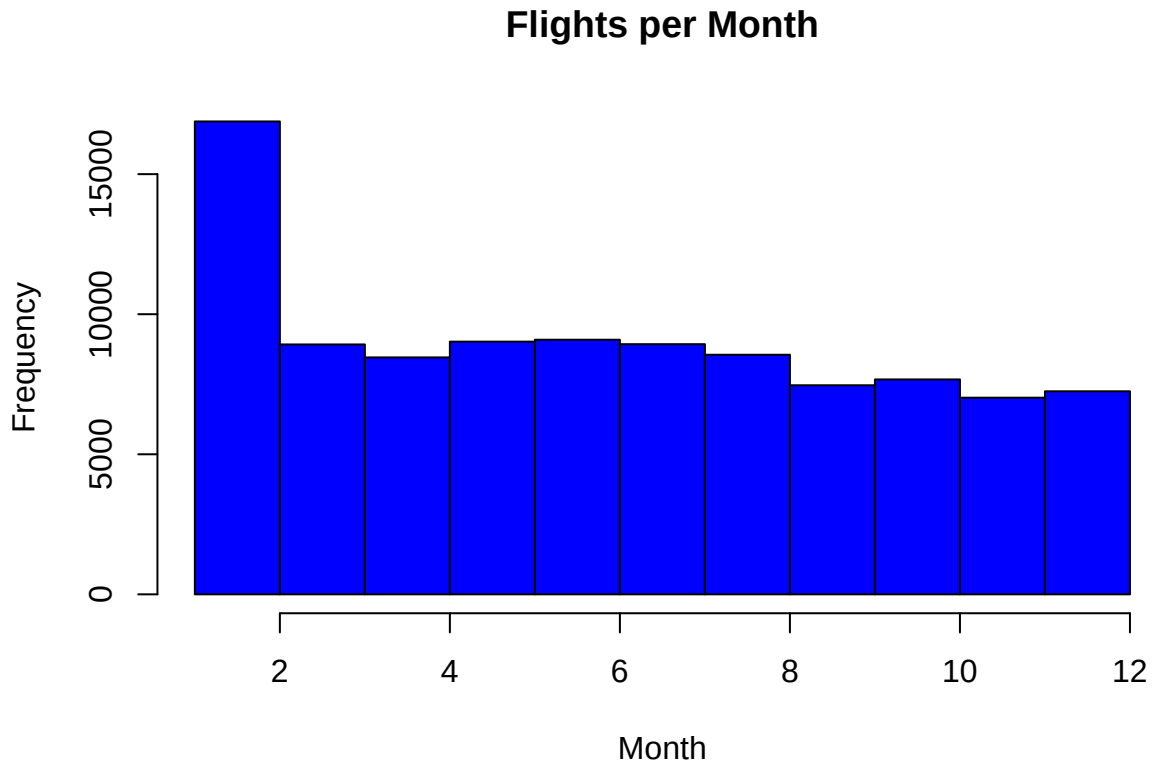
```
library(tidyverse)
```

```
## -- Attaching packages ----------- tidyverse 1.3.0 --

## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts -------------- tidyverse_conflicts() --
## x mosaic::count()            masks dplyr::count()
## x purrr::cross()             masks mosaic::cross()
## x mosaic::do()               masks dplyr::do()
## x tidyr::expand()            masks Matrix::expand()
## x dplyr::filter()            masks stats::filter()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x dplyr::lag()               masks stats::lag()
## x tidyr::pack()              masks Matrix::pack()
## x mosaic::stat()             masks ggplot2::stat()
## x mosaic::tally()            masks dplyr::tally()
## x tidyr::unpack()            masks Matrix::unpack()
```

```r
ABIA <- read.csv("~/GitHub/SDS323/data/ABIA.csv")

#question: Which airport origin minimizes departure delays the best?
#question: What day of the week and month have the most flights from Austin?

fav_stats(ABIA$AirTime)
```

```
##  min Q1 median  Q3 max     mean       sd     n missing
##    3 38    105 142 402 99.81212 58.46554 97659    1601
```

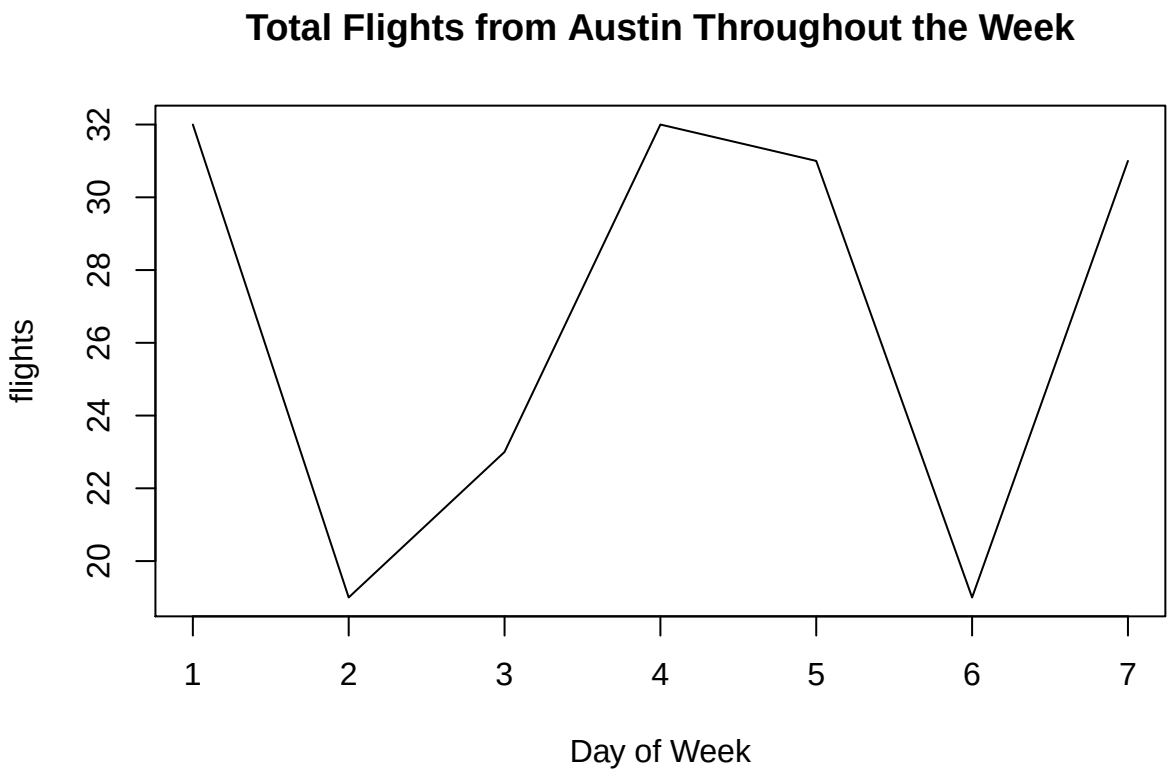
```r
#Most flights occur in January from Austin, otherwise there is a similar average amount of flights ever
hist(ABIA$Month, breaks=12, main="Flights per Month", xlab="Month", col="blue")
```



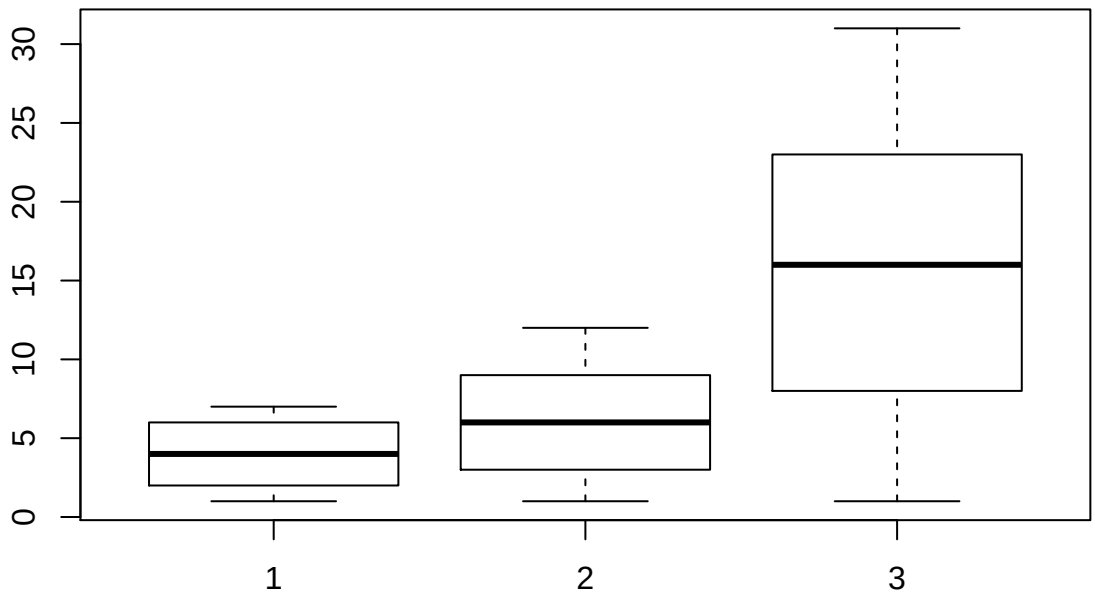

```r
flights_total = ABIA %>%
  filter(Origin == 'AUS') %>%
  group_by(Month) %>%
  summarize(flights = count(Origin))
```

```r
#Most flights occur during the middle of the week or on the weekends.
weekly_total = ABIA %>%
  filter(DepDelay >= '0') %>%
  group_by(DayOfWeek) %>%
  summarize(flights = count(Origin))
```

```
plot(weekly_total, type='l', main="Total Flights from Austin Throughout the Week", xlab="Day of Week")
```

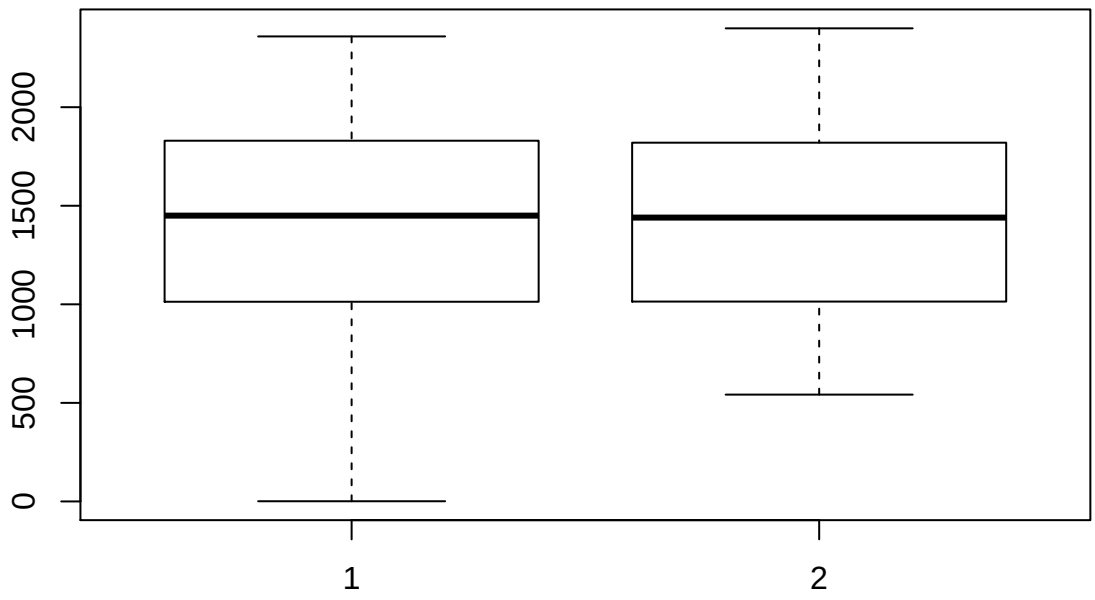



```
#People tend to fly the most during the middle of the week, middle of the year, and middle of the month
boxplot(ABIA$DayOfWeek, ABIA$Month, ABIA$DayofMonth)
```




```
#Most flights occur at the end of the year or during the summer months.
```

