Assignment4.2_BrownLincoln.R

x

2021-07-04

```
# Lincoln Brown
# Assignment 4.2
# DSC520-T301
# Dr. Bushart

# Imports
library(psych)
library(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
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
library(pastecs)
```

```
##
## Attaching package: 'pastecs'

## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
# Load the data
scores <- read.csv("/media/x/disk/School/DSC520/Wk4/scores.csv")
describe(scores)
```

```
##          vars  n   mean    sd median trimmed   mad min max range  skew kurtosis
## Count       1 38  14.47  6.45   10.0   13.44  0.00  10  30    20  1.07    -0.05
## Score       2 38 317.50 47.77  322.5  321.09 40.77 200 395   195 -0.69    -0.10
## Section*    3 38   1.50  0.51    1.5    1.50  0.74   1   2     1  0.00    -2.05
##            se
```

```
## Count    1.05
## Score    7.75
## Section* 0.08
```

```
str(scores)
```

```
## 'data.frame':    38 obs. of  3 variables:
##  $ Count  : int  10 10 20 10 10 10 10 30 10 10 ...
##  $ Score  : int  200 205 235 240 250 265 275 285 295 300 ...
##  $ Section: chr  "Sports" "Sports" "Sports" "Sports" ...
```

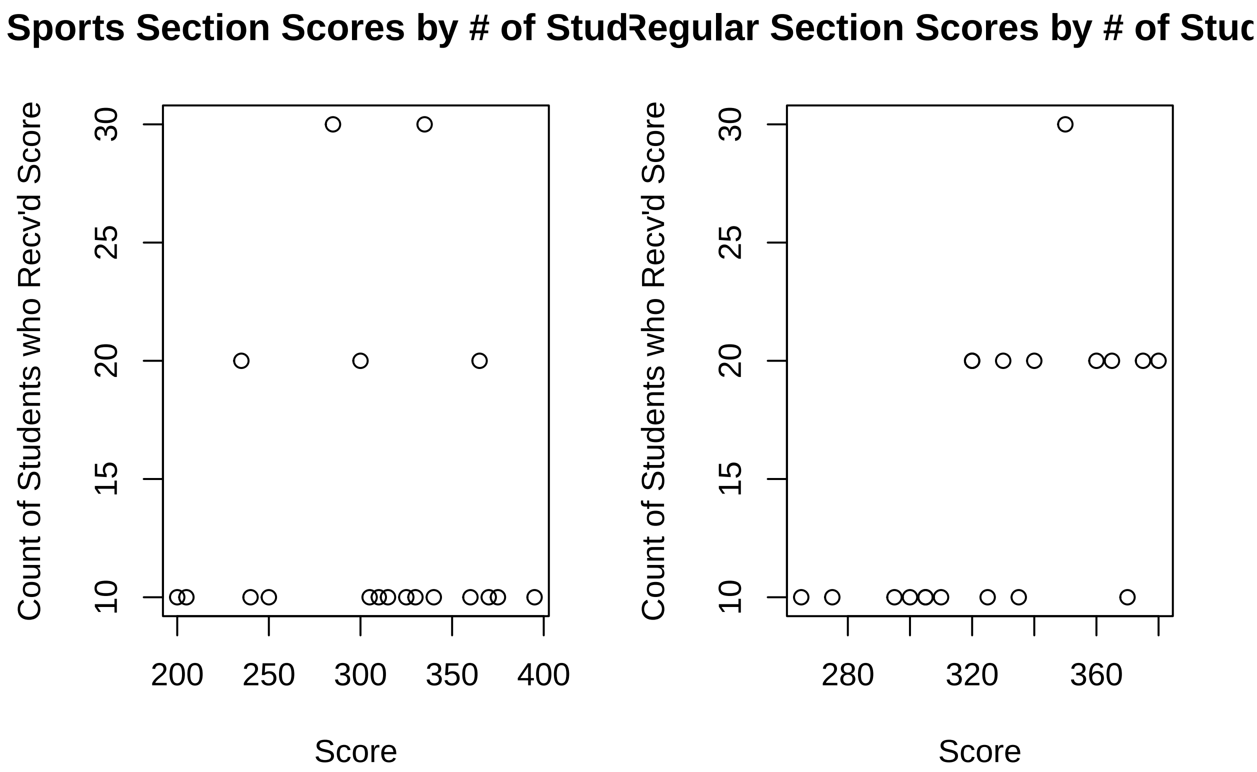
