Exercise 12-1

Gao

2023-10-06

First, we must install our libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## 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
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --
## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tune         1.1.2
## v infer        1.0.5     v workflows    1.1.3
## v modeldata    1.2.0     v workflowsets 1.0.1
## v parsnip      1.1.1     v yardstick    1.2.0
## v recipes      1.0.8
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

Set working directory

```
getwd()
```

```
## [1] "/Users/andrewgao/Documents/GitHub/Advanced-Data-Science/Gao/Unit 3"
```

Retrieve the .csv file

```
housing <- read_csv("melbourne_housing.csv")
```

```
## Warning: One or more parsing issues, call ‘problems()‘ on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 34857 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (8): Suburb, Address, Type, Method, SellerG, Date, CouncilArea, Regionname
## dbl (13): Rooms, Price, Distance, Postcode, Bedroom2, Bathroom, Car, Landsiz...
##
## 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.
```

```
names(housing) <- names(housing) %>% str_to_title()
```

Examine for missing values

```
apply(X = housing, MARGIN = 2, FUN = function(col)
  sum(is.na(col)))
```

```
##        Suburb       Address         Rooms          Type         Price
##             0             0             0             0          7610
##        Method       Sellerg          Date      Distance      Postcode
##             0             0             0             1             1
##      Bedroom2      Bathroom           Car      Landsize  Buildingarea
##          8217          8226          8728         11810         21115
##     Yearbuilt   Councilarea     Lattitude    Longtitude    Regionname
##         19306             0          7976          7976             0
## Propertycount
##             3
```

Drop all rows with missing values

```
housing <- na.omit(housing)
```

Return correlation

```
cor(x = housing$Price, y = housing$Buildingarea)
```

```
## [1] 0.5072844
```

```
cor(x = housing$Price, y = housing$Rooms)
```

```
## [1] 0.4750737
```

```
cor(x = housing$Price, y = housing$Car)
```

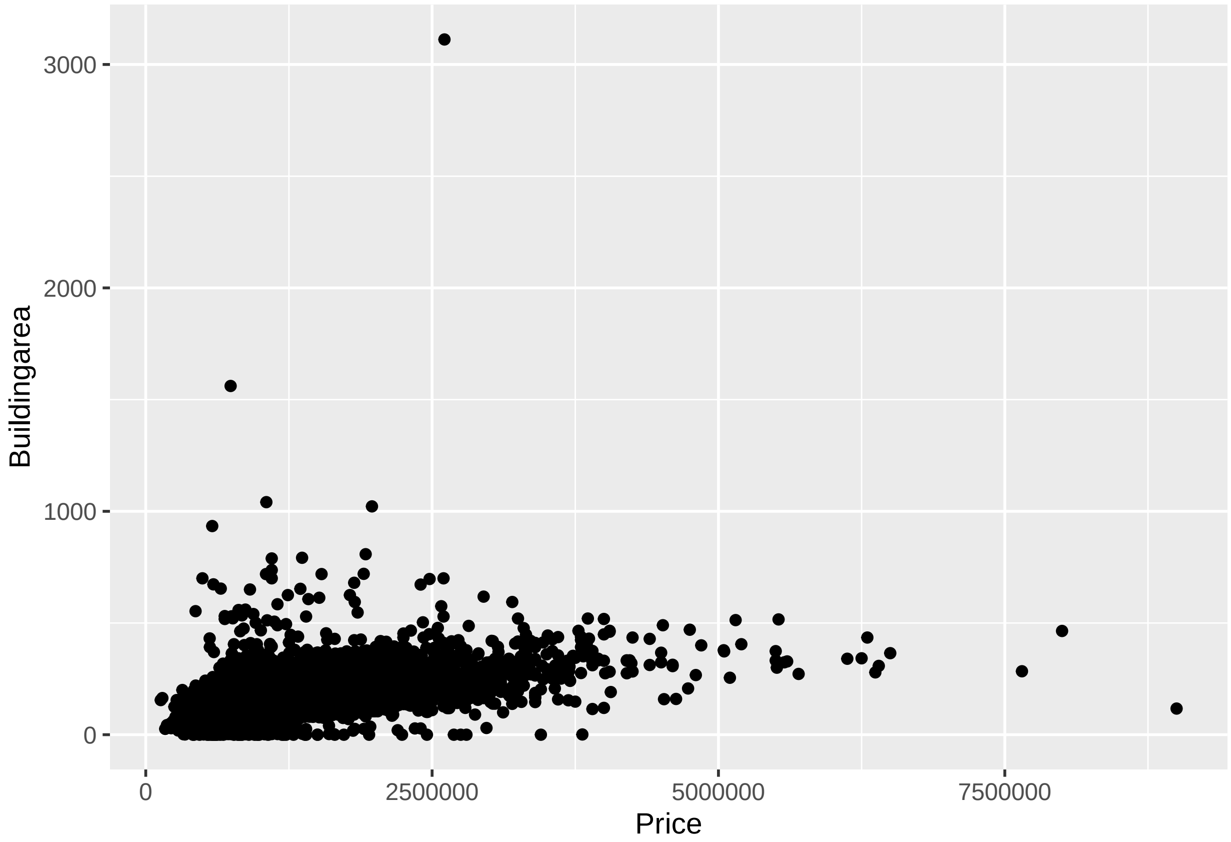
```
## [1] 0.2094641
```

```
cor(x = housing$Price, y = housing$Landsize)
```

```
## [1] 0.0583748
```

