SML 201 Project 3 Predicting House Sold Prices for Seattle and Greater Area

Stuart Duffield and James Armstrong

2019-01-12

Project 3 is due by 11:59pm on Tuesday January 15. Please submit both a .Rmd and a .pdf file on Blackboard by the deadline **and** drop off a hard copy of the pdf file at 26 Prospect Avenue by **11am** of the **next day** of the due date. To look for the drop-off cabinet, after you enter the building turn to the left to enter the lounge area and the file cabinet will then be on your right; you can just drop your report into the open slot of the cabinet labeled "SML 201 Homework"; note that the building might be locked after 6pm and on weekends. You are also welcome to drop off the pdf copy in advance of the deadline.

Late **projects** will be penalized at intervals rounded up to multiples of 24 hours. For example, if you are 3 hours late, 10% off or if you are 30 hours late, 20% off.

Make sure that you have all your digital signatures along with the honor pledge in each of these documents (there should be two signatures if you work in groups).

This project can be completed in groups of up to 2 students. It is okay to work by yourself, if this is preferable. **You may not work with a given student on more than one project.** In other words, if you work with Student_A on Project 1, then you cannot work with Student_A on any other projects. You must work with a new group mate for every project.

When working in a group it is your responsibility to make sure that you are satisfied with all parts of the report and the submission is on time (e.g., we will not entertain arguments that deficiencies are the responsibility of the other group member). We expect that the work on any given problem set or project contains approximately equal contributions from both members of the group; we expect that you each work independently first and then compare your answers with each other once you all finish or you work together. Failing to make contributions and then putting your name on a project will be considered a violation of the honor code. Also, please do not divide work between group mates. For all parts of this problem set, you **MUST** use R commands to print the output as part of your R Markdown file. You are not permitted to find the answer in the R console and then copy and paste the answer into this document.

In general you are not allowed to get help on projects from other people except from your group partner. There is an exception for this project: you are allowed to get help from the instructors if you need help to understand the definitions of the variables of the dataset or the procedure of the experiment.

You are always welcome to ask the instructors clarification questions. *Please treat projects as take-home exams.* Do not make suggestions on how to solve or the possible approach to a question to other students–please keep this in mind when asking questions on Piazza.

For all parts of this problem set, you **MUST** use R commands to print the output as part of your R Markdown file. You are not permitted to find the answer in the R console and then copy paste the answer into this document.

If you are completing this project in a group, please have only **one** person in your group turn in the .Rmd and .pdf files; the other person in your group should turn in the list of the names of the people in your group in the *Text Submission* field on the submission page. This means that **everyone should make a submission**–either a file-upload or a text submission–regardless of whether you are working in a group or not.

Please type your name(s) after "Digitally signed:" below the honor pledge to serve as digital signature(s). Put the pledge and your signature(s) at the beginning of each document that you turn in.

I pledge my honor that I have not violated the honor code when completing this assignment.

Digitally signed: Stuart Duffield and James Armstrong

In order to receive full credits, please

- avoid hard-coding and do not round intermediate calculations;
- round all final numerical answers to four digits after the decimal.
- have sensible titles and axis labels for all your graphs;
- adjust values for all the relevant graphical parameters so that your plots are informative.
- annotate your code and have answers in the write up section, not in the code chunks;
- have all answers written in complete sentences.
- drop off a **color** copy of the pdf by 11am the day after the due date.

Objective of this project

(Hypothetical) Congrats! You have been hired as an intern at Redfin (Redfin.com) in Seattle. As for your first project your manager would like you to build a new model for Redfin to predict the sold prices for houses in the King county area of the Washington state.

To see the prices predicted by Redfin's current model, you can see the number shown near the top of the web page of a listing; e.g., for this house (https://www.redfin.com/WA/Seattle/132-NE-95th-St-98115/unit-B108/home/2316), the Redfin estimate is $401,144.)

Background info and the dataset

We will use a subset of the dataset `kc_house_data.csv` to build the model. The dataset contains sold prices for houses in King County (in Washington state), including Seattle, for transactions made between May 2014 and May 2015. We will use only a subset of the variables in `kc_house_data.csv` because we only want to include variables that have clear definitions and seem relevant for house prices. A description of the original dataset `kc_house_data.csv` and the complete list of the variable definitions can be found here (https://www.kaggle.com/harlfoxem/housesalesprediction/data).

We will use the dataset `subset_kc_house_data.csv` and the definitions for the variables in the dataset are listed below (see the website address provided above for the complete list of the variables).

- **date** The date the house was sold
- **price** Sold price of the house
- **bedrooms** Number of bedrooms in the house
- **bathrooms** Number of bathrooms in the house
- **sqft_living** Square footage of the house

- **sqft_lot** Square footage of the lot
- **floors** Total number of floors (levels) in house
- **waterfront** Does the house have a view to a waterfront (0-No; 1-Yes)
- **condition** How good the overall condition is
- **grade** Overall grade given to the housing unit, based on King County grading system (see table for `BLDGGRADE` on https://www5.kingcounty.gov/sdc/FGDCDocs/RESBLDG_EXTR_faq.htm)
- **sqft_above** Square footage of house apart from the basement
- **sqft_basement** Square footage of the basement
- **yr_built** Year when house was built
- **yr_renovated** Year when house was renovated
- **zipcode** Zip code of the house address

Question 1 (1 pt)

Read the dataset `subset_kc_house_data.csv` into `R` and name it `house`. `house` should have 21613 rows and 15 columns.

```
# load in dataset as `house`
house <- read.csv("C:/Users/Stuart/Desktop/Sophomore Year/SML 201/Project 3/subset_kc_house_data.csv")

# Check dimensions
dim(house)
[1] 21613    15
```

`house` has 21613 rows and 15 columns.

Exploring the relationship between `price` and other variables.

Question 2 (20 pts)

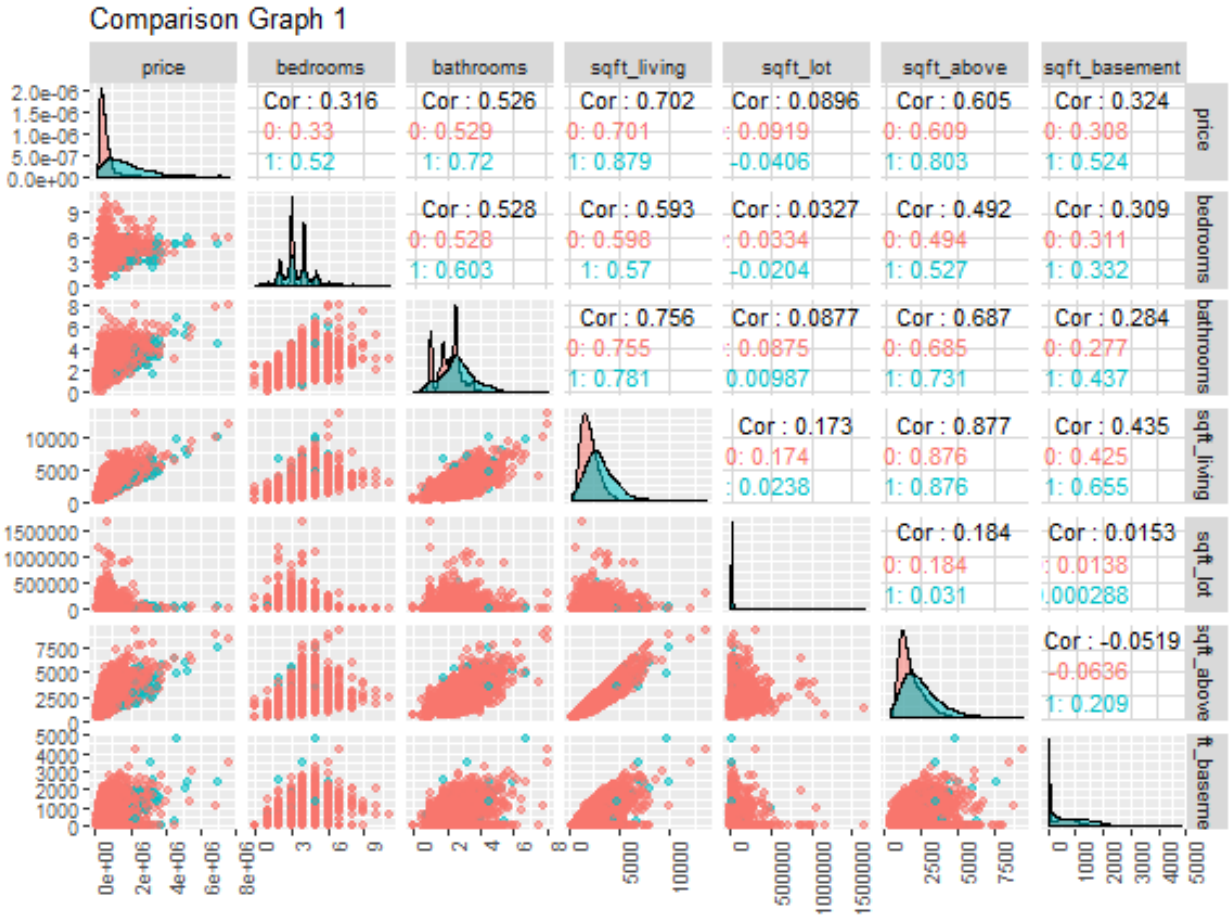
Part a (9 pts)