```
colnames(scores)
```

```
## [1] "Count"   "Score"   "Section"
```

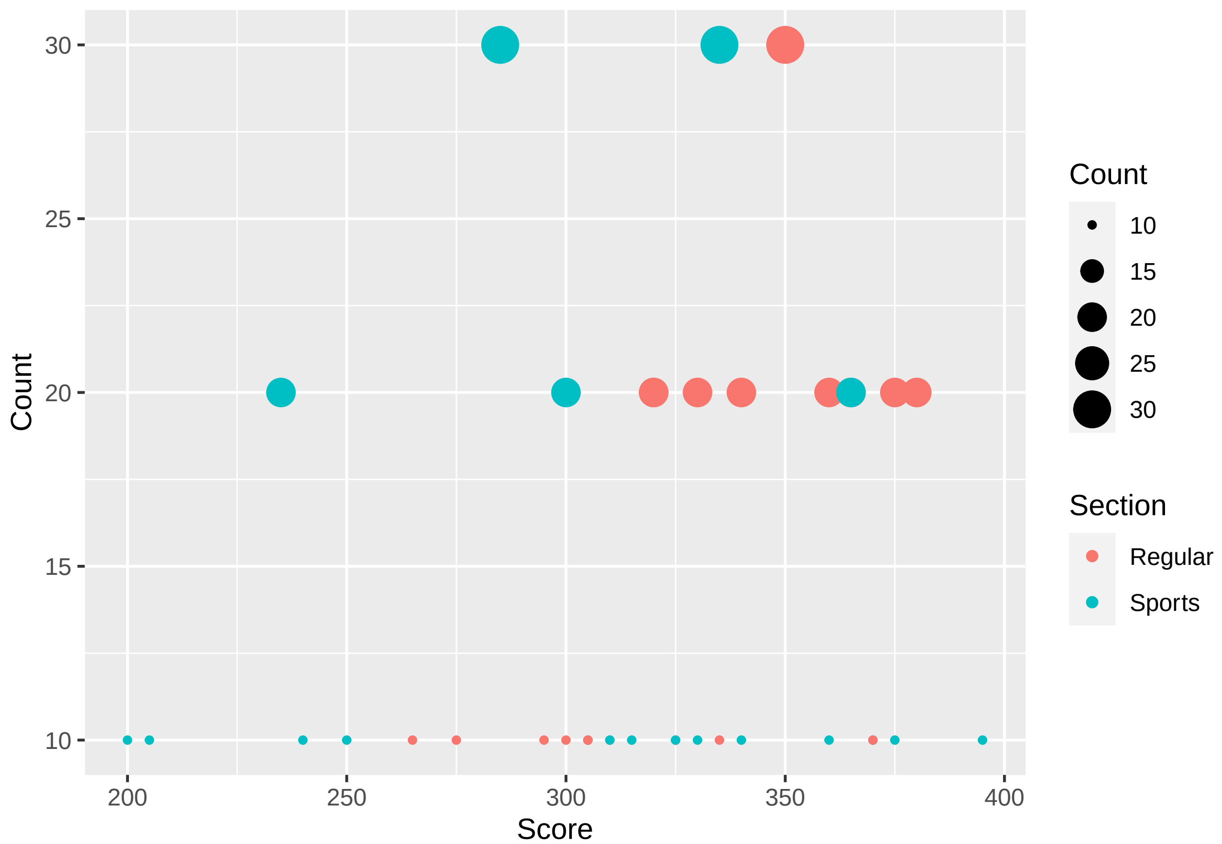
```
scores
```

```
##    Count Score Section
## 1     10   200  Sports
## 2     10   205  Sports
## 3     20   235  Sports
## 4     10   240  Sports
## 5     10   250  Sports
## 6     10   265 Regular
## 7     10   275 Regular
## 8     30   285  Sports
## 9     10   295 Regular
## 10    10   300 Regular
## 11    20   300  Sports
## 12    10   305  Sports
## 13    10   305 Regular
## 14    10   310 Regular
## 15    10   310  Sports
## 16    20   320 Regular
## 17    10   305 Regular
## 18    10   315  Sports
## 19    20   320 Regular
## 20    10   325 Regular
## 21    10   325  Sports
## 22    20   330 Regular
## 23    10   330  Sports
## 24    30   335  Sports
## 25    10   335 Regular
## 26    20   340 Regular
## 27    10   340  Sports
## 28    30   350 Regular
## 29    20   360 Regular
## 30    10   360  Sports
## 31    20   365 Regular
## 32    20   365  Sports
## 33    10   370  Sports
## 34    10   370 Regular
## 35    20   375 Regular
## 36    10   375  Sports
## 37    20   380 Regular
## 38    10   395  Sports
```