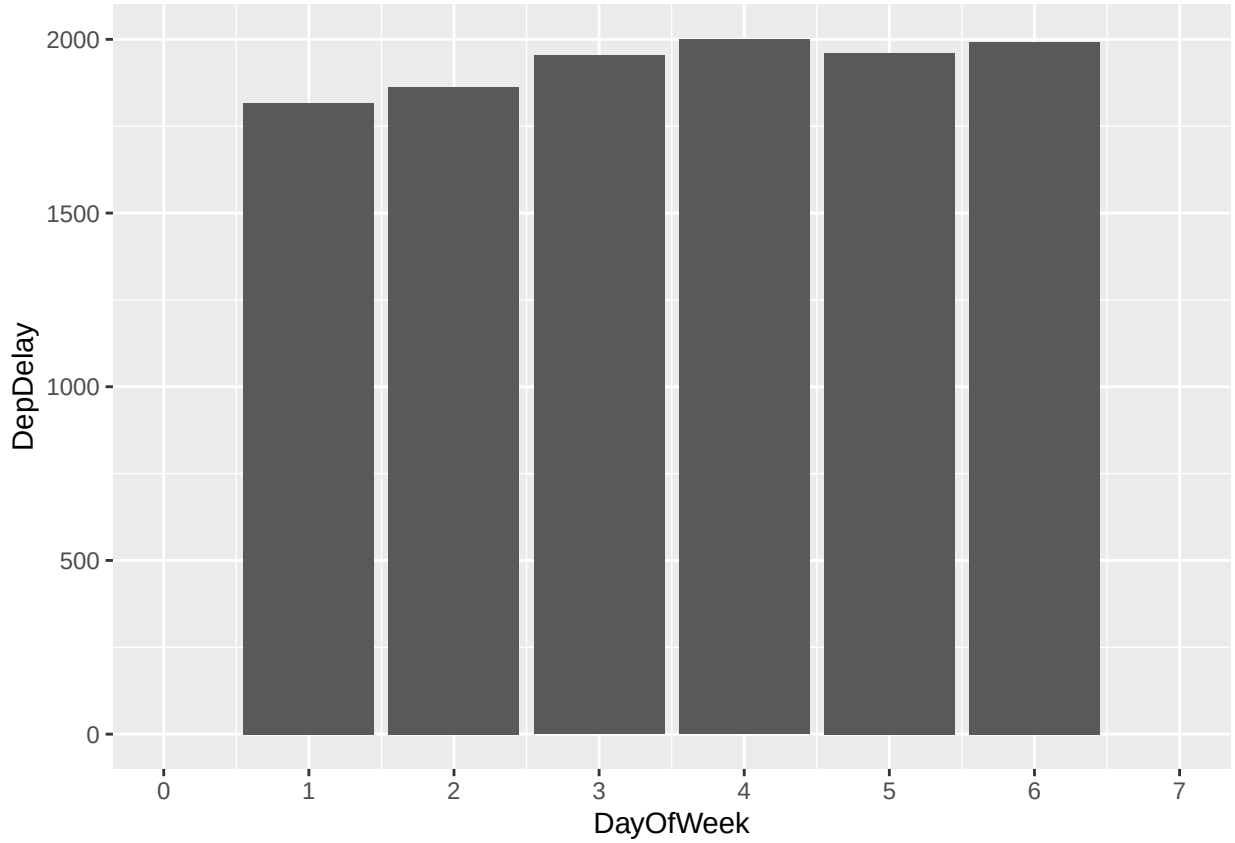
```
#On average, the median Arrival Time is about equal to the median scheduled Arrival time. There is more
newdata <- subset(ABIA, Origin == "AUS",
                  select=Year:DepDelay)
boxplot(newdata$ArrTime, newdata$CRSArrTime)
```




```
#On average, most flight delays occur on Sunday or Monday.
ggplot(newdata, aes(x=DayOfWeek, y=DepDelay)) +
  geom_bar(stat="identity")+
  scale_y_continuous(limits = c(0, 2000), breaks = seq(0, 2000, by = 500))+
  scale_x_continuous(limits = c(0, 7), breaks = seq(0, 7, by = 1))
```

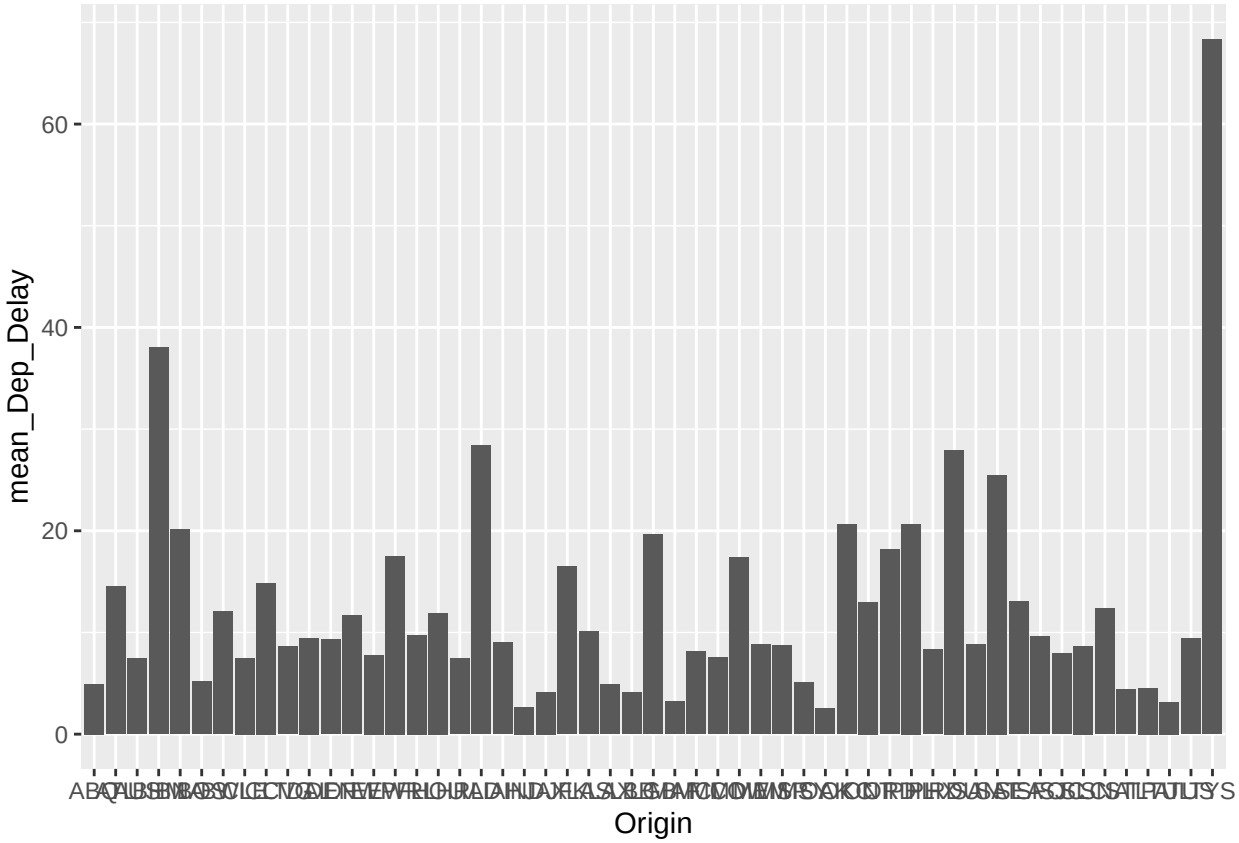
```
## Warning: Removed 27609 rows containing missing values (position_stack).
```

```
## Warning: Removed 21337 rows containing missing values (geom_bar).
```




```
#This data set shows the average delays in departure from all of the airports.
mean_Dep_Delay <- ABIA %>%
  group_by(Origin) %>%
  summarize(mean_Dep_Delay = mean(DepDelay, na.rm=TRUE))
```

```
#Almost all of the airports have a less than 40 minute departure delay.
ggplot(data = mean_Dep_Delay)+
  geom_bar(mapping = aes(x=Origin, y=mean_Dep_Delay),
           position="dodge", stat="identity")
```



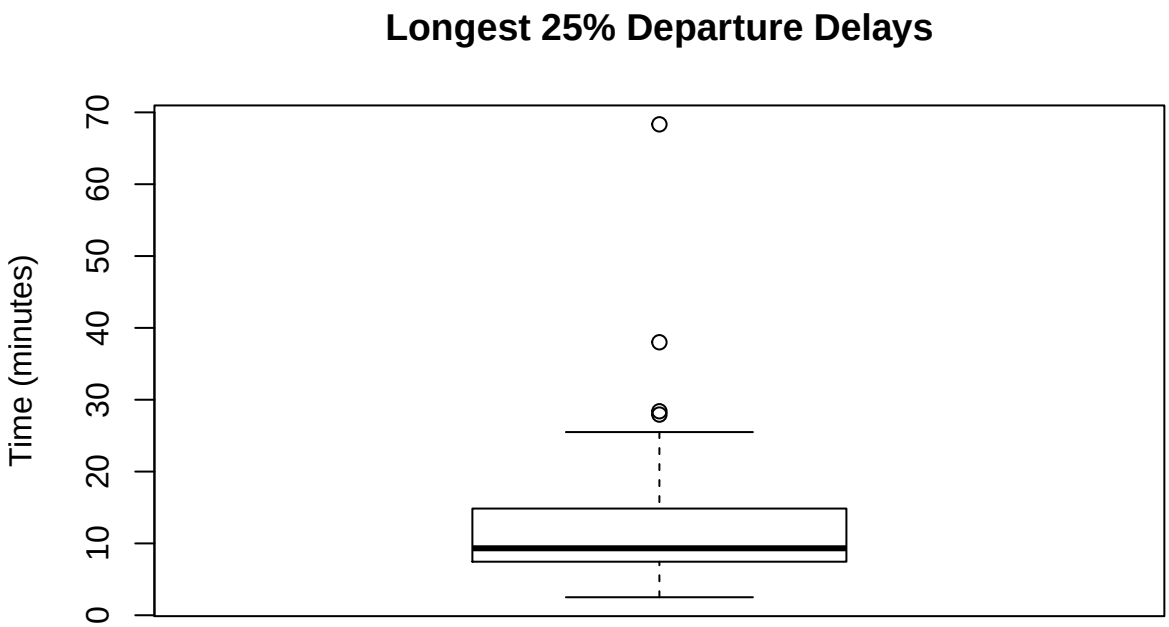

```
fav_stats(mean_Dep_Delay$mean_Dep_Delay)
```

```
##       min       Q1   median       Q3      max     mean       sd  n missing
##  2.508475 7.454039 9.323996 14.85258 68.33333 12.51935 10.70252 53       0
```