Use `ggpairs()` in the `GGally` package and make matrices of scatterplots to investigate the pairwise relationships between the variables;

Group `sqft_living`, `sqft_lot`, `sqft_above` and `sqft_basement` in the same plot to see if any of these variables are correlated. In general, it is good to have x-variables that are highly correlated (i.e., with correlation close to -1 or 1) among themselves in your model? Explain why yes or why no.

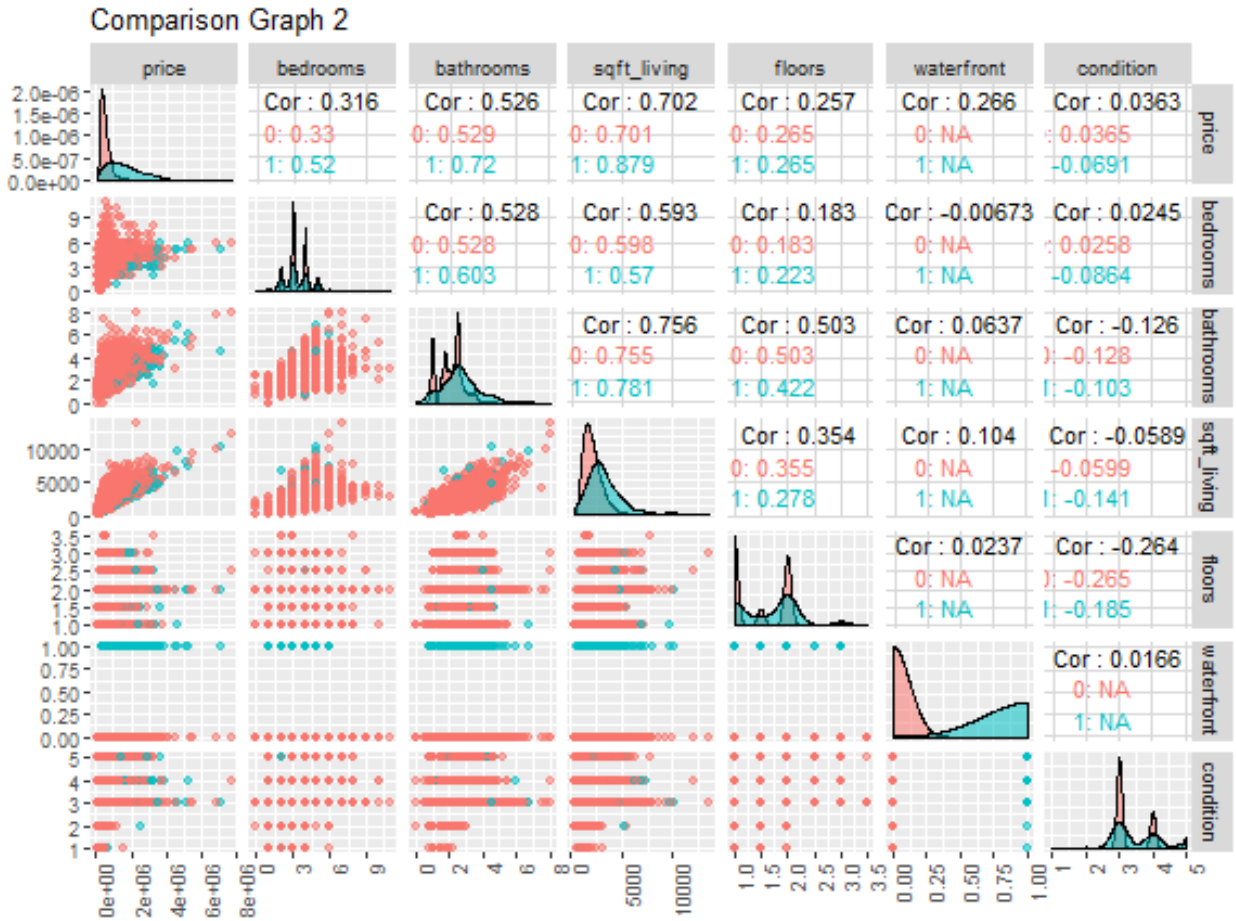
```
# Check sqare footage of living area of 30+ bedroom house
house[house$bedrooms > 30, ]$sqft_living
[1] 1620
```

```
# Remove erroneous data
house = house[-which(house$bedrooms == 0 & house$bathrooms == 0 |
    house$bedrooms > 30), ]

# Make Plots
library(ggplot2)
library(GGally)

ggpairs(data = house[, c("price", "bedrooms", "bathrooms", "sqft_living",
    "sqft_lot", "sqft_above", "sqft_basement")], title = "Comparison Graph 1",
    aes(colour = as.factor(house$waterfront), alpha = 0.4)) + theme(axis.text.x = element_text(angle = 9
    hjust = 1))
```