```r
# 1. What are the observational units in this study?
# The observational units are the scores and counts of the students
# who received that score in both sections of the class.

# 2. Identify which variables are quantitative/categorical

quantitative <- c("Count", "Score")

categorical <- c("Section")

# 3. Create a variable for each section's subset
names <- unique(scores['Section'])
sports_section <- filter(scores, scores$Section==names[1,1])
reg_section <- filter(scores, scores$Section==names[2,1])


# 4. Plot each Section's scores and the count of students reaching the score.
#Sport section

sport_x <- sports_section$Score
sport_y <- sports_section$Count
reg_x <- reg_section$Score
reg_y <- reg_section$Count
sport_title <- "Sports Section Scores by # of Students"
reg_title <- "Regular Section Scores by # of Students"
y_lab <- "Count of Students who Recv'd Score"
x_lab <- "Score"



par(mfrow=c(1,2))
#Sports Section
plot(sport_x, sport_y, xlab=x_lab, ylab=y_lab, type="p", main=sport_title)

#Regular Section
plot(reg_x, reg_y, xlab=x_lab, ylab=y_lab, type="p", main=reg_title)
```




```
#Both plots using ggplot
ggplot(scores, aes(y=Count, x=Score, color=Section, size=Count)) + geom_point()
```




```
# 4.1 Comparing and contrasting the point distributions
sport_students <- sum(sports_section$Count)
reg_students <- sum(reg_section$Count)
sport_scores <- sum(sports_section$Count * sports_section$Score)
sport_avg <- sport_scores / sport_students
reg_scores <- sum(reg_section$Count * reg_section$Score)
reg_avg <- reg_scores / reg_students

# The distributions are fairly consistent, but the Regular section has
# a higher concentration of students scoring above 300.
# The Regular section also has less variance than the Sports section.
# The mean for the Regular section is also higher (335) than the Sports (307)

# 4.2
# As stated above, the mean for the Regular section was higher than the mean for
# the Sports section. The variance was also less for the Regular section, with a
# larger minimum value than is found in the Sports section. Signifying better
# performance in the Regular section. However, not every student in
# the Regular section scored better than every student in the Sports section.
# The Central Tendency of the Regular Section implies that the regular Section
# is more likely to score higher.

# 4.3
# What could be one additional influencing variable that wasn't mentioned?
# An influencing variable could be the size of the classes in each section,
# I can't imagine that each class had 260 or 290 students, so maybe these classes
```