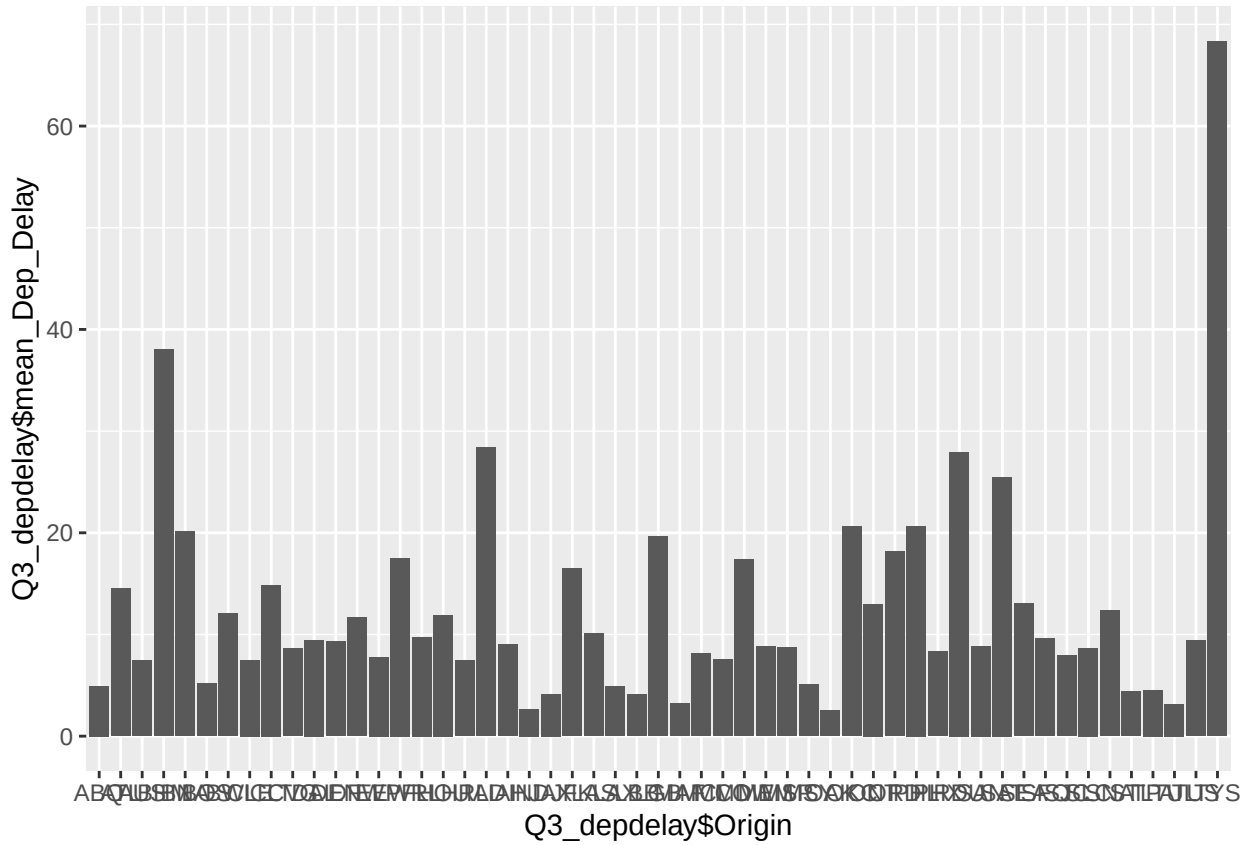
```
#airport's origin with the worst departure delay time from Austin
Q3_depdelay <- subset(mean_Dep_Delay, mean_Dep_Delay = 15:69)

fav_stats(Q3_depdelay$mean_Dep_Delay)
```

```
##       min       Q1   median       Q3      max     mean       sd  n missing
##  2.508475 7.454039 9.323996 14.85258 68.33333 12.51935 10.70252 53       0
```

```
boxplot(Q3_depdelay$mean_Dep_Delay, main="Longest 25% Departure Delays",
        ylab="Time (minutes)")
```



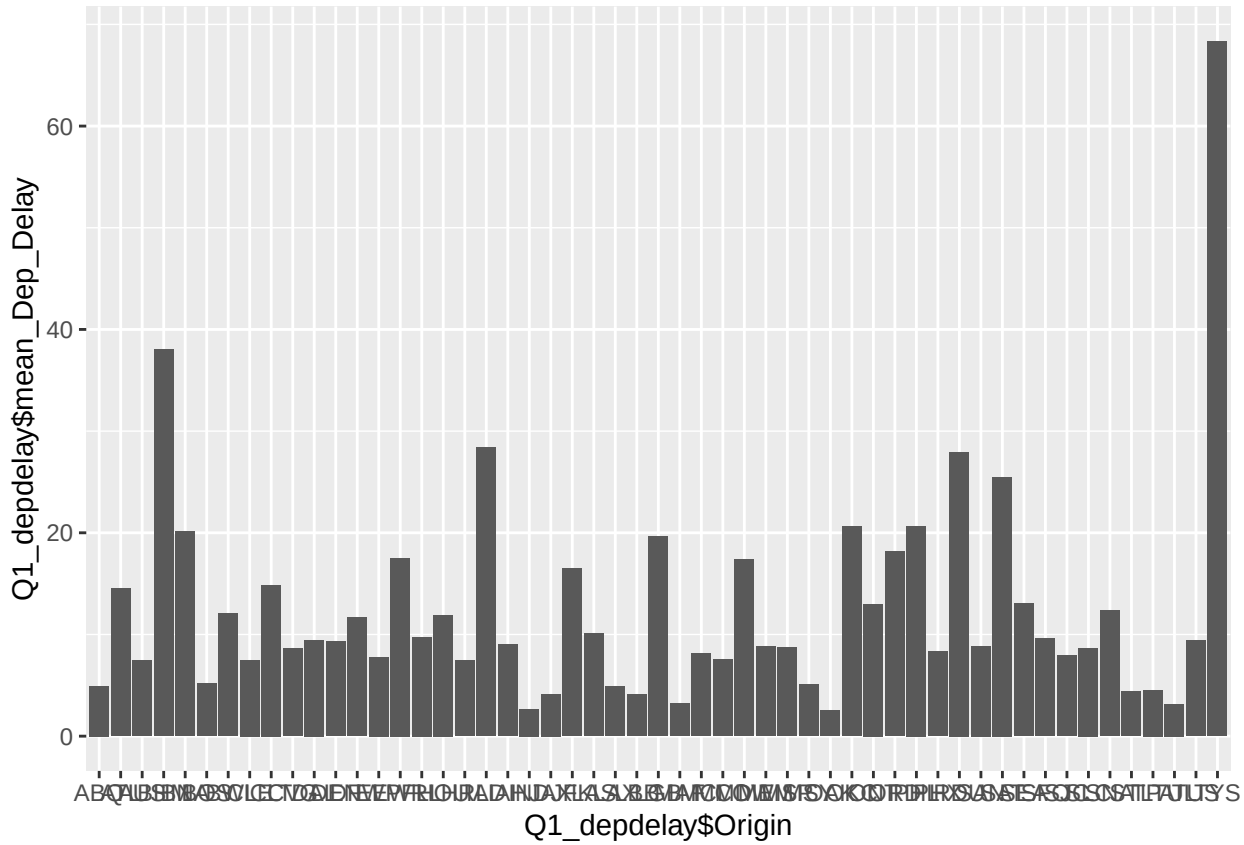

```
#It can be seen from this graph that the airport with the most departure delays is TYS, with a time of
ggplot(Q3_depdelay, aes(x=Q3_depdelay$Origin, y=Q3_depdelay$mean_Dep_Delay)) +
  geom_bar(position="stack", stat="identity")
```




```
#best departure delay from Austin
Q1_depdelay <- subset(mean_Dep_Delay, mean_Dep_Delay = 2:8)


fav_stats(Q1_depdelay$mean_Dep_Delay)
```

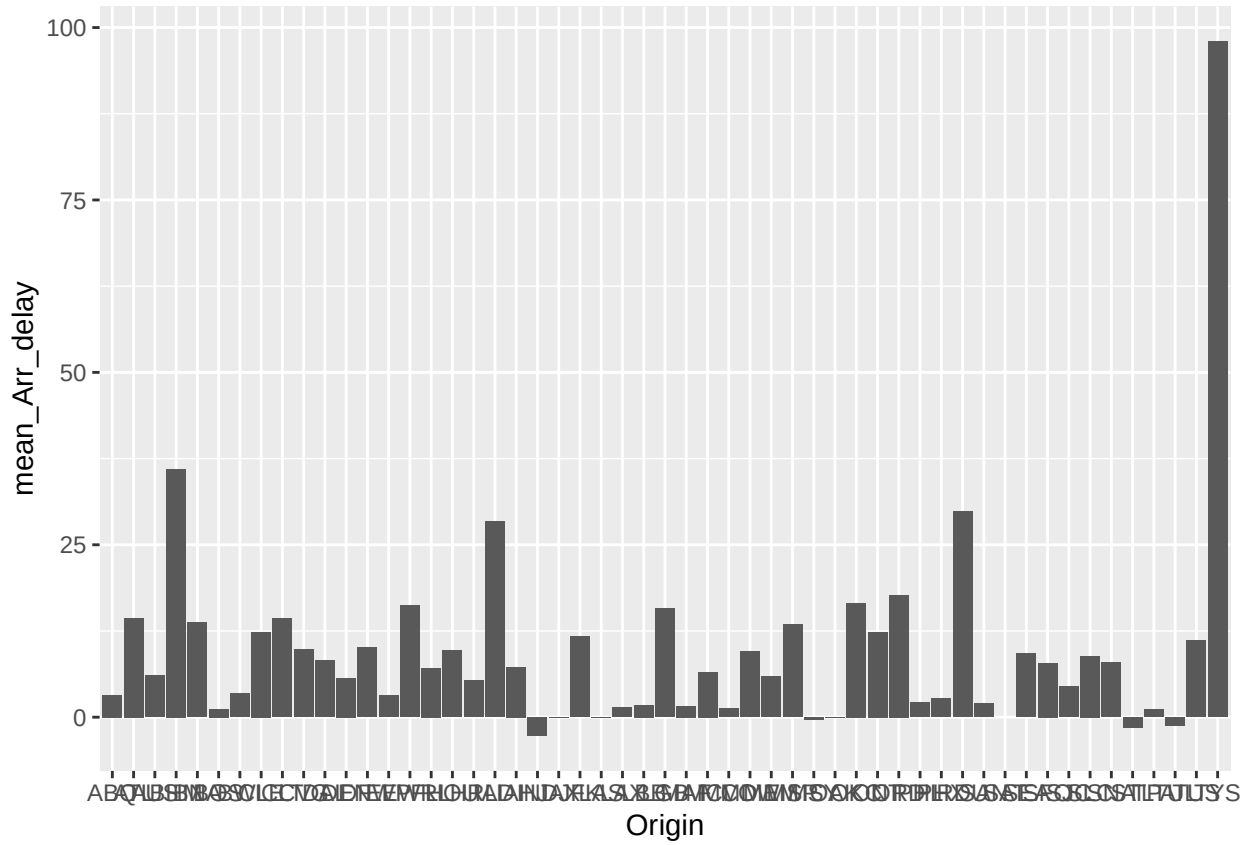
```
##       min       Q1   median       Q3      max     mean       sd  n missing
##  2.508475 7.454039 9.323996 14.85258 68.33333 12.51935 10.70252 53       0
```

```
#It can be seen from this graph that the airport with the lowest average departure delay is OAK with 2.
ggplot(Q1_depdelay, aes(x=Q1_depdelay$Origin, y=Q1_depdelay$mean_Dep_Delay)) +
  geom_bar(position="stack", stat="identity")
```




```
#Almost all of the flights have an arrival delay of less than 40 minutes

mean_Arr_delay <- ABIA %>%
  group_by(Origin) %>%
  summarize(mean_Arr_delay = mean(ArrDelay, na.rm=TRUE))

ggplot(data = mean_Arr_delay)+
  geom_bar(mapping = aes(x=Origin, y=mean_Arr_delay),
           position="dodge", stat="identity")
```