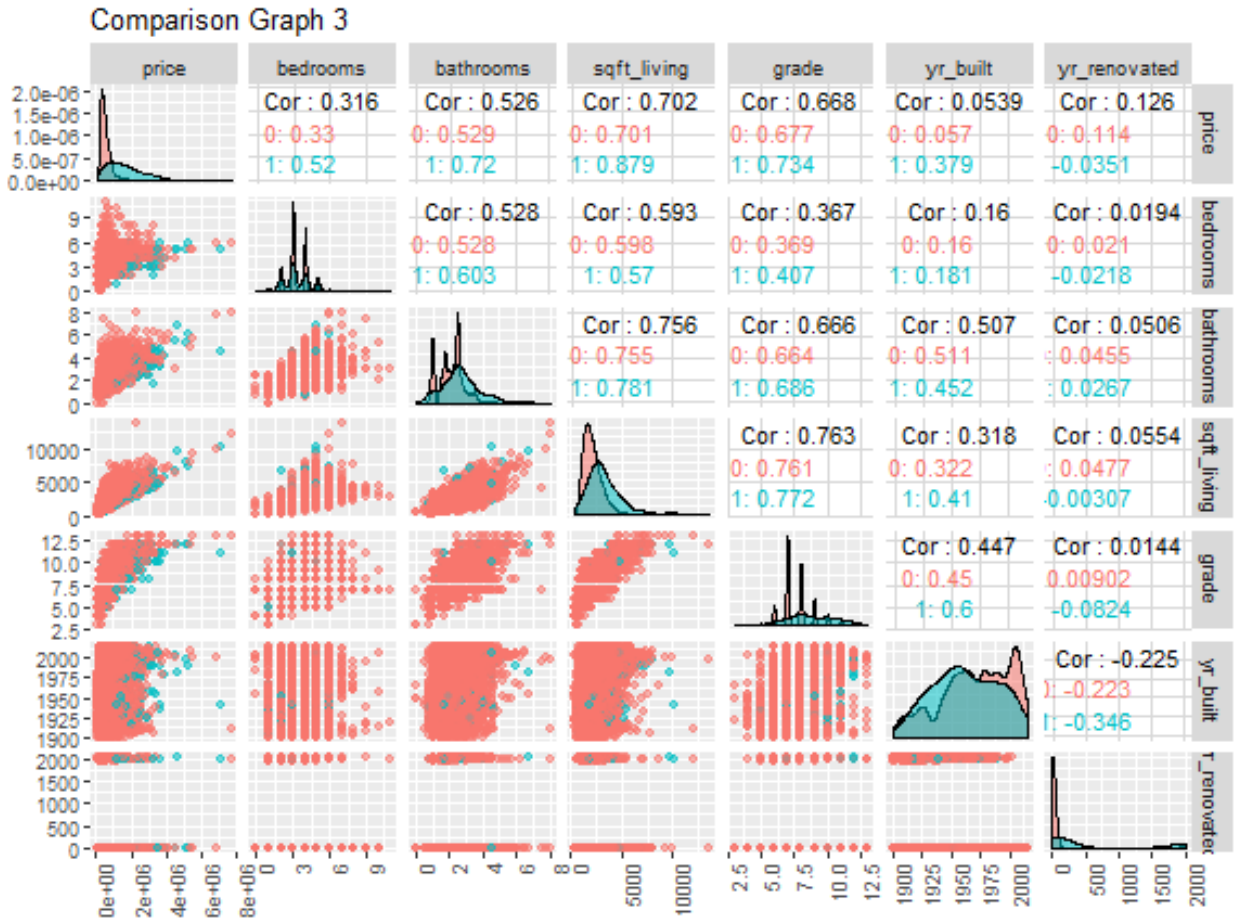



```
ggpairs(data = house[, c("price", "bedrooms", "bathrooms", "sqft_living",
    "floors", "waterfront", "condition")], title = "Comparison Graph 2",
    aes(colour = as.factor(house$waterfront), alpha = 0.4)) + theme(axis.text.x = element_text(angle = 9
    hjust = 1))
Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero
```

```
Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero
```

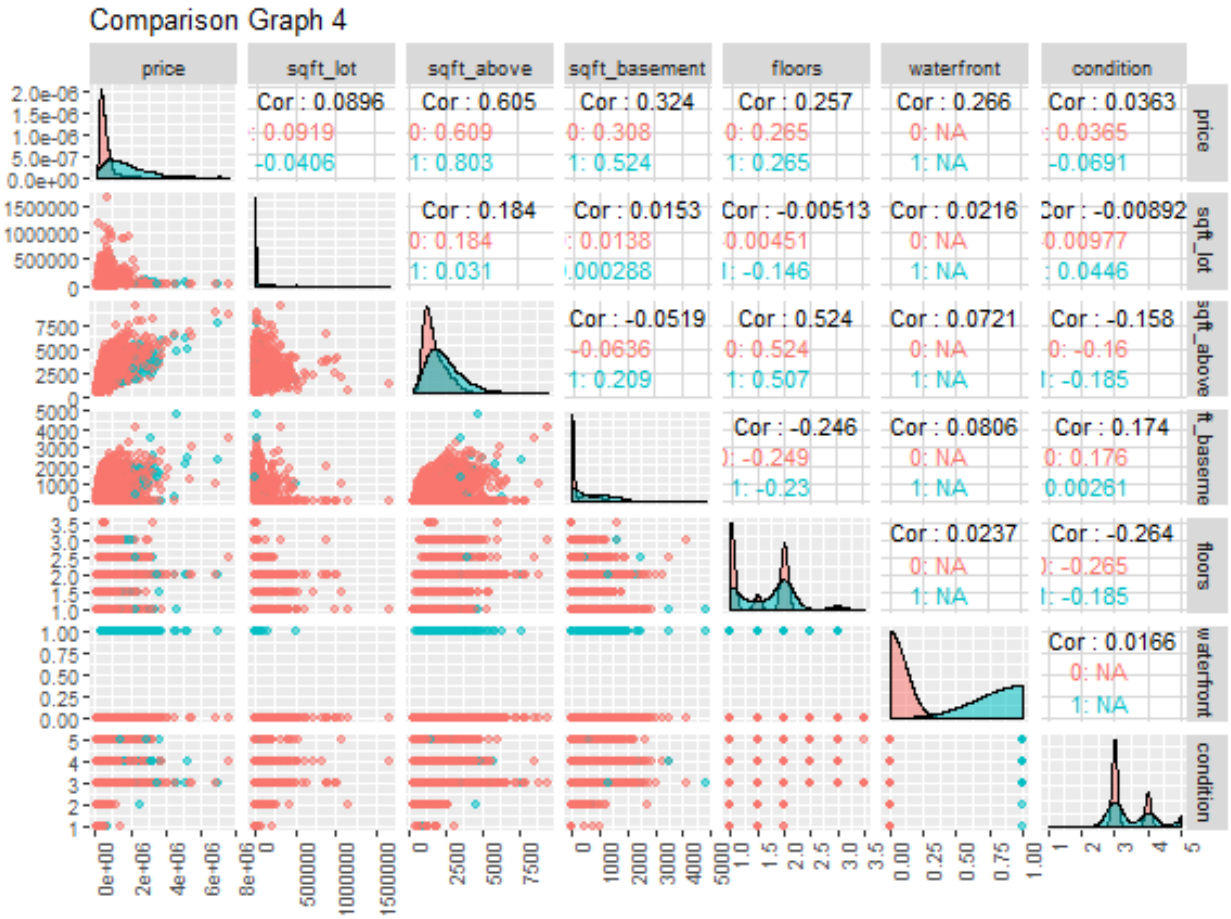



```
ggpairs(data = house[, c("price", "bedrooms", "bathrooms", "sqft_living",
    "grade", "yr_built", "yr_renovated")], title = "Comparison Graph 3",
    aes(colour = as.factor(house$waterfront), alpha = 0.4)) + theme(axis.text.x = element_text(angle = 9
    hjust = 1))
```