```r
# had a difference in size.




# 2. Housing Data
library(readxl)
library(plyr)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```r
library(scales)
```

```
##
## Attaching package: 'scales'
## The following objects are masked from 'package:psych':
##
##     alpha, rescale
```

```r
housing <- read_excel("/media/x/disk/School/DSC520/datasets/week-6-housing.xlsx")
housing
```

```
## # A tibble: 12,865 x 24
##    `Sale Date`         `Sale Price` sale_reason sale_instrument sale_warning
##    <dttm>                     <dbl>       <dbl>           <dbl> <chr>
##  1 2006-01-03 00:00:00       698000           1               3 <NA>
##  2 2006-01-03 00:00:00       649990           1               3 <NA>
##  3 2006-01-03 00:00:00       572500           1               3 <NA>
##  4 2006-01-03 00:00:00       420000           1               3 <NA>
##  5 2006-01-03 00:00:00       369900           1               3 15
##  6 2006-01-03 00:00:00       184667           1              15 18 51
##  7 2006-01-04 00:00:00      1050000           1               3 <NA>
##  8 2006-01-04 00:00:00       875000           1               3 <NA>
##  9 2006-01-04 00:00:00       660000           1               3 <NA>
## 10 2006-01-04 00:00:00       650000           1               3 <NA>
## # ... with 12,855 more rows, and 19 more variables: sitetype <chr>,
## #   addr_full <chr>, zip5 <dbl>, ctyname <chr>, postalctyn <chr>, lon <dbl>,
## #   lat <dbl>, building_grade <dbl>, square_feet_total_living <dbl>,
## #   bedrooms <dbl>, bath_full_count <dbl>, bath_half_count <dbl>,
## #   bath_3qtr_count <dbl>, year_built <dbl>, year_renovated <dbl>,
## #   current_zoning <chr>, sq_ft_lot <dbl>, prop_type <chr>, present_use <dbl>
```

```r
dim(housing)
```

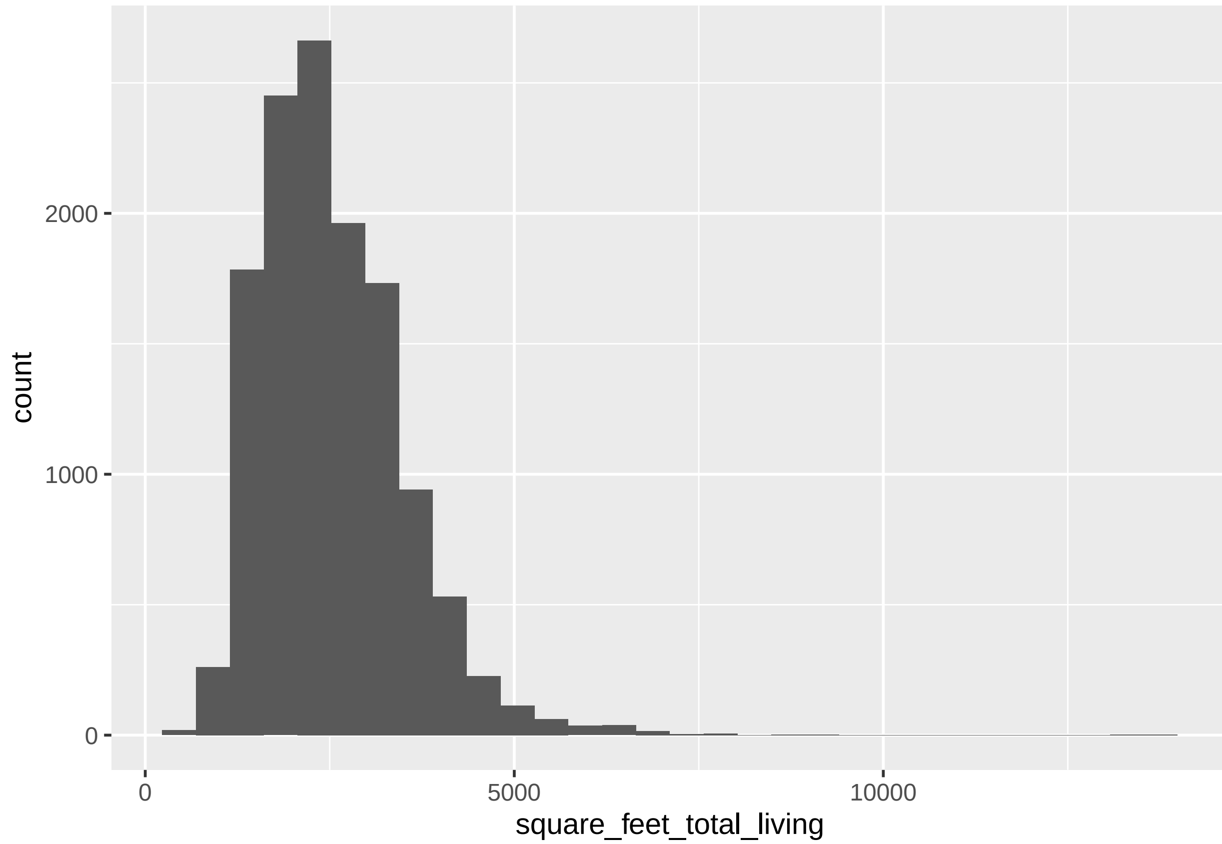
```
## [1] 12865    24
```

