4.2.2 Exercise

Felipe Rodriguez

2022-01-07

Import Libraries

```
setwd('/Users/feliperodriguez/OneDrive - Bellevue University/Github/dsc520/')
library(ggplot2)
library(readxl)
library(plyr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## v purrr   1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
```

```
library(dplyr)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

Read in excel doc

```
housing <- read_excel('/Users/feliperodriguez/OneDrive - Bellevue University/Github/dsc520/data/week-7-h
```

Use the apply function with a variable in your dataset

```
# Total mean for all houses
sales_price_mean <- apply(housing[c('Sale Price')], 2, mean)
sales_price_mean
```

```
## Sale Price
##   660737.7
```

Use the aggragate function on a variable in your dataset

```
# Calculates Sale Price mean by Zipcode
aggregate(housing$`Sale Price` ~ zip5, housing, mean)
```

```
##    zip5 housing$`Sale Price`
## 1 98052             649375.4
## 2 98053             672623.7
## 3 98059             645000.0
## 4 98074             951543.8
```

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

```
# Produces max salesprice for each zip
max_price_zip <- ddply(housing, ~ zip5, summarize, max_price = max(`Sale Price`))
max_price_zip
```

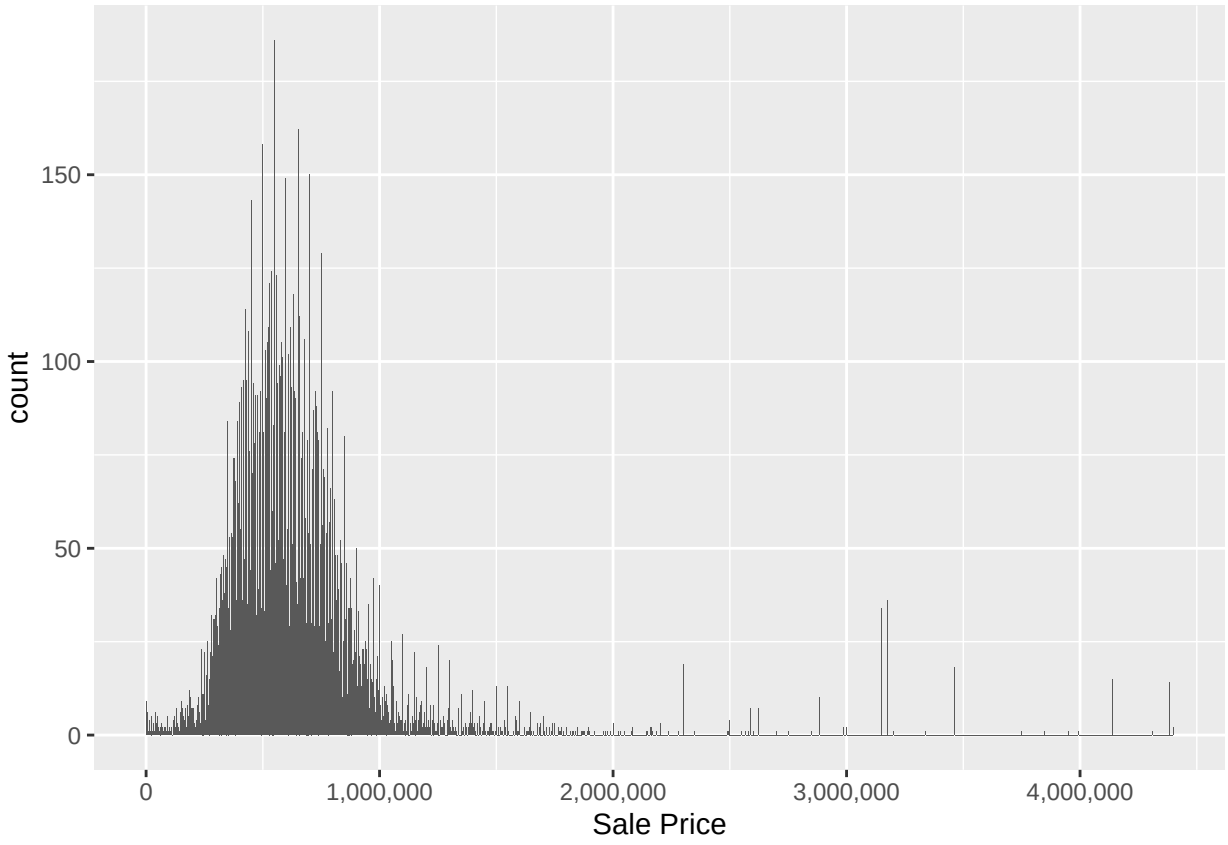
```
##    zip5 max_price
## 1 98052   4400000
## 2 98053   3850000
## 3 98059    645000
## 4 98074   2160200
```