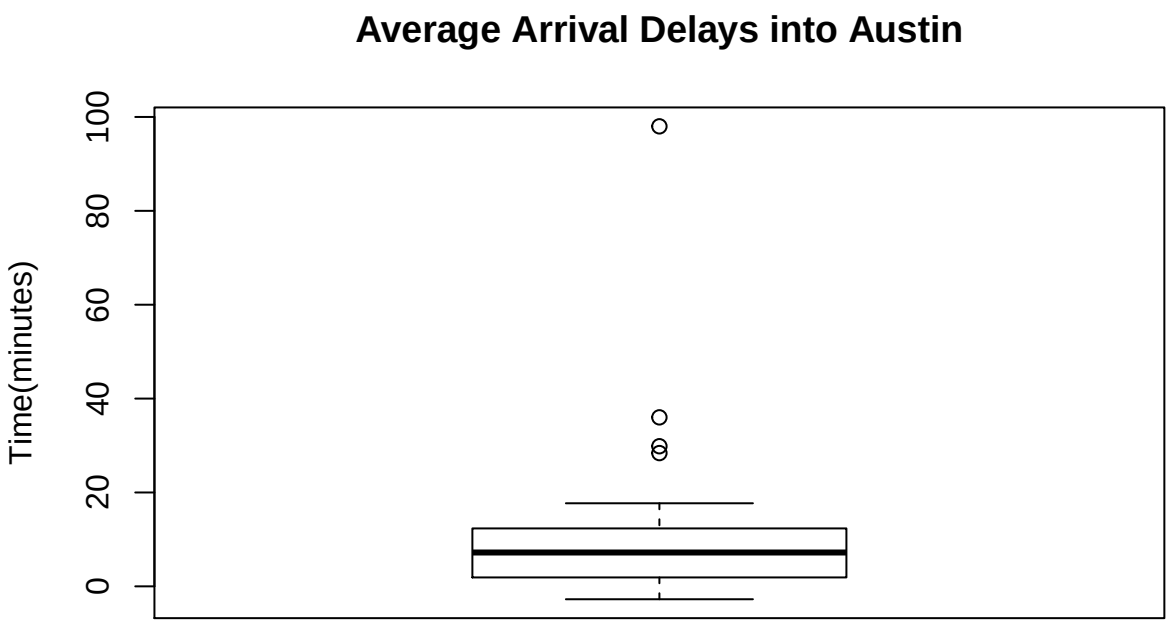
```
## Warning: Removed 1 rows containing missing values (geom_bar).
```




```
#airport's origin with worst arrival delay to Austin
mean_Arr_delay <- ABIA %>%
  group_by(Origin) %>%
  summarize(mean_Arr_delay = mean(ArrDelay, na.rm=TRUE))


fav_stats(mean_Arr_delay$mean_Arr_delay)
```

```
##        min       Q1   median     Q3 max     mean      sd  n missing
##  -2.748837 1.952593 7.210284 12.339  98 9.812812 14.7817 52       1
```

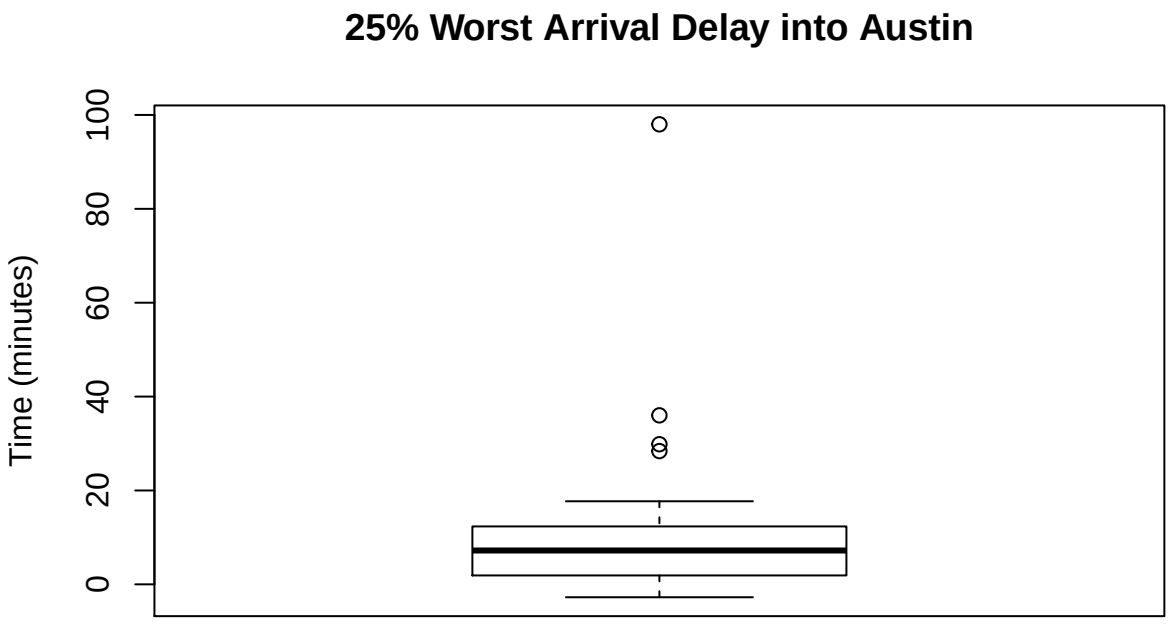
```
boxplot(mean_Arr_delay$mean_Arr_delay, main="Average Arrival Delays into Austin", ylab="Time(minutes)")
```




```
Q3_Arrdelay <- subset(mean_Arr_delay, mean_Arr_delay = 12.339:98)

fav_stats(Q3_Arrdelay$mean_Arr_delay)
```

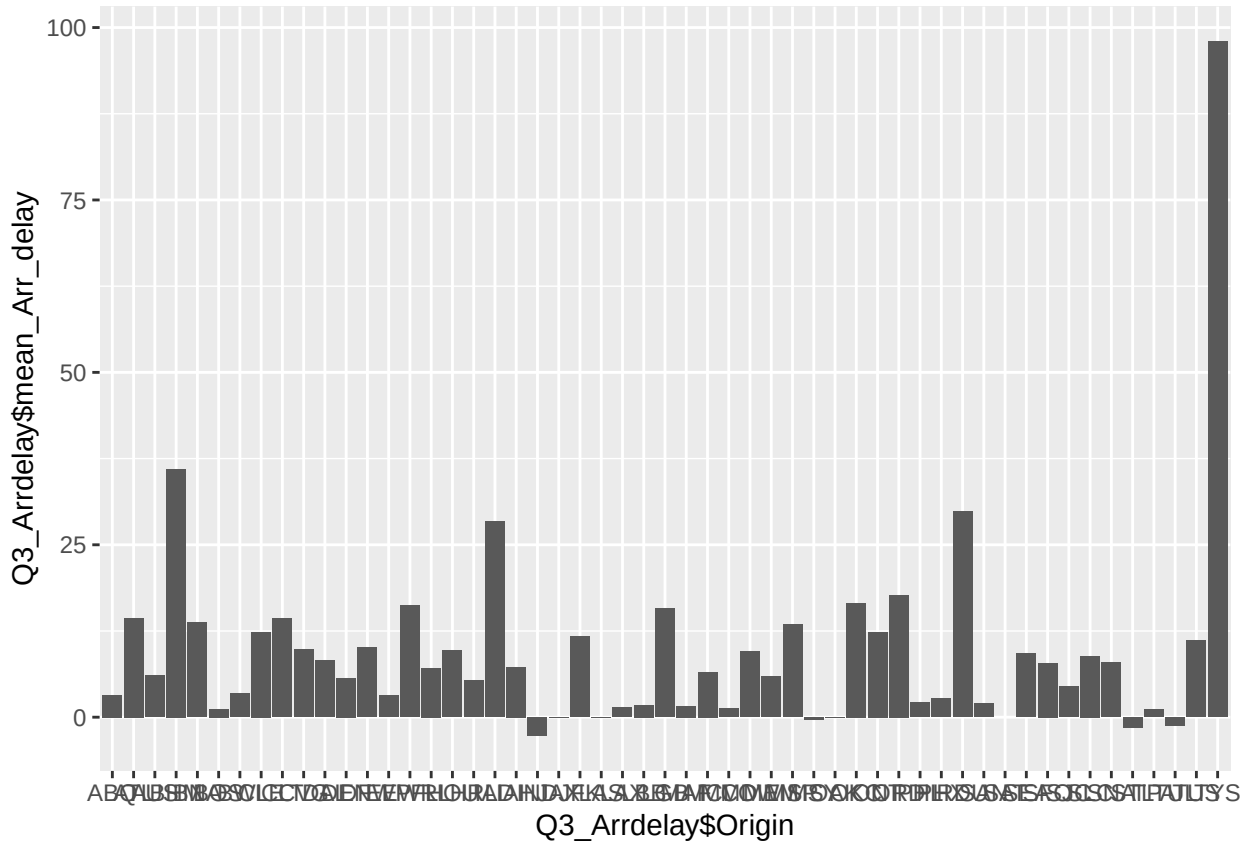
```
##        min       Q1   median     Q3 max     mean      sd  n missing
##  -2.748837 1.952593 7.210284 12.339  98 9.812812 14.7817 52       1
```

```
boxplot(Q3_Arrdelay$mean_Arr_delay, main = "25% Worst Arrival Delay into Austin", ylab="Time (minutes)"
```




```
#From this graph, it can be seen that the flights coming from TYS have the most arrival delays into the
ggplot(Q3_Arrdelay, aes(x=Q3_Arrdelay$Origin, y=Q3_Arrdelay$mean_Arr_delay)) +
  geom_bar(position="stack", stat="identity")
```

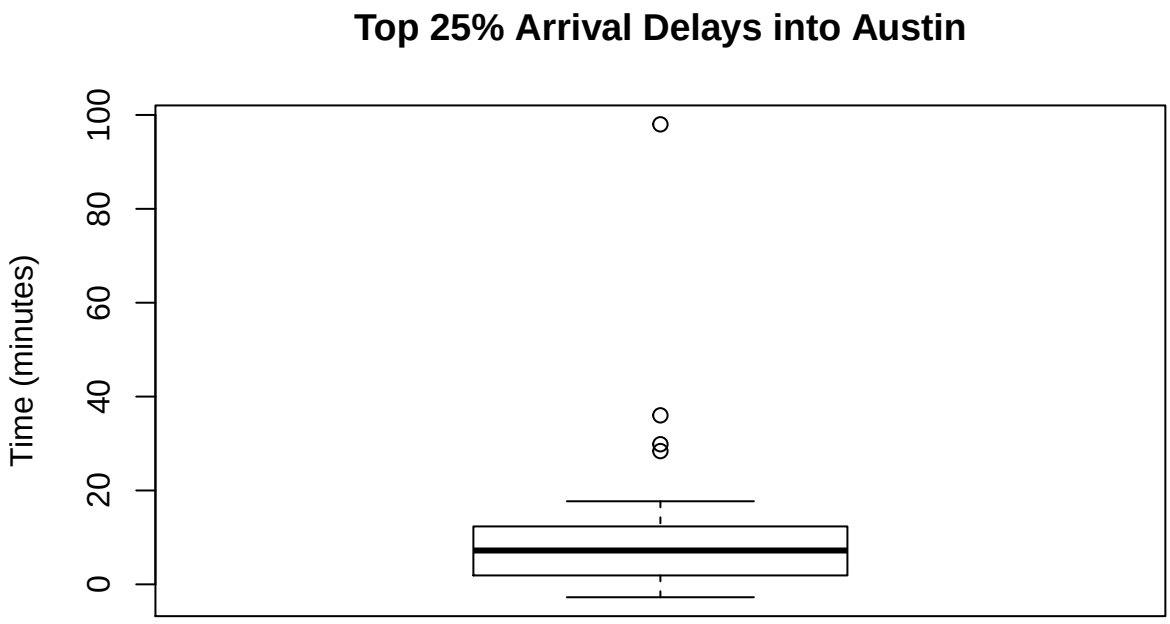
```
## Warning: Removed 1 rows containing missing values (position_stack).
```




```
#best arrival time into Austin
Q1_Arrdelay <- subset(mean_Arr_delay, mean_Arr_delay = -2.748837:1.952593 )

fav_stats(Q1_Arrdelay$mean_Arr_delay)
```