```r
# 2.1 Use the apply function on a variable in your dataset.
#Two ways of finding mean:
#sum_sp <- apply(housing['Sale Price'], 2, sum)
#mean_sp <- sum_sp / nrow(housing)
mean_sp <- apply(housing['Sale Price'], 2, mean)


# 2.2 Use the aggregate function on a variable in your dataset.
cities <- unique(housing$ctyname)
cities
```

```
## [1] "REDMOND"   NA          "SAMMAMISH"
```

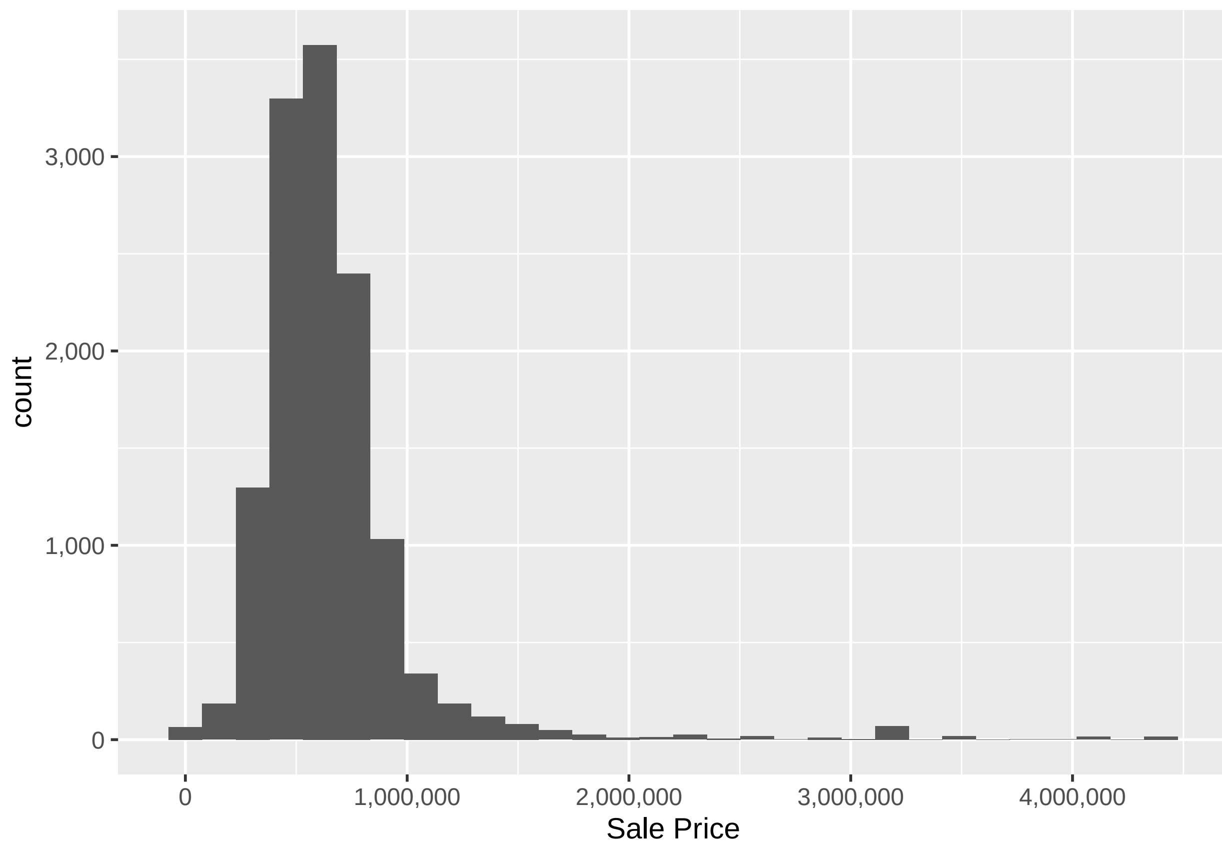
```r
zips <- unique(housing$zip5)



# Calculate the mean of each zip
avg_zip <- aggregate(`Sale Price` ~ zip5, housing, mean)
#redmond <- filter(housing, housing$ctyname=='REDMOND')
#sammamish <- filter(housing, housing$ctyname=='SAMMAMISH')