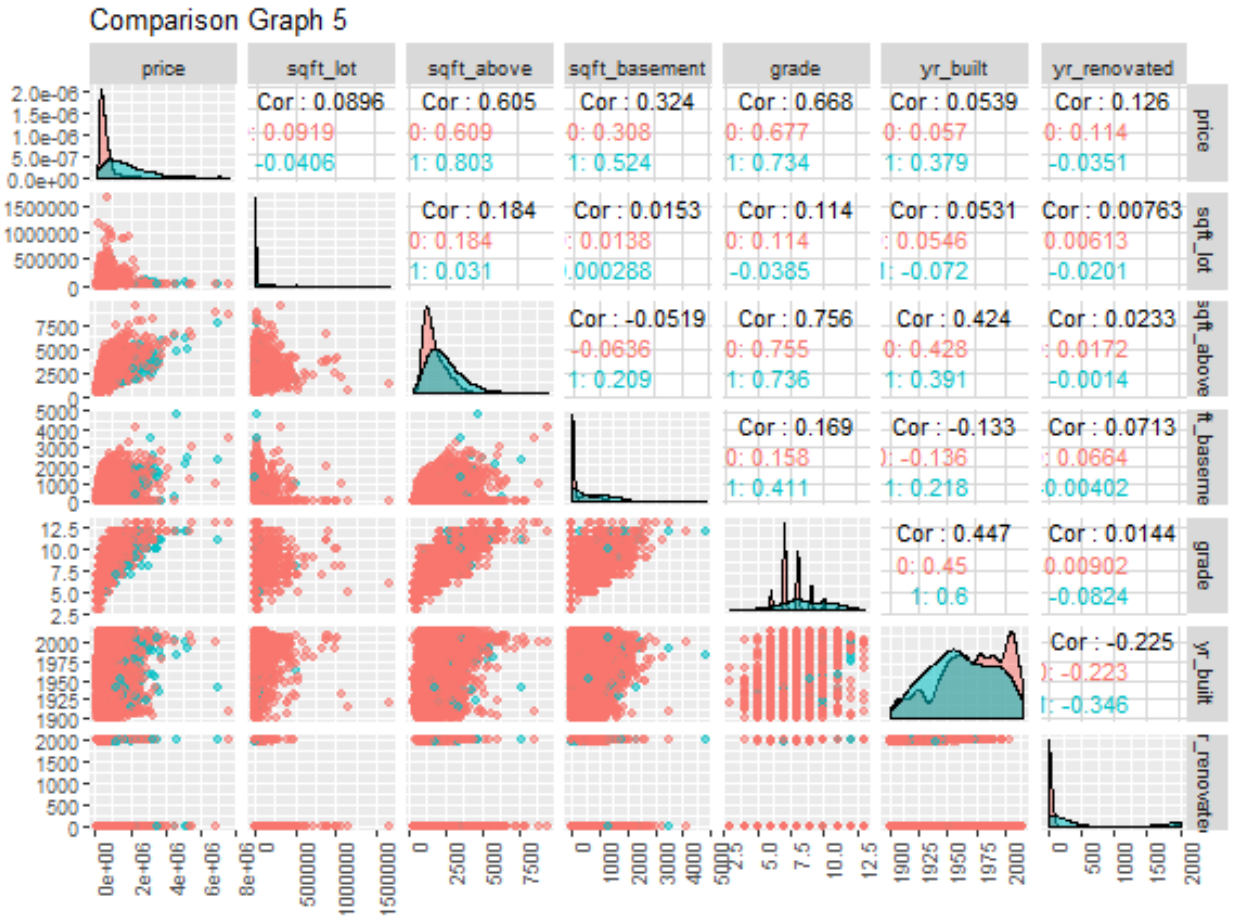



```
ggpairs(data = house[, c("price", "sqft_lot", "sqft_above", "sqft_basement",
    "floors", "waterfront", "condition")], title = "Comparison Graph 4",
    aes(colour = as.factor(house$waterfront), alpha = 0.4)) + theme(axis.text.x = element_text(angle = 9
    hjust = 1))
Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
```

```
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero
```

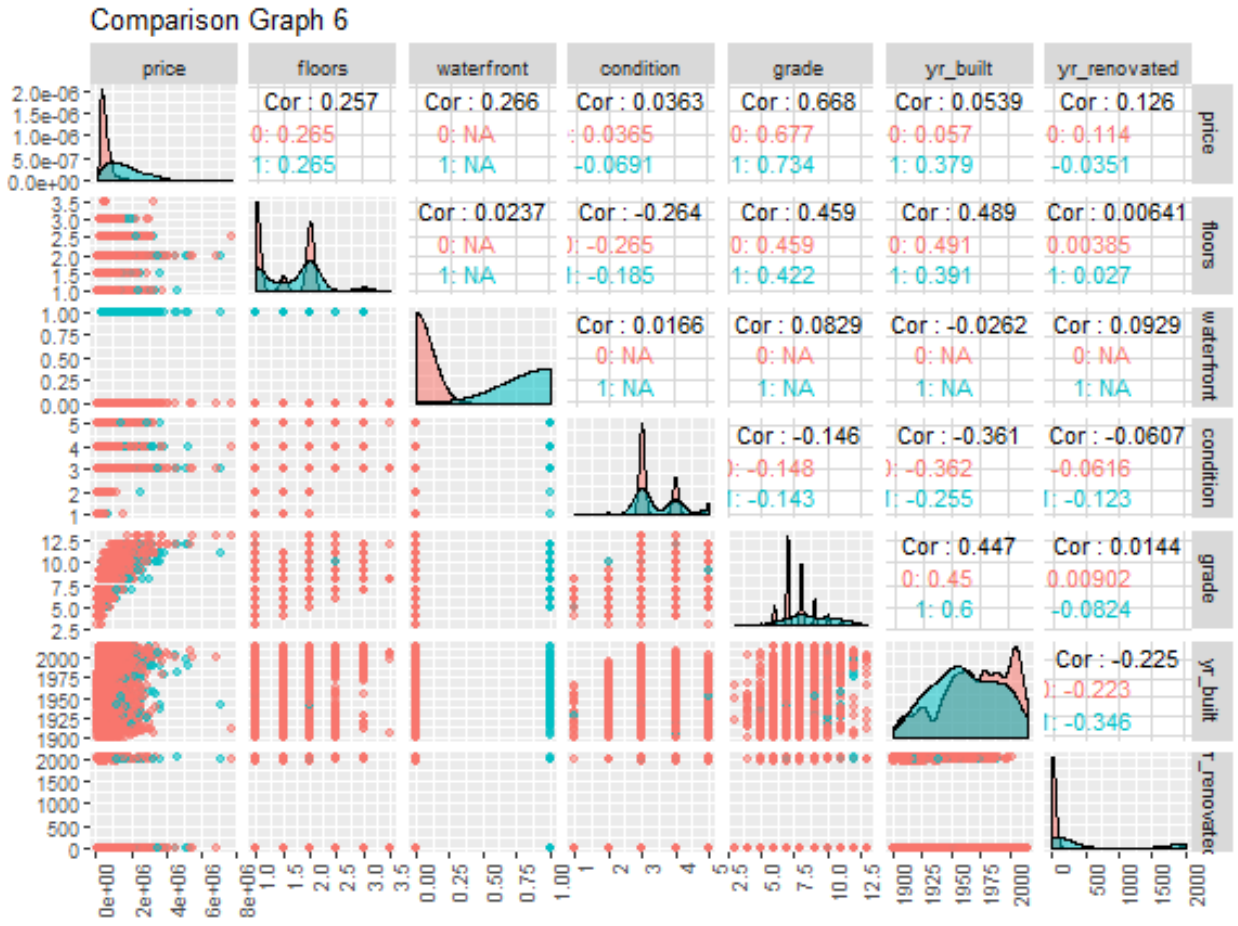



```
ggpairs(data = house[, c("price", "sqft_lot", "sqft_above", "sqft_basement",
    "grade", "yr_built", "yr_renovated")], title = "Comparison Graph 5",
    aes(colour = as.factor(house$waterfront), alpha = 0.4)) + theme(axis.text.x = element_text(angle = 9
    hjust = 1))
```