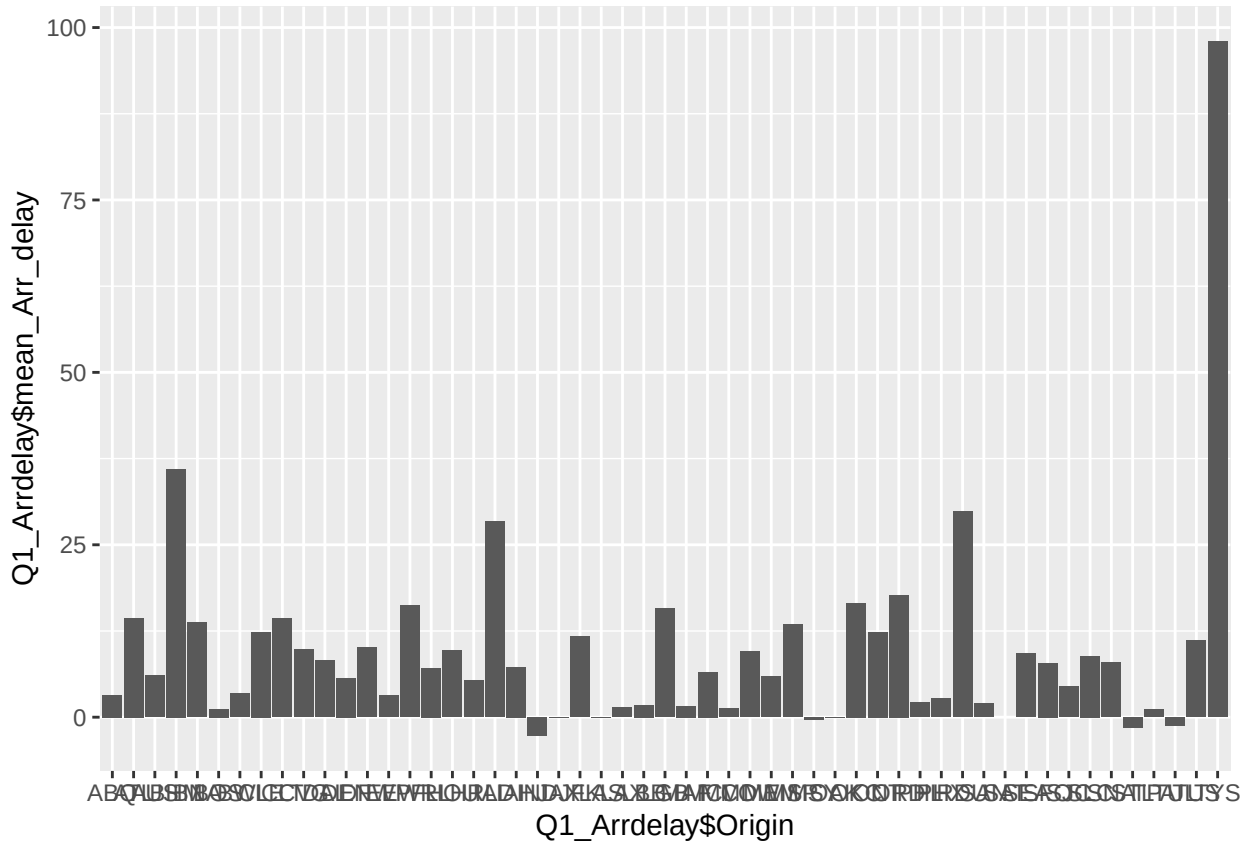
```
##        min       Q1   median     Q3 max     mean      sd  n missing
##  -2.748837 1.952593 7.210284 12.339  98 9.812812 14.7817 52       1
```

```
boxplot(Q1_Arrdelay$mean_Arr_delay, main="Top 25% Arrival Delays into Austin", ylab="Time (minutes)")
```




```
#From this graph, it can be seen that flights coming from IND have the least amount of arrival delays,
ggplot(Q1_Arrdelay, aes(x=Q1_Arrdelay$Origin, y=Q1_Arrdelay$mean_Arr_delay)) +
  geom_bar(position="stack", stat="identity")
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```

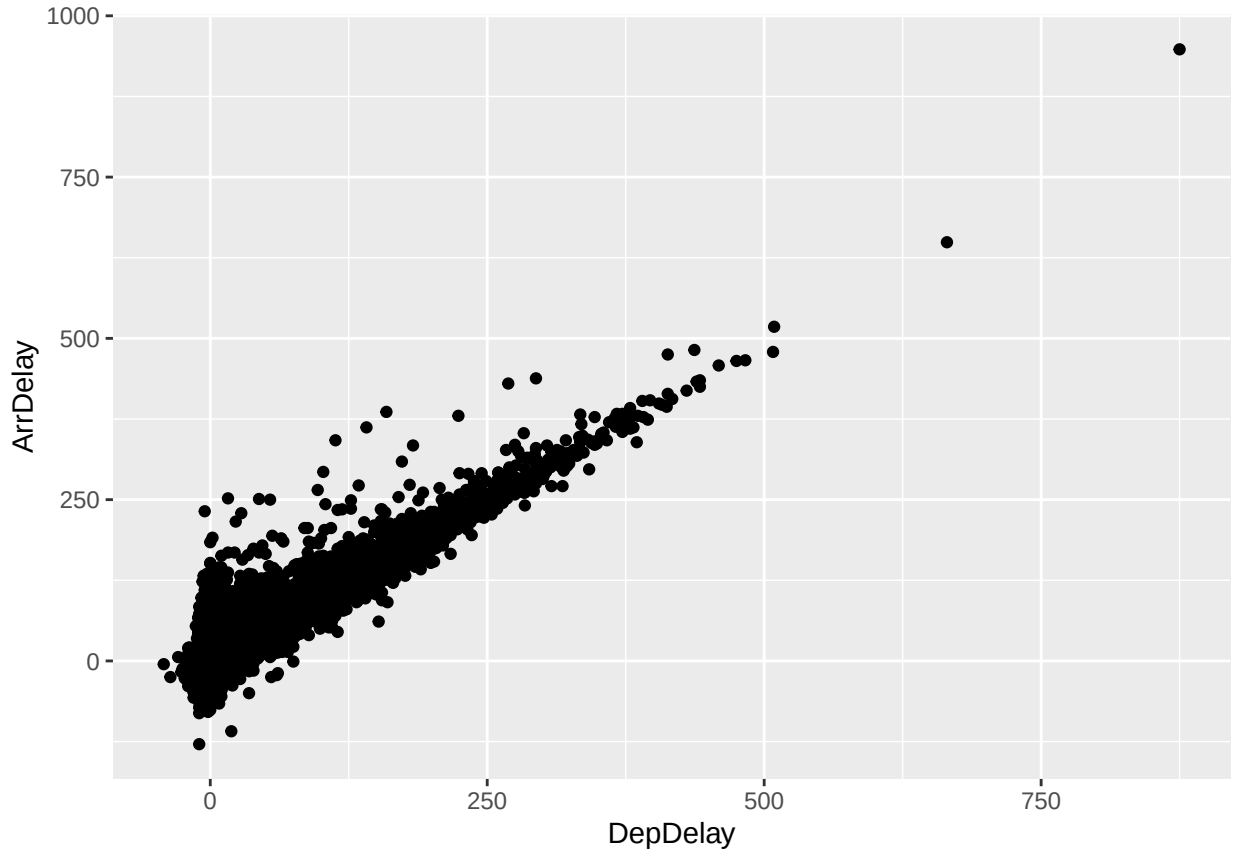



```
#There is a high correlation between a late arrival time and a late departure delay.
cor.test(~ABIA$DepDelay+ABIA$ArrDelay, na.rm=TRUE)
```

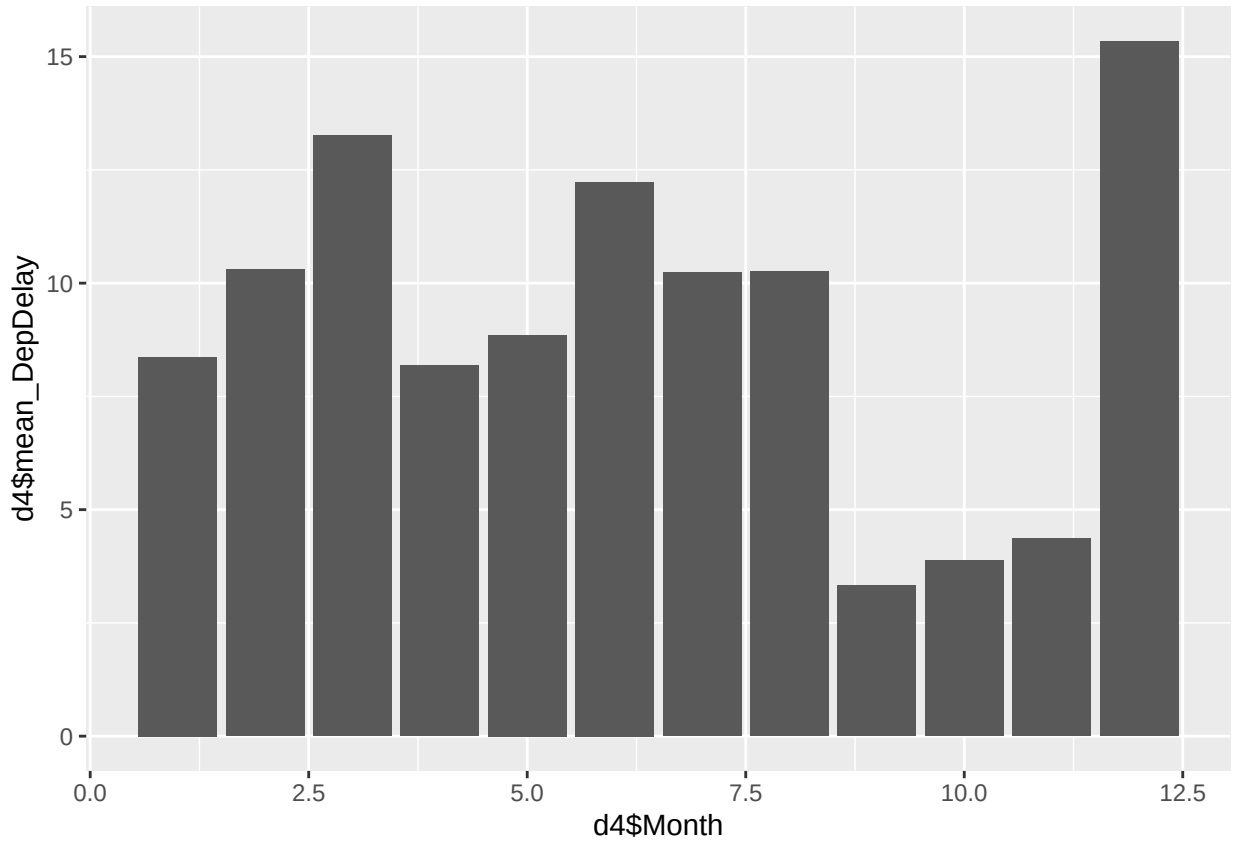
```
##
##  Pearson's product-moment correlation
##
## data:  ABIA$DepDelay and ABIA$ArrDelay
## t = 805.39, df = 97657, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9314551 0.9330965
## sample estimates:
##       cor
## 0.9322806
```

```
ggplot(data = ABIA)+
  geom_point(mapping=aes(x=DepDelay, y=ArrDelay))
```

```
## Warning: Removed 1601 rows containing missing values (geom_point).
```

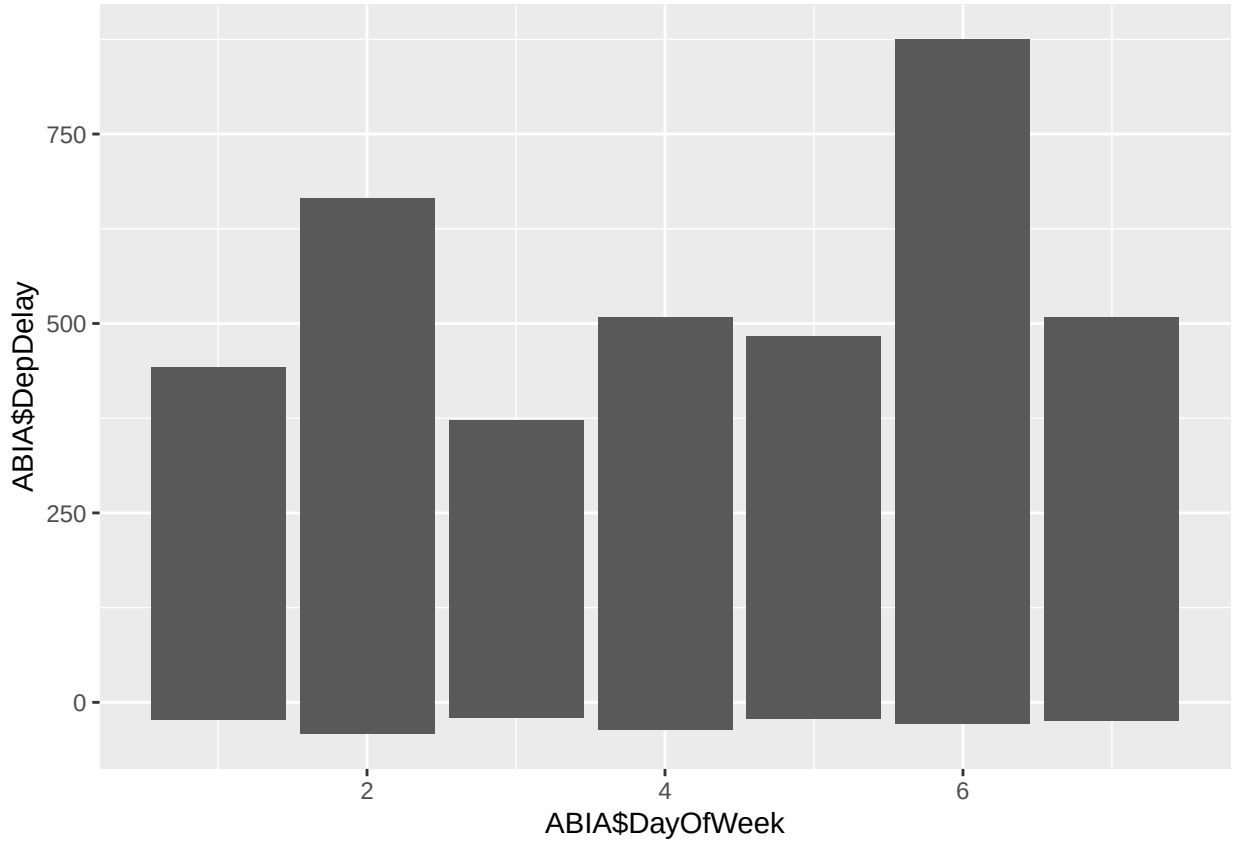