Display a scatter pot that shows the relationshiop between the Price and BuildingArea variables

```
ggplot(housing, aes(x=Price, y=Buildingarea)) + geom_point()
```



Split the data

```
housing_split <- initial_split(housing, prop = 0.75)
train <- training(housing_split)
test <- testing(housing_split)
```

Create functions to delete outliers

```
get_upper_fence <- function(x) {
  quantile(x, 0.75) + (1.5 * IQR(x))
}
```

```r
get_lower_fence <- function(x) {
  quantile(x, 0.25) - (1.5 * IQR(x))
}
```

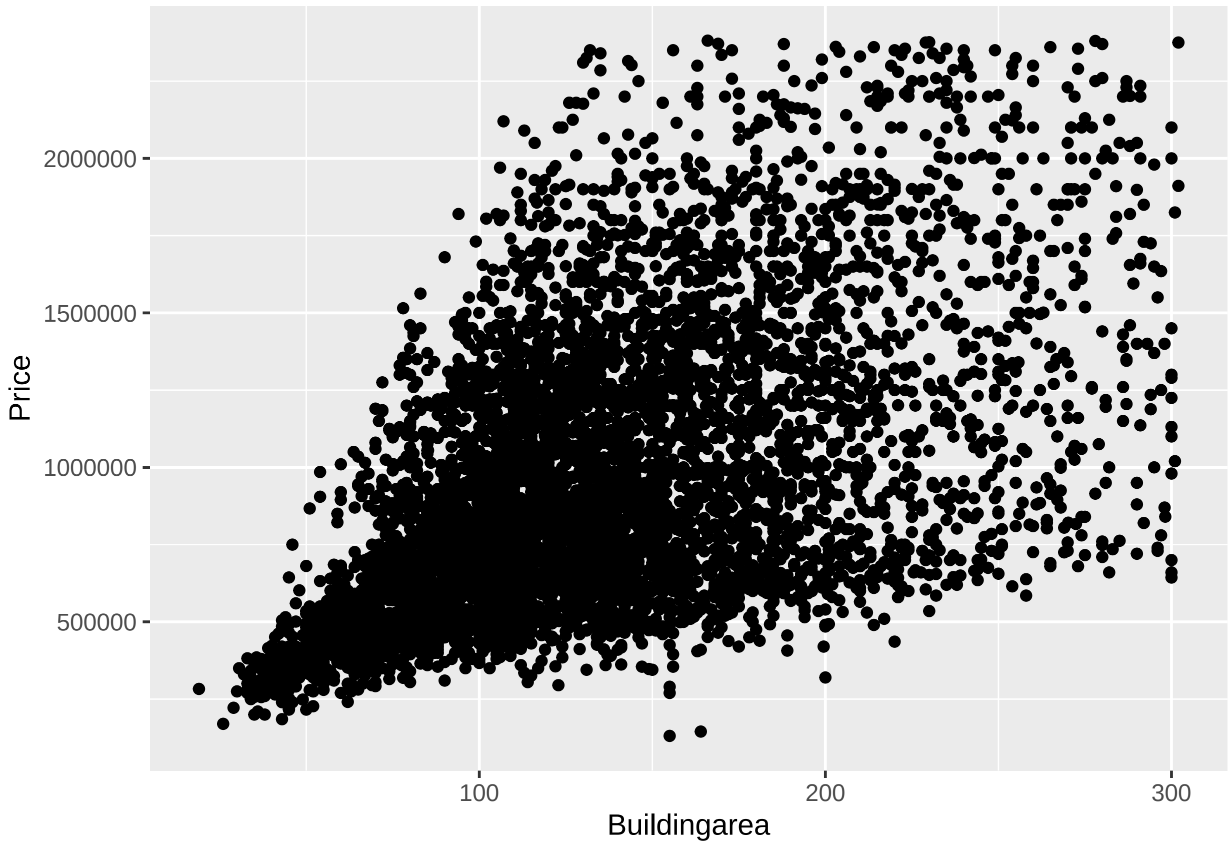
Use the fences to delete outliers

```r
train <- train %>%
  filter_at(vars(Price, Buildingarea),
            all_vars(. > get_lower_fence(.) &
                       . < get_upper_fence(.)))
test <- test %>%
  filter_at(vars(Price, Buildingarea),
            all_vars(. > get_lower_fence(.) &
                       . < get_upper_fence(.)))
```