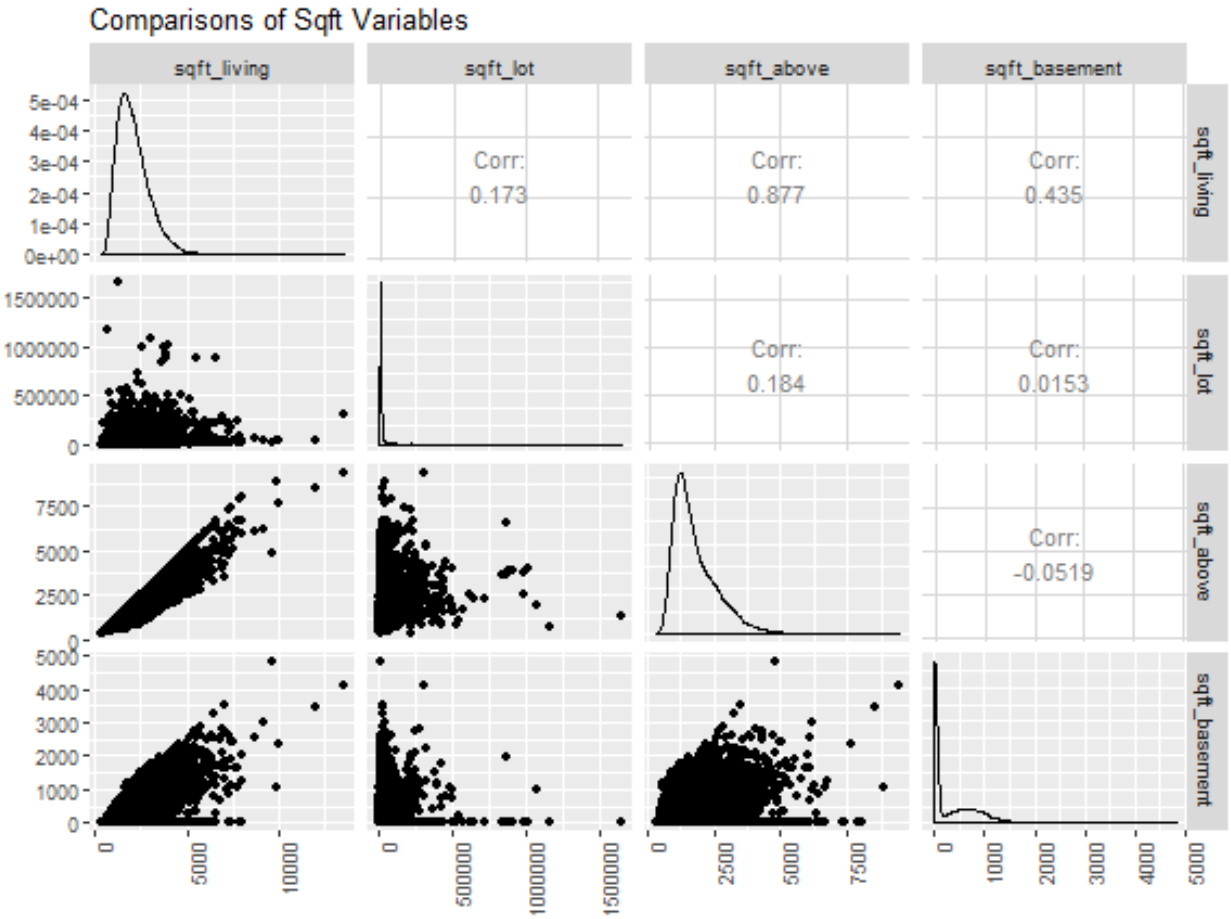



```
ggpairs(data = house[, c("price", "floors", "waterfront", "condition",
    "grade", "yr_built", "yr_renovated")], title = "Comparison Graph 6",
    aes(colour = as.factor(house$waterfront), alpha = 0.4)) + theme(axis.text.x = element_text(angle = 9
    hjust = 1))
Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
```

```
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero

Warning in cor(x, y, method = method, use = use): the standard
deviation is zero
```




```
ggpairs(data = house[, c("sqft_living", "sqft_lot", "sqft_above",
    "sqft_basement")], title = "Comparisons of Sqft Variables") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



In the plot of `Sqft Variables`, `sqft_living` and `sqft_above` are highly correlated with an R-squared value of 0.877. In general, it is not good to have two x-variables that are highly correlated in the model for that might lead to explanatory redundancy.

Part b (2 pts)

If you could have only one predictor for your linear model, based on your scatterplot matrices in part a which variable is the most desirable one to be included in your model? Explain.

The most desirable single predictor for price is `sqft_living`, for it has the highest correlation with price, 0.702, amongst all the possible x-variables in `house` dataset. This means that the x-variable for `sqft_living` explains the most variation of `price`.

Part c (2 pts)

From the scatterplots in Part a we see that there is a house with more than 30 bedrooms; what is the living square footage of this house? Also, there are 7 houses recorded with 0 bedroom and 0 bathroom but with positive values for the square footage. All these 8 records are likely to be errors. Add a chunk of code in Part a before the code for making the matrix scatterplots to remove these 8 observations in `house` and assign the resulting data frame to the same name `house`; this way, your scatterplot matrices will not include these 8 observations. Check the dimension of the new data frame `house` to make sure that the replacement was done correctly.

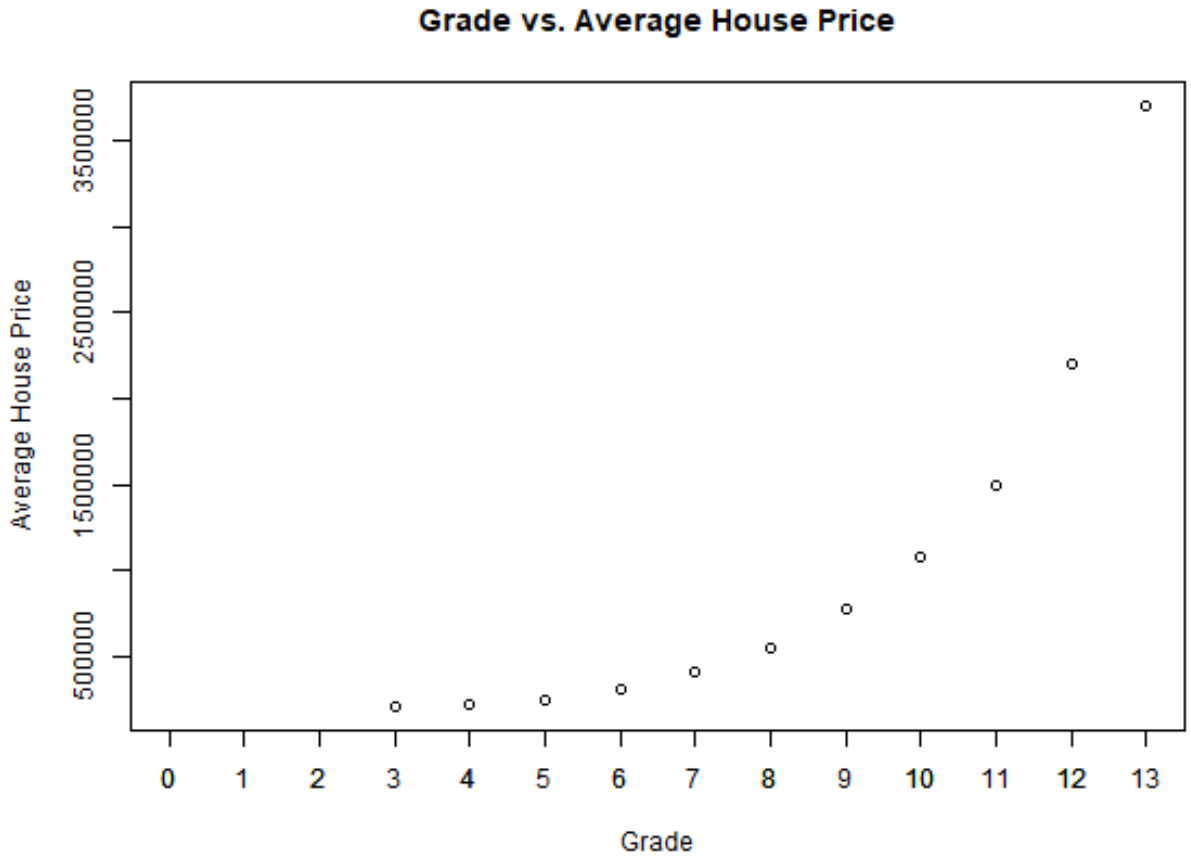
```
# Check dimensions again
dim(house)
[1] 21605    15
```

Looking at Part a, the square footage of the 30+ bedroom house is 1620. The deletion of the erroneous data seems to have been done correctly, for the new dimensions of `house` has eight fewer observations than before (21613 to 21605).

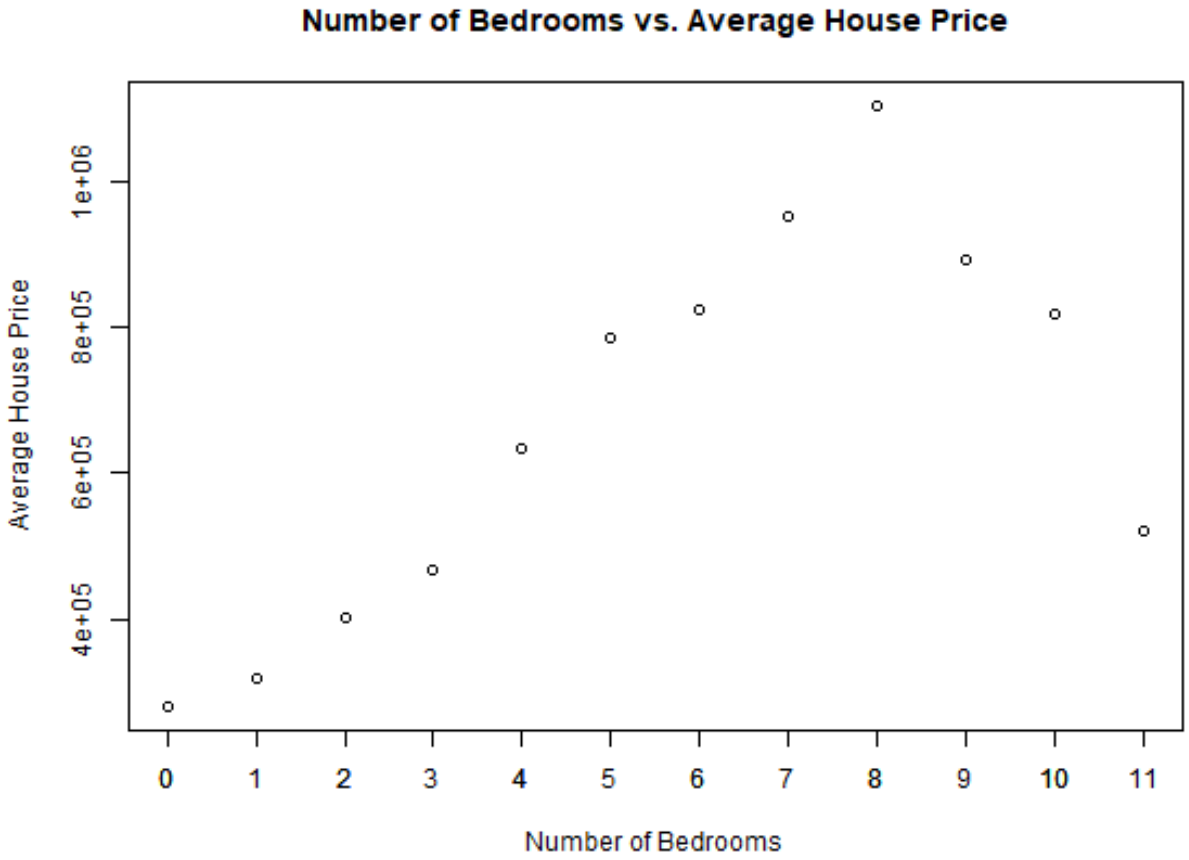
Part d (7 pts)

For each of the variables `grade` and `bedrooms`, calculate the average house price for each unique value of the variable and make a scatterplot for the average house prices v.s. the unique values of the variable; e.g., `grade` takes on the integer values 3 through 13 so your scatterplot for `grade` should have 11 points, the first point should have x-coordinate 3 and y-coordinate the average price for all the houses with `grade` = 3, the second point should have x-coordinate 4 and the y-coordinate the mean price for all the houses with `grade` = 4, and so on. Please include the origin for the graph for `grade`. Do the patterns on the two scatterplots look linear?