```
#There are far more departure delays occuring in December than in any other month.
d4 <- ABIA %>%
  group_by(Month) %>%
  summarize(mean_DepDelay = mean(DepDelay,na.rm=TRUE))


ggplot(d4, aes(x=d4$Month, y=d4$mean_DepDelay))+
  geom_bar(position="dodge", stat="identity")
```




```
#percentage of departure delays every day of the week: it can be seen that there are more departure del

d6 <- ABIA %>%
  group_by(DayOfWeek) %>%
  summarize(delay_pct = sum(Origin=='yes')/n())



ggplot(ABIA, aes(x=ABIA$DayOfWeek, y=ABIA$DepDelay))+
  geom_bar(position="dodge", stat="identity")
```

```
## Warning: Removed 1413 rows containing missing values (geom_bar).
```




```
#There is a low correlation between the day of the week and departure delays.
cor_test(~ABIA$DayOfWeek+ABIA$DepDelay)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ABIA$DayOfWeek and ABIA$DepDelay
## t = 2.8393, df = 97845, p-value = 0.004523
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.002810915 0.015341447
## sample estimates:
##         cor
## 0.009076537
```

Regression Practice - Creatine Levels

1. The expected Creatinine clearance rate of a 55-year old is 113.7.

```
creatinine <- read.csv("~/GitHub/SDS323/data/creatinine.csv")
library(mosaic)
library(tidyverse)

lml = lm(creatclear ~ age, data = creatinine)
coef(lml)
```

```
## (Intercept)         age
## 147.8129158  -0.6198159
```

```
147.8+(55*-0.62)
```

```
## [1] 113.7
```

2. Creatinine clearance rate drops by .62 for every additional year of age.

```
library(mosaic)
library(tidyverse)

lml = lm(creatclear ~ age, data = creatinine)
coef(lml)
```

```
## (Intercept)         age
## 147.8129158  -0.6198159
```

Green Buildings

Opening file, Header, Packages

```
filename<-"greenbuildings.csv"
setwd("~/GitHub/SDS323/data")
greenbuildings <- read.csv(filename)
head(greenbuildings)
```

```
##   CS_PropertyID cluster   size empl_gr  Rent leasing_rate stories age renovated
## 1        379105       1 260300    2.22 38.56        91.39      14  16         0
## 2        122151       1  67861    2.22 28.57        87.14       5  27         0
## 3        379839       1 164848    2.22 33.31        88.94      13  36         1
## 4         94614       1  93372    2.22 35.00        97.04      13  46         1
## 5        379285       1 174307    2.22 40.69        96.58      16   5         0
## 6         94765       1 231633    2.22 43.16        92.74      14  20         0
##   class_a class_b LEED Energystar green_rating net amenities cd_total_07
## 1       1       0    0          1            1   0         1        4988
## 2       0       1    0          0            0   0         1        4988
## 3       0       1    0          0            0   0         1        4988
## 4       0       1    0          0            0   0         0        4988
## 5       1       0    0          0            0   0         1        4988
## 6       1       0    0          0            0   0         1        4988
##   hd_total07 total_dd_07 Precipitation  Gas_Costs Electricity_Costs
## 1         58        5046         42.57 0.01370000        0.02900000
## 2         58        5046         42.57 0.01373149        0.02904455
## 3         58        5046         42.57 0.01373149        0.02904455
## 4         58        5046         42.57 0.01373149        0.02904455
## 5         58        5046         42.57 0.01373149        0.02904455
## 6         58        5046         42.57 0.01373149        0.02904455
##   cluster_rent
## 1        36.78
## 2        36.78
```

```
## 3        36.78
## 4        36.78
## 5        36.78
## 6        36.78
```

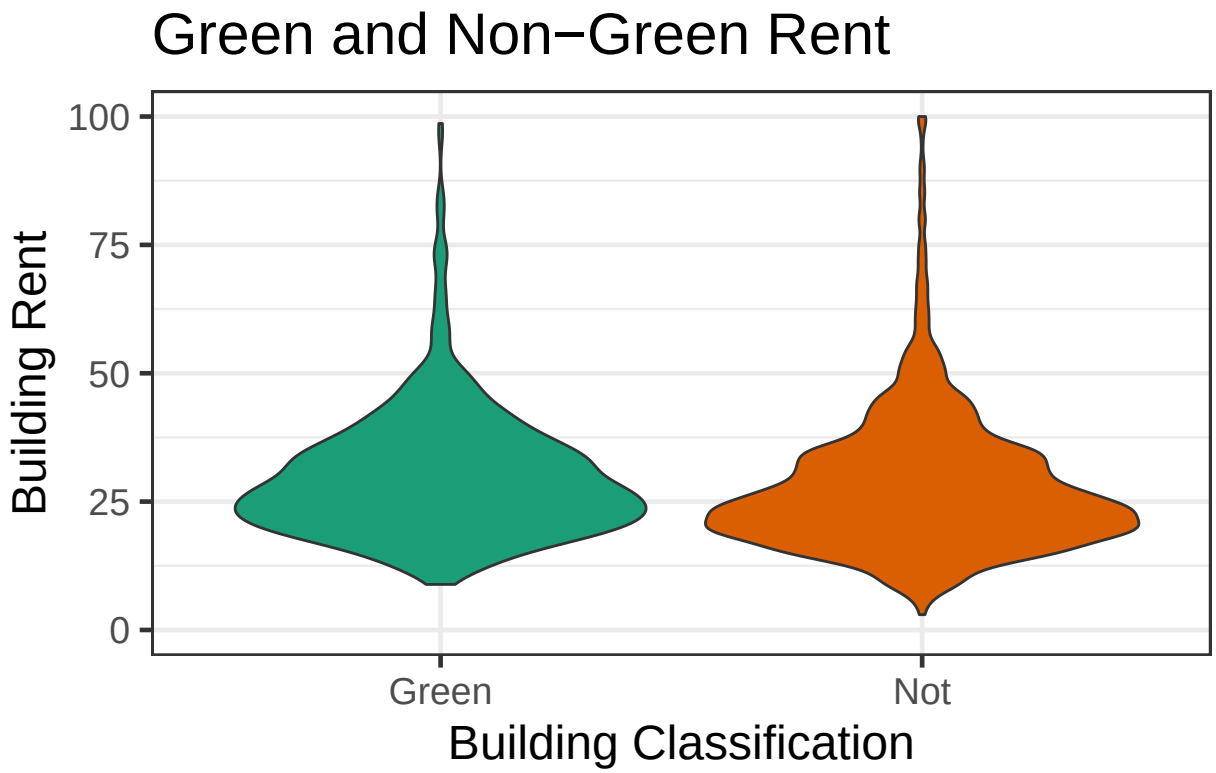
```
library(tidyverse)
library(knitr)
library(mosaic)
```

While the developer's "data guru" correctly finds that green buildings charge a higher rent, he failed to account for several considerations that may call into question his findings. Both size and age are variables that could confound the green rent data, and the type of green rating a building attains may also affect rent. Finally, the data guru forgot to account for the time value of money when he predict they could recover their upfront costs in 7.7 years.

The rent for green buildings is statisitically higher than non-green buildings. This can be seen in the graph and linear regression below. The green_rating variable is statistically significant when regressed by itself against rent. We are 95% confident that the true coefficient for a green building is between 57¢ and $2.93 worth of rent increase per square foot. This would result somewhere between 142,500 and 732,500 in additional revenue per year, with the estimated cost of building a green building at $5,000,000 up front.

```
#cleaning
greenbuildings$green_clean <- ifelse(greenbuildings$green_rating==1,"Green","Not")
#Rent by green or not
ggplot(data = greenbuildings) +
  geom_violin(aes(x=green_clean, y=Rent, fill=green_clean)) +
  ylim(c(0,100)) +
  theme_bw(base_size=18) +
  scale_fill_brewer(palette="Dark2") +
  labs(title="Green and Non-Green Rent",
       y="Building Rent",
       x = "Building Classification",
       caption = "26 values removed") +
  theme(legend.position="none")
```

```
## Warning: Removed 26 rows containing non-finite values (stat_ydensity).
```



26 values removed

```
#model
greenonly_mod <- lm(Rent~green_rating, data = greenbuildings)
summary(greenonly_mod)
```

```
##
## Call:
## lm(formula = Rent ~ green_rating, data = greenbuildings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.287  -9.044  -3.267   5.733 221.733
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   28.2668     0.1775 159.275   <2e-16 ***
## green_rating   1.7493     0.6025   2.903   0.0037 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.07 on 7892 degrees of freedom
## Multiple R-squared:  0.001067,   Adjusted R-squared:  0.0009405
## F-statistic:  8.43 on 1 and 7892 DF,  p-value: 0.003701
```

```
confint(greenonly_mod, 'green_rating', level = .95)
```

```
##                   2.5 %   97.5 %
## green_rating 0.5682584 2.930245
```

```
0.57*250000 #142500
```

```
## [1] 142500
```

```
2.93*250000 #732500
```

```
## [1] 732500
```

However, this increase in rent can be explained by other, more significant, variables than green_rating. We started by regressing rent based on the variables we knew about the developer's new building: size, stories, age, and green_rating. Stories was insignificant, so we removed the variable. In the resulting model, green_rating was still statisitcally insignificant, but both size and age were statistically and practically significant in predicting rent. We ran the boxplots shown below to represent the difference in these variables between green and non-green buildings. Green buildings were typically younger and larger than the non-green buildings in the dataset, which may have resulted in the rents for those buildings being higher based on their size and age rather than their green status.