# 2.3 Use the plyr function on a variable in your dataset - more specifically, I
# want to see you split some data, perform a modification to the data,
# and then bring it back together.
housing_refined <- housing[c(2,6,8,9,14,15,16,17,18,19)]

est_dp <- function(x)
{
  c(dwn_pmt=with(x, x[2] * .25))
}

housing_size <- ddply(housing, .(square_feet_total_living), transform,
                    house.size = cut(square_feet_total_living, breaks = c(-Inf, 1000, 2000, 3000, Inf),
                        labels = c("Tiny", "Small", "Medium", "L//////////////////arge"))
                    )

# 2.4 Check distributions of the data
gsqft <- ggplot(housing, aes(square_feet_total_living))

gsqft + geom_histogram()
```

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




```r
gprice <- ggplot(housing, aes(`Sale Price`)) +
  scale_x_continuous(labels = comma) + scale_y_continuous(labels = comma)

gprice +geom_histogram()
```

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



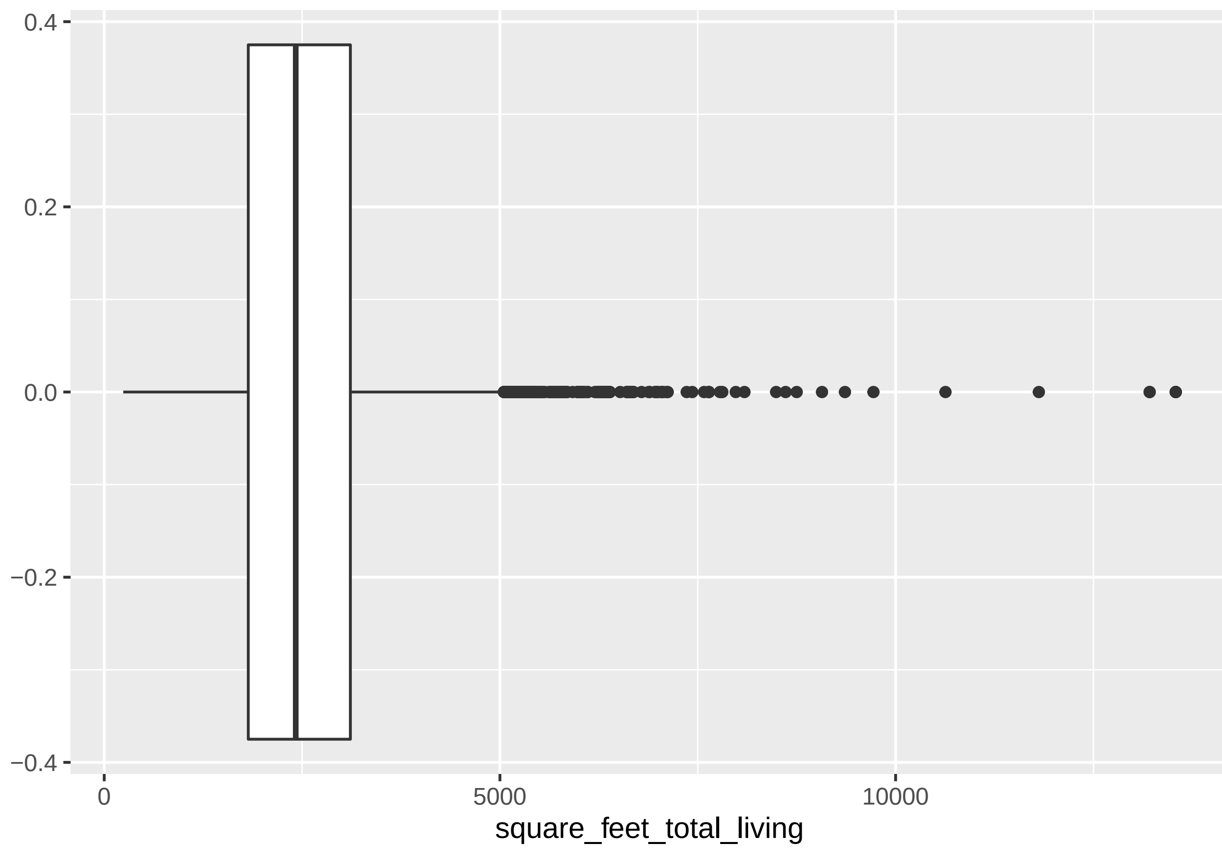

```
describe(housing$square_feet_total_living)
```

```
##    vars     n    mean     sd median trimmed    mad min   max range skew
## X1    1 12865 2539.51 989.82   2420 2453.44 948.86 240 13540 13300 1.61
##    kurtosis   se
## X1     8.59 8.73
```

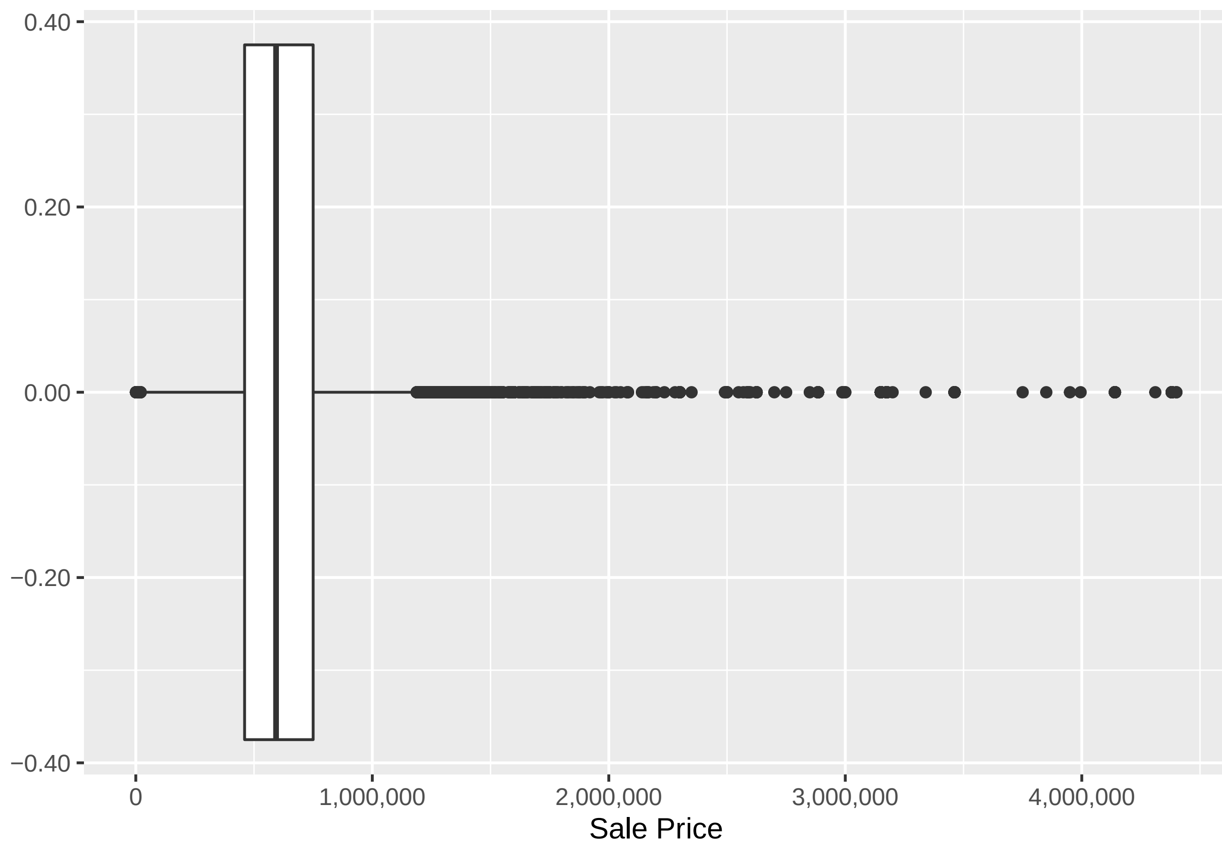
```
describe(housing$`Sale Price`)
```

```