```
# Create average house price vectors
prices.grade <- tapply(X = house$price, INDEX = house$grade, FUN = mean)
prices.bedrooms <- tapply(X = house$price, INDEX = house$bedrooms,
    FUN = mean)

# Create Plot of Grade
plot(x = as.numeric(names(prices.grade)), y = prices.grade, main = "Grade vs. Average House Price",
    xlab = "Grade", ylab = "Average House Price", xlim = c(0, max(as.numeric(names(prices.grade)))))

# Set Axis Ticks
axis(side = 1, at = c(0:max(as.numeric(names(prices.grade)))))
```




```
# Create Plot of Bedrooms
plot(x = as.numeric(names(prices.bedrooms)), y = prices.bedrooms,
    main = "Number of Bedrooms vs. Average House Price", xlab = "Number of Bedrooms",
    ylab = "Average House Price")

# Set Axis Ticks
axis(side = 1, at = c(0:max(as.numeric(names(prices.bedrooms)))))
```



The pattern for the `grade` scatterplot looks like there is an exponential relationship between `grade` and the average `price`.

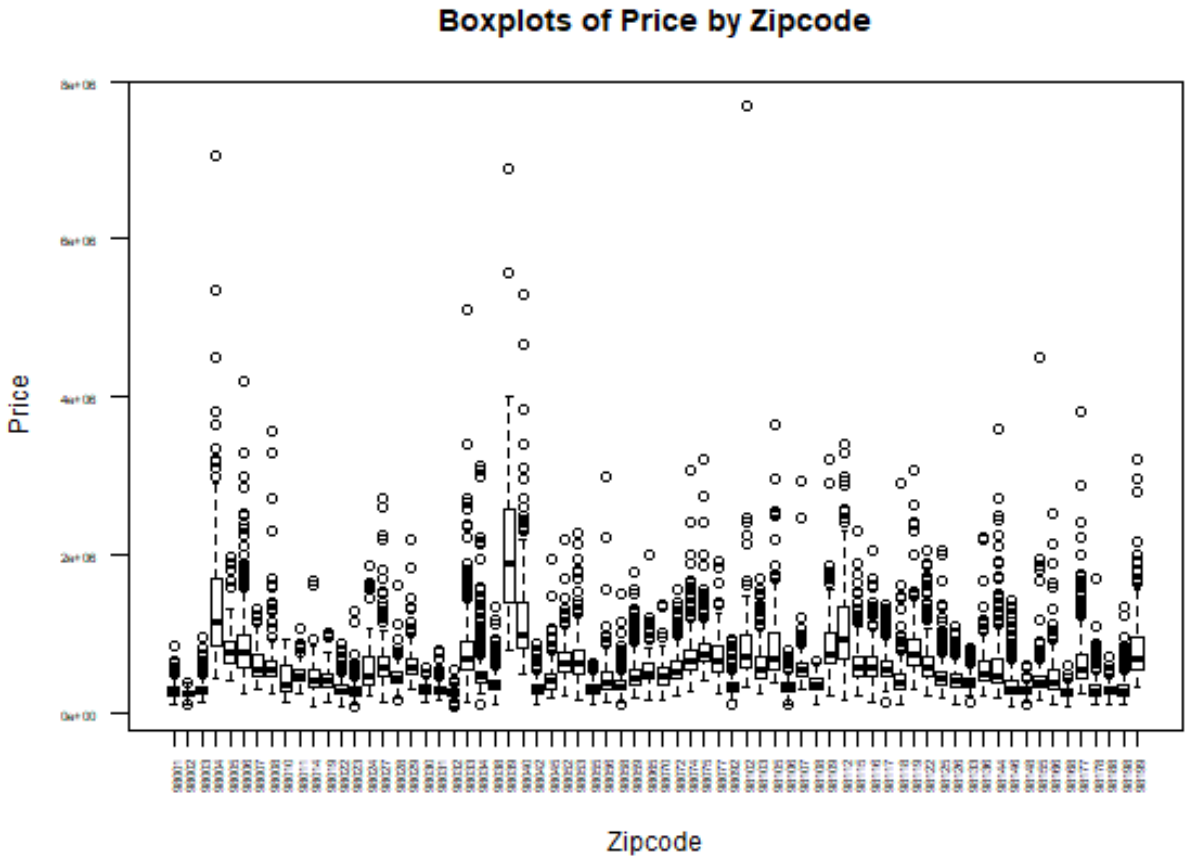
For the `bedroom` scatterplot, it seems there are two different linear relationships between `bedroom` and the average `price`, one positive linear relationship for the `bedroom` range of [0,8], and a second linear relationship for the `bedroom` range of (8,11].

Question 3 (24 pts) Zipcode variable

Part a (5 pts; 1 for answer; 2 for explanation)

Make side by side boxplots to compare the values in `price` by zip code. Based on your boxplots, is it good to include the dummy variables for some of the zip codes in your model? Explain.