```
#starting model
base_mod <- lm(Rent~size+stories+age+green_rating, data = greenbuildings)
summary(base_mod)
```

```
##
## Call:
## lm(formula = Rent ~ size + stories + age + green_rating, data = greenbuildings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.539  -9.437  -2.941   5.751 211.447
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.857e+01  3.872e-01  73.784  < 2e-16 ***
## size          5.565e-06  1.016e-06   5.478 4.43e-08 ***
## stories       1.706e-02  2.431e-02   0.702    0.483
## age          -3.616e-02  5.443e-03  -6.644 3.26e-11 ***
## green_rating  2.347e-01  6.126e-01   0.383    0.702
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.89 on 7889 degrees of freedom
## Multiple R-squared:  0.02479,    Adjusted R-squared:  0.0243
## F-statistic: 50.14 on 4 and 7889 DF,  p-value: < 2.2e-16
```

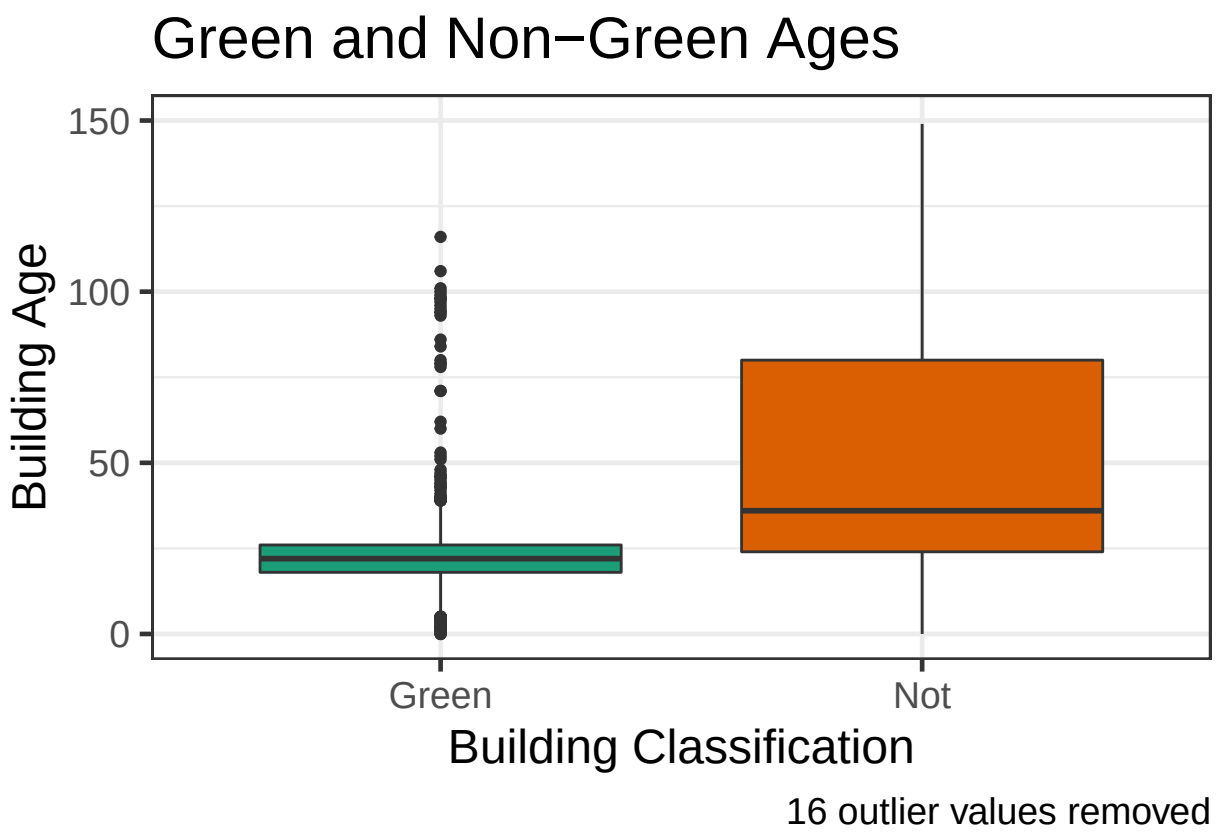
```
#removestories
base_mod <- lm(Rent~size+age+green_rating, data = greenbuildings)
summary(base_mod)
```

```
##
```

```
## Call:
## lm(formula = Rent ~ size + age + green_rating, data = greenbuildings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.977  -9.454  -2.986   5.727 211.018
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.866e+01  3.652e-01  78.480  < 2e-16 ***
## size          6.153e-06  5.762e-07  10.678  < 2e-16 ***
## age          -3.604e-02  5.440e-03  -6.625  3.7e-11 ***
## green_rating  2.118e-01  6.117e-01   0.346    0.729
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.89 on 7890 degrees of freedom
## Multiple R-squared:  0.02473,    Adjusted R-squared:  0.02436
## F-statistic: 66.69 on 3 and 7890 DF,  p-value: < 2.2e-16
```

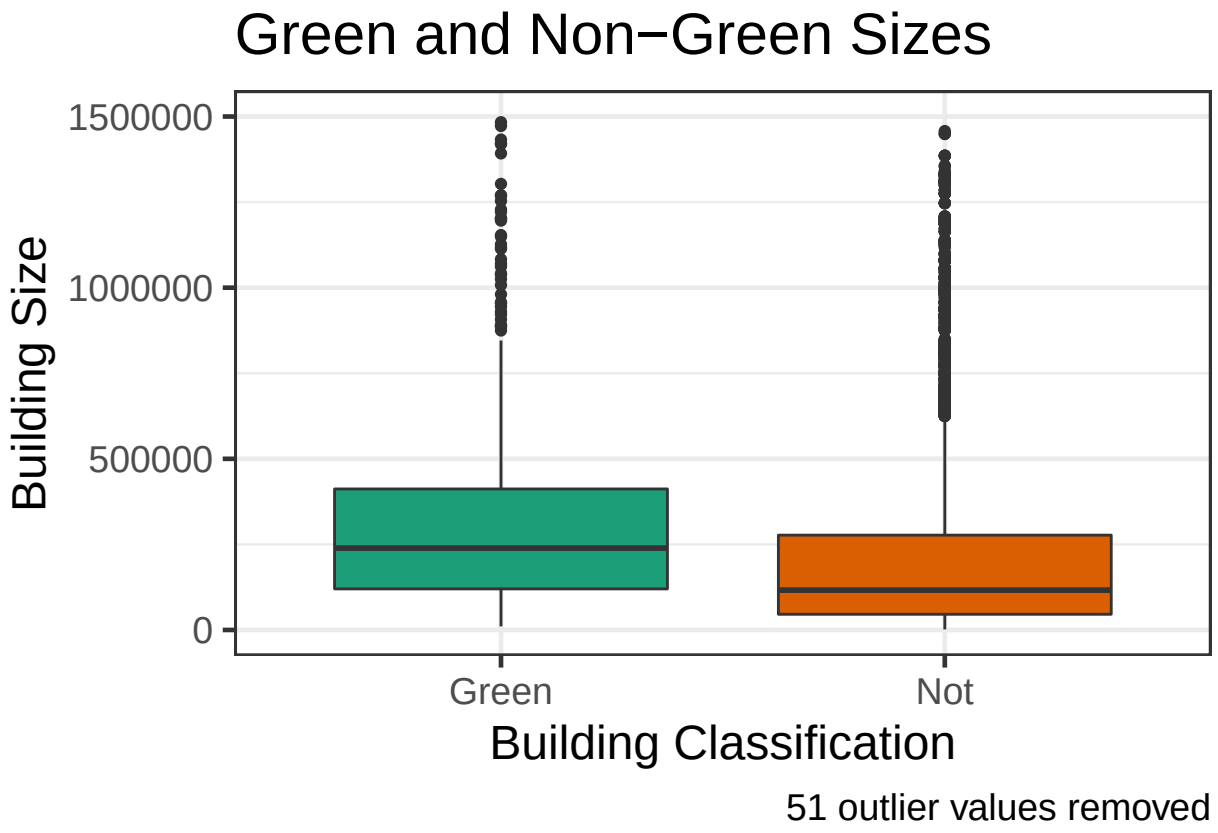
```
#Ages by Green status
ggplot(data = greenbuildings) +
  geom_boxplot(aes(x=green_clean, y=age, fill=green_clean)) +
  ylim(c(0,150)) +
  theme_bw(base_size=18) +
  scale_fill_brewer(palette="Dark2") +
  labs(title="Green and Non-Green Ages",
       y="Building Age",
       x = "Building Classification",
       caption = "16 outlier values removed") +
  theme(legend.position="none")
```

```
## Warning: Removed 16 rows containing non-finite values (stat_boxplot).
```




```
#Sizes by Green status
ggplot(data = greenbuildings) +
  geom_boxplot(aes(x=green_clean, y=size, fill=green_clean)) +
               ylim(c(0,1500000)) +
  theme_bw(base_size=18) +
  scale_fill_brewer(palette="Dark2") +
  labs(title="Green and Non-Green Sizes",
       y="Building Size",
       x = "Building Classification",
       caption = "51 outlier values removed") +
  theme(legend.position="none")
```

```
## Warning: Removed 51 rows containing non-finite values (stat_boxplot).
```



The developer and her data guru also failed to consider the effect of attaining different kinds of green classification. Buildings with an Energystar rating typically charged higher rents than those with LEED green ratings. NO variables were statistically significant when we regressed known variables with green_type against rent. LEED and Energystar rated buildings have different density structures of rent, and this should be taken into consideration when planning the construction of a potentially green building.

```
LEED <- subset(greenbuildings,LEED==1)
Energystar <- subset(greenbuildings,Energystar==1)
#Energystar higher than LEED
summary(LEED$Rent)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   21.12   24.27   29.70   31.53   98.65
```

```
summary(Energystar$Rent)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.87   21.49   28.02   30.08   35.79  138.07
```

```
#model
greenbuildings$green_type <-
  ifelse(greenbuildings$LEED==1,"LEED",
         ifelse(greenbuildings$Energystar==1,"Energystar",NA))
green_mod <- lm(Rent~size+stories+age+green_type, data = greenbuildings, na.rm = TRUE)
```

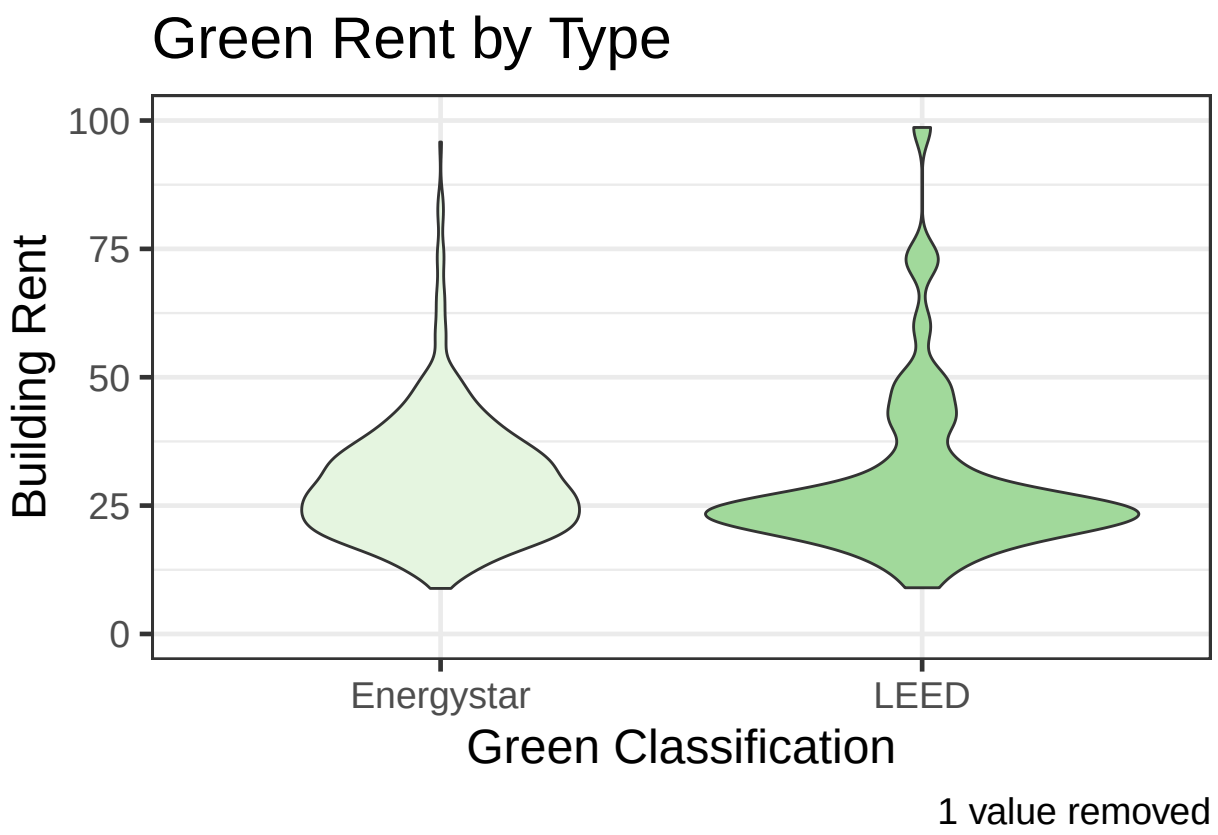
```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'na.rm' will be disregarded
```