```
# Produces min sale price for each zip
min_price_zip <- ddply(housing, ~ zip5, summarize, min_price = min(`Sale Price`))
min_price_zip
```

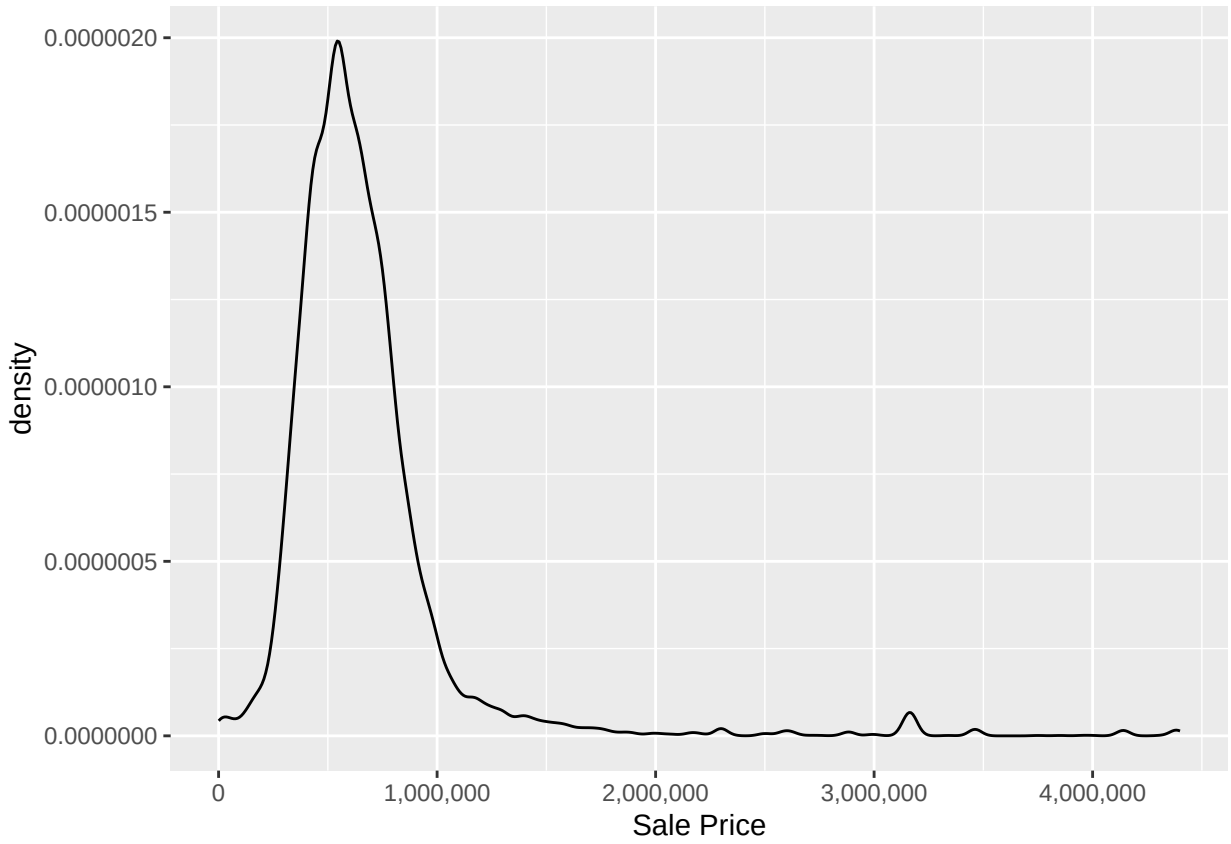
```
##    zip5 min_price
## 1 98052      2031
## 2 98053       698
## 3 98059    645000
## 4 98074    434000
```

Check the distributions of the data and Indentify any outliers

```
# Produces a histogram for Sale Price
housing_hist <- ggplot(housing, aes(`Sale Price`)) + geom_histogram(bins=1250)
housing_hist + scale_x_continuous(labels = comma)
```




```
# Produces density ploy for Sale Price
housing_density <- ggplot(housing, aes(`Sale Price`)) + geom_density()
housing_density + scale_x_continuous(labels = comma) + scale_y_continuous(labels=comma)
```



The distribution of Sale Price of the data gives normal tendencies until Sale Price reaches 1.2 million. The majority of homes fall in between 0 and 1.2 million. After that value, the count of homes in that higher price ranges begin to decrease. The outliers that can be seen are above the value of 2 million and represent a small section of the total data.

Create at least 2 new variables

```
# Creates new variable 'Price Per SQFT' and produces a new merged data set
housing_new <- mutate(housing, price_per_sqft = (`Sale Price`/sq_ft_lot))
# Creates new variable 'SQFT Total Nonliving' and merges it new data set
housing_new <- mutate(housing_new, sq_ft_total_nonliving = (sq_ft_lot-square_feet_total_living))
# Display new variables
head(select(housing_new, price_per_sqft, sq_ft_total_nonliving))
```

```
## # A tibble: 6 x 2
##   price_per_sqft sq_ft_total_nonliving
##            <dbl>                 <dbl>
## 1          105.                   3825
## 2          117.                   2690
## 3           67.8                  5674
## 4           43.8                  7980
## 5           49.1                  6086
## 6           25.4                  3120
```