```
# Create boxplots
boxplot(data = house, price ~ zipcode, las = 2, cex.axis = 0.5,
    xlab = "Zipcode", ylab = "Price", main = "Boxplots of Price by Zipcode")
```



Yes, it is good to include the dummy variables for some zip codes for `price` seems to have different distributions for different zipcodes.

Part b (8 pts; 1 for each answer, 3 for each interpretation)

Consider the following model (you will need to remove `eval=F` to run the code below):

```
summary(lm(price ~ factor(zipcode), data = house))

Call:
lm(formula = price ~ factor(zipcode), data = house)

Residuals:
     Min       1Q   Median       3Q      Max
-1373107  -126733   -36653    64205  6800605

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            281195      14898  18.875  < 2e-16 ***
factor(zipcode)98002   -46911      24992  -1.877 0.060526 .
factor(zipcode)98003    12916      22541   0.573 0.566646
factor(zipcode)98004  1074732      21788  49.327  < 2e-16 ***
```

```
factor(zipcode)98005   528970      26437  20.009  < 2e-16 ***
factor(zipcode)98006   578490      19567  29.565  < 2e-16 ***
factor(zipcode)98007   335910      28111  11.949  < 2e-16 ***
factor(zipcode)98008   364313      22474  16.210  < 2e-16 ***
factor(zipcode)98010   142471      31988   4.454 8.47e-06 ***
factor(zipcode)98011   209157      25157   8.314  < 2e-16 ***
factor(zipcode)98014   174422      29464   5.920 3.27e-09 ***
factor(zipcode)98019   143594      25371   5.660 1.53e-08 ***
factor(zipcode)98022    34514      23757   1.453 0.146282
factor(zipcode)98023     5538      19558   0.283 0.777065
factor(zipcode)98024   304814      34979   8.714  < 2e-16 ***
factor(zipcode)98027   335796      20407  16.455  < 2e-16 ***
factor(zipcode)98028   181285      22474   8.066 7.61e-16 ***
factor(zipcode)98029   331459      21716  15.264  < 2e-16 ***
factor(zipcode)98030    14993      23129   0.648 0.516835
factor(zipcode)98031    19146      22704   0.843 0.399080
factor(zipcode)98032   -29899      29376  -1.018 0.308791
factor(zipcode)98033   522525      20185  25.887  < 2e-16 ***
factor(zipcode)98034   240458      19209  12.518  < 2e-16 ***
factor(zipcode)98038    85673      18915   4.529 5.95e-06 ***
factor(zipcode)98039  1879412      42714  44.000  < 2e-16 ***
factor(zipcode)98040   913035      22496  40.586  < 2e-16 ***
factor(zipcode)98042    30437      19188   1.586 0.112689
factor(zipcode)98045   158276      24177   6.547 6.02e-11 ***
factor(zipcode)98052   364037      19014  19.145  < 2e-16 ***
factor(zipcode)98053   395440      20501  19.289  < 2e-16 ***
factor(zipcode)98055    23067      22824   1.011 0.312189
factor(zipcode)98056   139696      20477   6.822 9.21e-12 ***
factor(zipcode)98058    72414      19951   3.630 0.000285 ***
factor(zipcode)98059   212358      19828  10.710  < 2e-16 ***
factor(zipcode)98065   247714      21938  11.292  < 2e-16 ***
factor(zipcode)98070   206285      30016   6.872 6.49e-12 ***
factor(zipcode)98072   288764      22704  12.719  < 2e-16 ***
factor(zipcode)98074   404411      20091  20.129  < 2e-16 ***
factor(zipcode)98075   509382      21098  24.143  < 2e-16 ***
factor(zipcode)98077   401580      25033  16.042  < 2e-16 ***
factor(zipcode)98092    53726      21219   2.532 0.011348 *
factor(zipcode)98102   618201      31502  19.624  < 2e-16 ***
factor(zipcode)98103   303633      18849  16.109  < 2e-16 ***
factor(zipcode)98105   581630      23913  24.322  < 2e-16 ***
factor(zipcode)98106    38387      21474   1.788 0.073858 .
factor(zipcode)98107   297859      22873  13.022  < 2e-16 ***
factor(zipcode)98108    74484      25549   2.915 0.003556 **
factor(zipcode)98109   598429      30936  19.344  < 2e-16 ***
factor(zipcode)98112   814305      22800  35.716  < 2e-16 ***
factor(zipcode)98115   338706      18958  17.866  < 2e-16 ***
factor(zipcode)98116   337439      21558  15.652  < 2e-16 ***
factor(zipcode)98117   295600      19153  15.433  < 2e-16 ***
factor(zipcode)98118   136443      19485   7.002 2.59e-12 ***
factor(zipcode)98119   568253      25640  22.163  < 2e-16 ***
factor(zipcode)98122   353165      22322  15.822  < 2e-16 ***
factor(zipcode)98125   188261      20430   9.215  < 2e-16 ***
factor(zipcode)98126   143512      21173   6.778 1.25e-11 ***
```

```
factor(zipcode)98133   105817      19608   5.397 6.86e-08 ***
factor(zipcode)98136   270494      22948  11.787  < 2e-16 ***
factor(zipcode)98144   313353      21344  14.681  < 2e-16 ***
factor(zipcode)98146    78288      22364   3.501 0.000465 ***
factor(zipcode)98148     3714      40344   0.092 0.926659
factor(zipcode)98155   142531      20040   7.112 1.18e-12 ***
factor(zipcode)98166   183037      23182   7.896 3.03e-15 ***
factor(zipcode)98168   -40867      22800  -1.792 0.073078 .
factor(zipcode)98177   394991      23155  17.058  < 2e-16 ***
factor(zipcode)98178    29418      22973   1.281 0.200376
factor(zipcode)98188     7884      28480   0.277 0.781931
factor(zipcode)98198    21684      22541   0.962 0.336079
factor(zipcode)98199   510626      21788  23.436  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 283100 on 21535 degrees of freedom
Multiple R-squared:  0.4074,    Adjusted R-squared:  0.4055
F-statistic: 214.5 on 69 and 21535 DF,  p-value: < 2.2e-16
```

What is the estimate for the y-intercept of the model? Interpret the meaning of the y-intercept. What is the coefficient estimate for the dummy variable for zip code 98004? interpret this number too.

The estimate of the y-intercept of the model is 281195. The y-intercept merely represents the average price of homes that are in the zipcode that does not have a dummy variable associated with it (This happens to be the zipcode with the lowest number in the datset, 98001). The coefficient estimate for the dummy variable for zipcode 98004 is 1074732. This number is how much the average price of homes in the zipcode 98004 is greater than the estimate of the y-intercept.

Part c (4 pts)

The p-value on the third row of the table in the `Coefficients` table is 0.566646. State the Null hypothesis of the test that this p-value is for. Does the p-value suggest an inclusion or exclusion of of the dummy variable $1_{zipcode=98003}$ to the model?

Null Hypoethsis: This dummy variable estimate is equal to 0.

The p-value suggests exclusion of the dummy variable from the model.

Part d (2 pts)

Answer TRUE or FALSE only for the following statement (no explanation is required):

The p-value (56.6665%) in part c above is the estimated chance that the coefficient for the dummy variable for zipcode 98003 is 0; i.e., the p-value is the chance that the Null hypothesis is true.

False.

Part e (5 pts)

For the model shown in Part b how many of the coefficient estimates are significant (i.e., with p-values less than $\alpha = .05$)? Use Bonferroni Correction factor and the method for controlling for False Discovery Rate (FDR) to achieve new cutoffs for the p-values. For each of the two methods, report the adjusted cutoff for the p-values and the number of coefficient estimates that are with p-values less than the new cutoff.