##    vars     n     mean       sd median  trimmed      mad min     max   range
## X1    1 12865 660737.8 404381.1 593000 605920.4 212011.8 698 4400000 4399302
##    skew kurtosis      se
## X1 4.49    29.22 3565.22
```

```
# 2.5 Identify if there are any outliers
#For both Sqft total living space and Sale Price there are outliers in the data.
gsqft + geom_boxplot()
```




```
gprice + geom_boxplot()
```




```r
# 2.6 Create at least 2 new variables.
#Refine the data frame to include Sale Price, Est Down Payment, and Square Ft
housing1 <- housing

housing1$dwn_pmt <- est_dp(housing1)

(refine_housing <- housing1[c(2, 14, 25)])
```

```
## # A tibble: 12,865 x 3
##    `Sale Price` square_feet_total_living dwn_pmt
##           <dbl>                    <dbl> <named list>
##  1       698000                     2810 <dbl [12,865]>
##  2       649990                     2880 <dbl [12,865]>
##  3       572500                     2770 <dbl [12,865]>
##  4       420000                     1620 <dbl [12,865]>
##  5       369900                     1440 <dbl [12,865]>
##  6       184667                     4160 <dbl [12,865]>
##  7      1050000                     3960 <dbl [12,865]>
##  8       875000                     3720 <dbl [12,865]>
##  9       660000                     4160 <dbl [12,865]>
## 10       650000                     2760 <dbl [12,865]>
## # ... with 12,855 more rows
```