```
summary(green_mod)
```

```
##
## Call:
## lm(formula = Rent ~ size + stories + age + green_type, data = greenbuildings,
##     na.rm = TRUE)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.960  -8.205  -2.303   5.416 106.765
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.985e+01  1.088e+00  27.444   <2e-16 ***
## size           -5.535e-07  4.539e-06  -0.122    0.903
## stories        -1.480e-02  9.864e-02  -0.150    0.881
## age             2.561e-02  3.276e-02   0.782    0.435
## green_typeLEED -5.024e-01  1.883e+00  -0.267    0.790
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.98 on 680 degrees of freedom
##   (7209 observations deleted due to missingness)
## Multiple R-squared:  0.001749,   Adjusted R-squared:  -0.004123
## F-statistic: 0.2978 on 4 and 680 DF,  p-value: 0.8794
```

```
#Rent by LEED or Energystar
ggplot(na.omit(greenbuildings)) +
  geom_violin(aes(x=green_type, y=Rent, fill=green_type)) +
  ylim(c(0,100)) +
  theme_bw(base_size=18) +
  scale_fill_brewer(palette="Greens") +
  labs(title="Green Rent by Type",
       y="Building Rent",
       x = "Green Classification",
       caption = "1 value removed") +
  theme(legend.position="none")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_ydensity).
```



Lastly, the data guru forgot to account for the time value of money. The time he predict it would take to recover the cost was arounf 7.7 years, but the actual time would be much longer if you discount at the developer's cost of capital. This is necessary because it demonstrates the opportunity cost of using this capital investment elsewhere, for example, on another development that could potentially be built as a green building.

All in all, the data guru correctly found that green buildings have a higher rent than non-green building in this dataset, but this finding may not be indicative of a causal relationship between green rating and rent. There are other variables with a greater impact on the rent a building can charge, such as age and size of the building. We are not convinced that making the building green would allow the developer to recover her cost by charging a higher rent in a timely enough way.

Milk Pricing Case Study

Opening file and Viewing the header

```
filename<-"milk.csv"
setwd("~/GitHub/SDS323/data")
milk <- read.csv(filename)
head(milk)
```

```
##   price sales
## 1  3.48    15
## 2  3.12    11
## 3  2.95    21
## 4  2.68    30
```

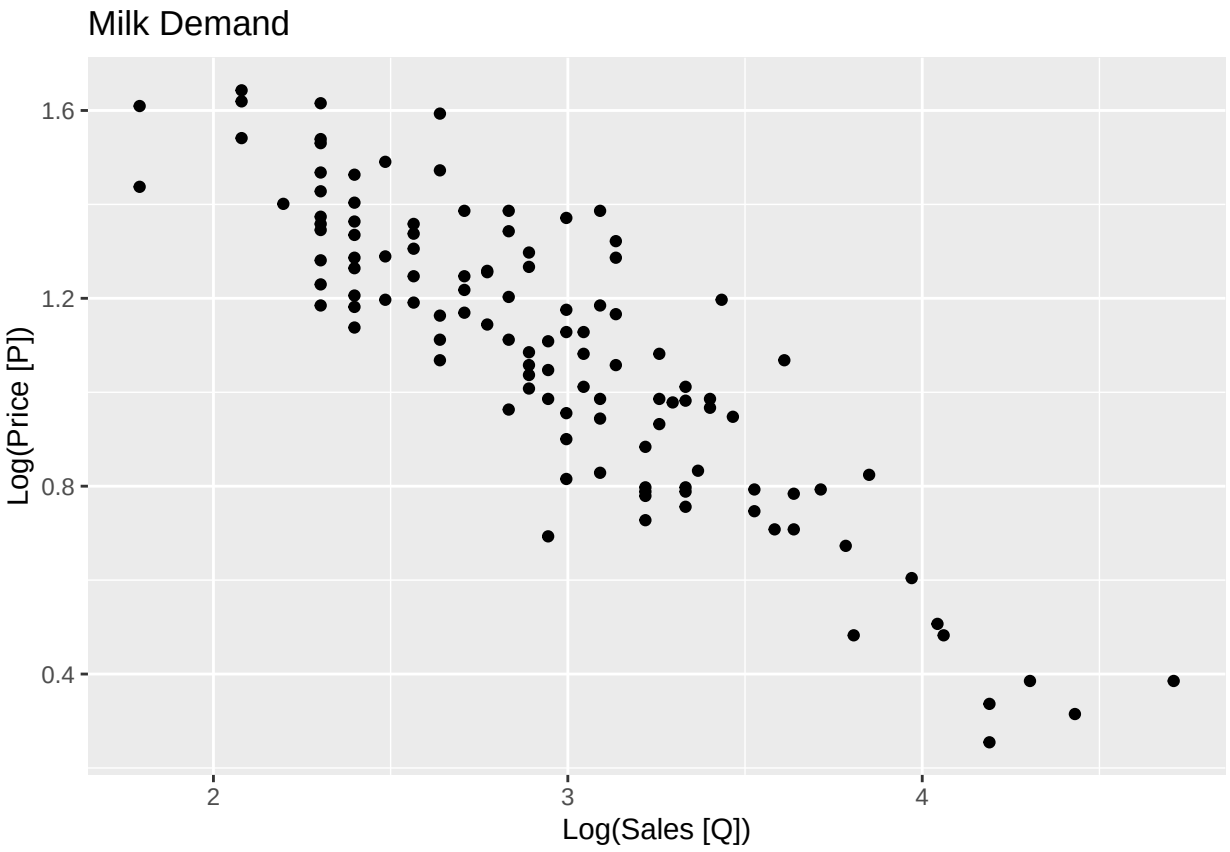
```
## 5  3.62    11
## 6  4.32    11
```

Demand Model

```
milk.demand <- lm(log(sales)~log(price), data = milk)
summary(milk.demand)
```

```
##
## Call:
## lm(formula = log(sales) ~ log(price), data = milk)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65425 -0.18405 -0.01262  0.17986  0.65074
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.72060    0.09172   51.47   <2e-16 ***
## log(price)  -1.61858    0.08116  -19.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2687 on 114 degrees of freedom
## Multiple R-squared:  0.7772, Adjusted R-squared:  0.7753
## F-statistic: 397.7 on 1 and 114 DF,  p-value: < 2.2e-16
```

```
#Q = 4.721 - 1.619*P
#P = (4.721 - Q) / 1.619
#Beta = -1.619
ggplot(data = milk) +
  geom_point(mapping = aes(x=log(sales), y=log(price)))  +
  labs(title="Milk Demand",
       y="Log(Price [P])",
       x = "Log(Sales [Q])")
```




```
#Q = e^4.721 * P^-1.619
```

Optimization through Marginal Equations:

total.profit = (P - 1) * [e^4.721 * P^(-1.619)] = 112.28(P-1) / P^1.619 marginal.profit = 181.782/P^2.619 - 69.502/P^1.619 0 = 181.782/P^2.619 - 69.502/P^1.619 optimal.P = 1619 / 619 = $2.62

General Functions

```
2.62-1 #$1.62
```

```
## [1] 1.62
```

```
#log(Q) = 4.721 - 1.619*log(P)
4.721 - 1.619*log(2.62) #3.161621
```

```
## [1] 3.161621
```

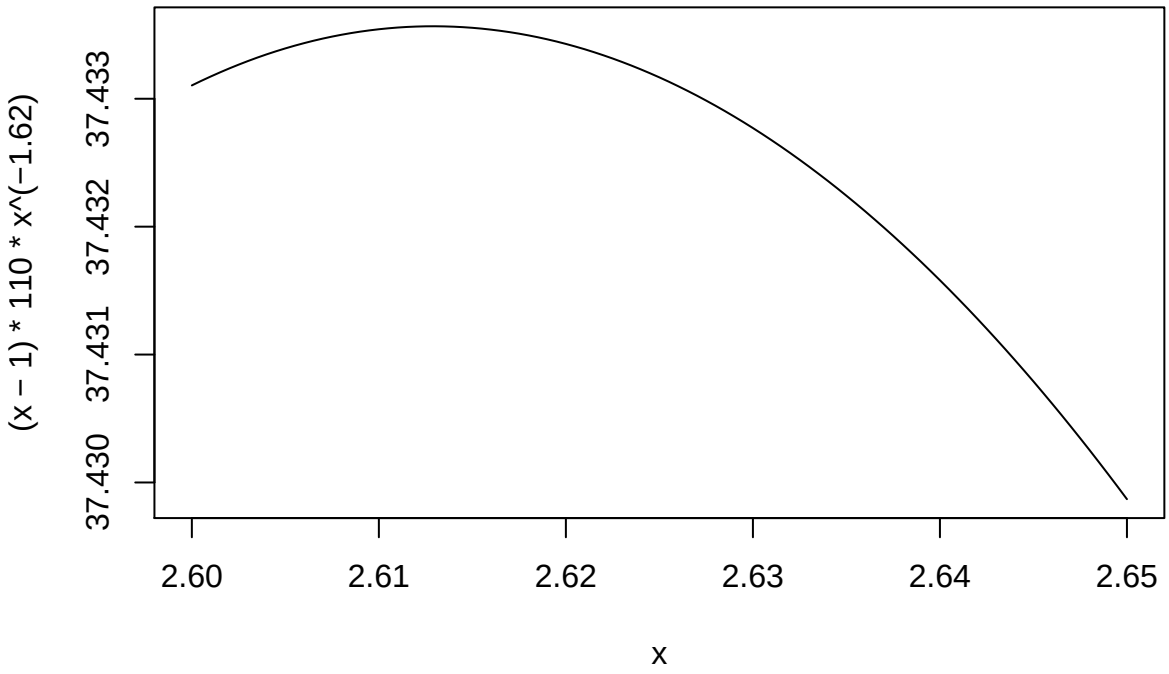
```
#log(Q) = 3.161621
#Q = e^3.161621
exp(3.161621) #23.60883
```

```
## [1] 23.60883
```

```
#average proft
1.62*23.60883 #$38.25
```

```
## [1] 38.2463
```

```
curve((x-1)*110*x^(-1.62), from=2.6, to=2.65)
```



As a profit-maximizing, price-setting merchant, you would charge a certain price for milk based on the maximization of the following total proft formula: total.profit = (P - 1) * [eˆ4.721 * Pˆ(-1.619)]. This equation can be derived from the general total profit equation, which is total revenue (Q * P) minus total cost (Q * c). Quanity can be factored out, leaving total.profit = Q * (P - c). Price can represented as a function of quantity by solving for an ordinary least squares regression of the natural logarithms of Sales (Q) and Price (P). This equation can be plugged in for Q, leaving a total profit equation in terms of only P and c. Taking the derivative of total profit with regards to P yields marginal profit, which, when set equal to 0, will give the optimal price for maximizing profit.

With a per-unit cost (c) of 1 dollar, a profit-maximizing seller would price milk at $2.62. At a this price, the store would sell, on average, around 23.6 cartons of milk at a gross margin of 2.62-1 = 1.62. The resulting average profit per day is 23.6*1.62 = $38.25 from milk sales.

Key Points: 1. Net profit (N) can be calculated as quantity sold (Q) multiplied by the difference between price (P) and unit cost (c). N = Q * (P - c) 2. The equation representing units sold (Q) given a certain price (P) can be found by creating a linear regression between the natural logarithms of the variables. Q = eˆ4.721 * Pˆ-1.619 3. The above equations can be combined by plugging in P as a function of Q: N = (112.280 * Pˆ-1.619) * (P - c) 4. With this equation, N can be maximized by solving for a value of P with a given c. Taking the derivative of the equation in Point 3 with respect to P will result in an equation for marginal net profit, which can then be set equal to 0 to find the profit-maximizing price.