Delete any really weird numbers

```r
train <- train %>% filter(Buildingarea > Price * (1/20000) & Price < Buildingarea * 100000)
test <- test %>% filter(Buildingarea > Price * (1/20000) & Price < Buildingarea * 100000)

ggplot(train, aes(y=Price, x=Buildingarea)) + geom_point()
```



Create a linear regression model using the training set

```
linear_reg()
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

```
model <- fit(object = linear_reg(), formula = Price ~ Buildingarea,
             data = train)
predict(model, new_data = test)
```

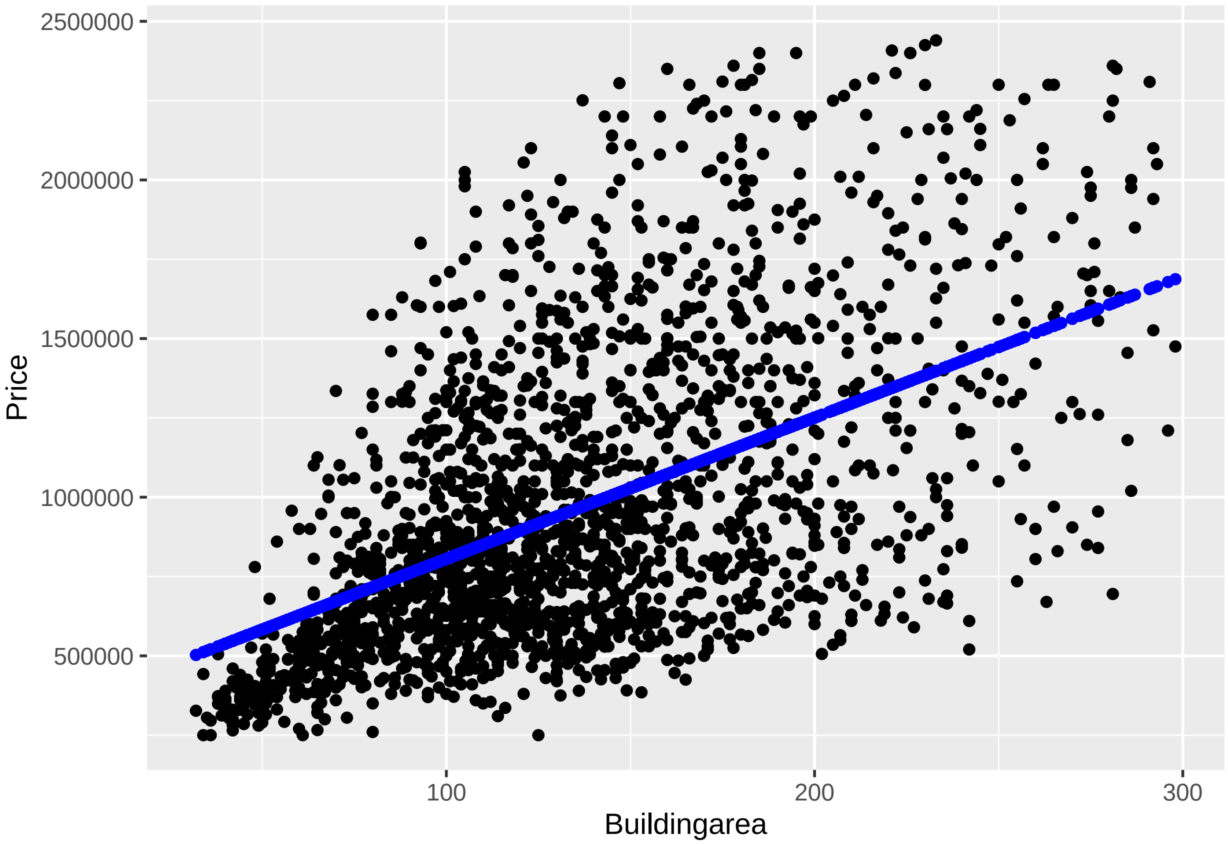
```
## # A tibble: 2,026 x 1
##       .pred
##       <dbl>
##  1  778863.
##  2  792221.
##  3  627474.
##  4 1135074.
##  5  765506.
##  6  720979.
##  7  876821.
##  8  743242.
##  9  952516.
## 10 1010400.
## # i 2,016 more rows
```

Use the testing set

```
model_results <- test %>%
  mutate(predict(model, new_data = test))
```

Plot the plot

```
ggplot(data = model_results) +
  geom_point(aes(x = Buildingarea, y = Price)) +
  geom_point(aes(x = Buildingarea, y = .pred), color = "blue")
```


2500000
2000000
1500000
1000000
500000
Price
100
200
300
Buildingarea