```r
biggest_houses <- refine_housing[order(refine_housing$square_feet_total_living, decreasing=TRUE),]
head(biggest_houses)
```

```
## # A tibble: 6 x 3
```

```
##   `Sale Price` square_feet_total_living dwn_pmt
##          <dbl>                    <dbl> <named list>
## 1      2300000                    13540 <dbl [12,865]>
## 2      1300000                    13540 <dbl [12,865]>
## 3      2280000                    13540 <dbl [12,865]>
## 4      3000000                    13210 <dbl [12,865]>
## 5      2491149                    13210 <dbl [12,865]>
## 6      3995000                    11810 <dbl [12,865]>
```

```
most_expensive <- refine_housing[order(refine_housing$`Sale Price`, decreasing=TRUE),]
head(most_expensive)
```

```
## # A tibble: 6 x 3
##   `Sale Price` square_feet_total_living dwn_pmt
##          <dbl>                    <dbl> <named list>
## 1      4400000                     5790 <dbl [12,865]>
## 2      4400000                     2410 <dbl [12,865]>
## 3      4380542                     3290 <dbl [12,865]>
## 4      4380542                     2450 <dbl [12,865]>
## 5      4380542                     2750 <dbl [12,865]>
## 6      4380542                     3010 <dbl [12,865]>
```

```
max_sqft <- max(housing$square_feet_total_living)
min_sqft <- min(housing_refined$square_feet_total_living)
```