```
# Make vector of p-values
p.v.zip <- summary(lm(price ~ factor(zipcode), data = house))$coefficients[,
    4]
# How many p-values are significant?
sum(p.v.zip < 0.05)
[1] 55
# Bonferroni Correction Factor
sum(p.v.zip < (0.05/length(p.v.zip)))
[1] 53
# Cutoff P-value of Benferroni Correction
max(p.v.zip[p.v.zip < (0.05/length(p.v.zip))])
[1] 0.000465191
# FDR
sort.p.zip <- sort(p.v.zip)
length(sort.p.zip[sort.p.zip <= ((1:length(sort.p.zip)/length(sort.p.zip))) *
    0.05])
[1] 55
max(sort.p.zip[sort.p.zip <= ((1:length(sort.p.zip)/length(sort.p.zip))) *
    0.05])
[1] 0.01134761
```

For the Bonferroni Correction Factor, 53 p-values are now significant, with a maximum p-value cutoff of 0.01134761. For the False Discovery Rate, 55 p-values are significant with a new p-value cutoff of 0.01134761.

Transforming some of the variables and creating additional ones

Question 4 (9 pts)

As we saw in question 2 some of the variables do not have a linear relationship with `price`. We will transform these variables in this question.

Part a (4 pts)

From the scatterplot for the average house prices v.s. the unique values in `grade` in question 2.d we see that the relationship between the two variables could be approximated by the equation

$$price = (grade)^b$$

for some constant b.

Taking log on both sides results in

$$log(price) = b \times log(grade)$$

Find out what the best value for b should be according to the data; this value should minimize $(log(price_i) - b \times log(grade_i))^2$ for the data points on average.

```
log.price <- log(house$price)
log.grade <- log(house$grade)
summary(lm(log.price ~ log.grade - 1))
```

```
Call:
lm(formula = log.price ~ log.grade - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4556 -0.4249  0.0314  0.4943  5.4853

Coefficients:
          Estimate Std. Error t value Pr(>|t|)
log.grade 6.423823   0.002374    2706   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7083 on 21604 degrees of freedom
Multiple R-squared:  0.9971,    Adjusted R-squared:  0.9971
F-statistic: 7.321e+06 on 1 and 21604 DF,  p-value: < 2.2e-16
```

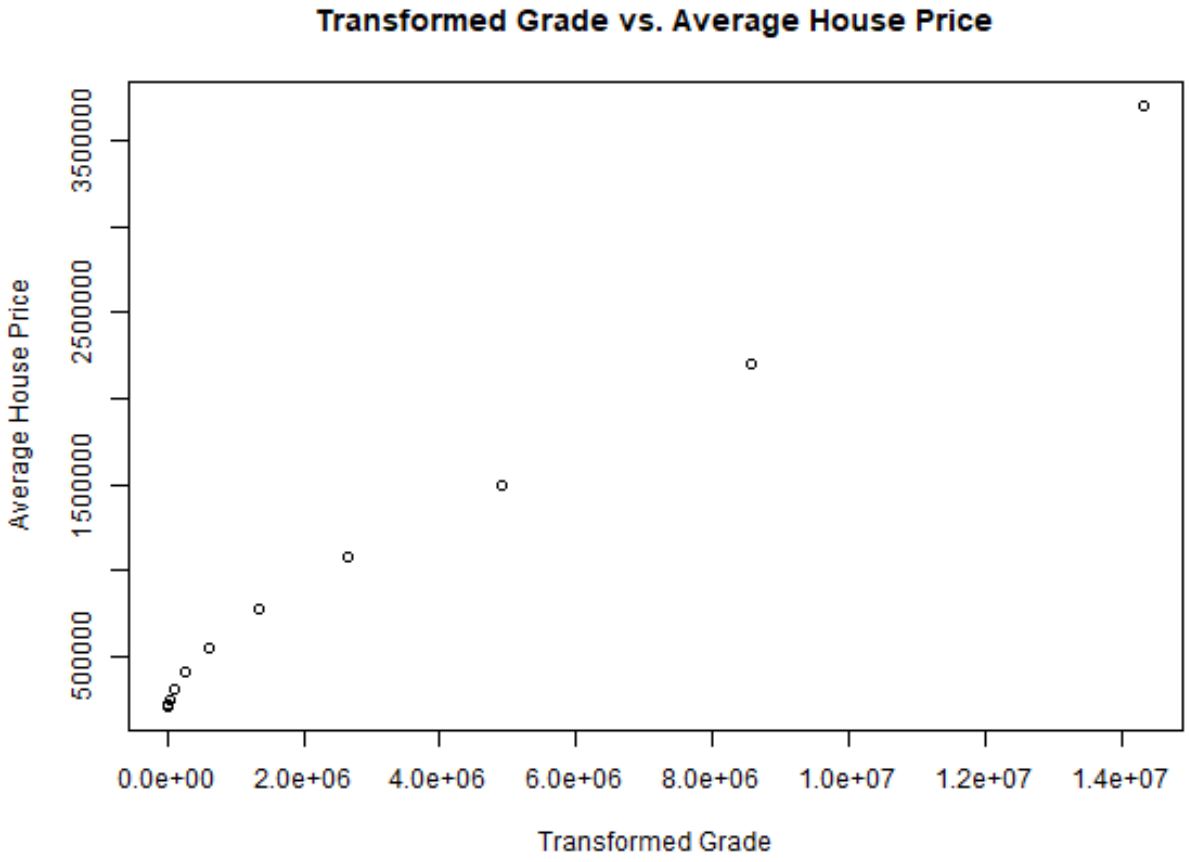
The best value for `b` should be 6.4238

Part b (2 pts)

Use the value that you found for b in part a and create the variable `trans.grade` by transforming `grade` into $trans.grade = (grade)^b$.

Make the scatterplot that you made for the average house prices v.s. the unique values in `grade` in question 2.d again, except that now replace the unique values of `grade` with that of `trans.grade`. Verify that the scatterplot now shows a more linear pattern.

```
trans.grade = house$grade^summary(lm(log.price ~ log.grade - 1))$coefficients[1]

# Create Plot of Grade
plot(x = sort(unique(trans.grade)), y = prices.grade, main = "Transformed Grade vs. Average House Price
    xlab = "Transformed Grade", ylab = "Average House Price", xlim = c(0,
        max(trans.grade)))
```



Part c (3 pts) Creating new variables for the model

Here, we will make a new data frame for all the variables that we will need for building the model.

We will create the new data frame `mod.variables` with all the transformed variables and some of the original variables. `mod.variables` should include `price`, `sqft_living`, `sqft_basement`, `grade`, `bedrooms`, `bathrooms` and `date` from the data frame `house` plus the following transformed variables:

- `f.waterfront`: the factor version of `house$waterfront`;
- `f.floor`: the factor version of `house$floor`;
- `f.cond`: the factor version of `house$condition`;
- `f.renov`: a factor vector and each element in `f.renov` is 1 if the corresponding element in `house$yr_renovated` does not equal to zero, and 0 otherwise;
- `f.bdrm.less.eq.8`: a factor vector and each element in `f.bdrm.less.eq.8` is 1 if the corresponding element in `house$bedrooms` is less or equal to 8, and 0 otherwise;
- `f.zipcode`: the factor version of `house$zipcode`;

and also the transformed variable

- `trans.grade`: the vector `trans.grade` that you have already made.

```
# Create new dataframe `mod.variables`
mod.variables <- data.frame(price = house$price, sqft_living = house$sqft_living,
    sqft_basement = house$sqft_basement, grade = house$grade, bedroom = house$bedroom,
    bathroom = house$bathroom, date = house$date, f.waterfront = as.factor(house$waterfront),
    f.floor = as.factor(house$floor), f.cond = as.factor(house$condition),
    f.renov = as.factor(as.numeric(house$yr_renovated != 0)), f.bdrm.less.eq.8 = as.factor(as.numeric(h
        8)), f.zipcode = as.factor(house$zipcode), trans.grade = trans.grade)
```

Divide data into three subsets: one training set and two test sets.

For this part you just need to remove the `eval=F` argument for each of the code chunks below and run the code; no action is required from you other than that.

`mod.variables` covers the period from May 2014 to May 2015. We will prepare a vector `date.format` that we can use to extract out the observations that correspond to transactions closed in May 2015.

```
# Prepare date vector
date.format = format(as.Date(mod.variables$date, "%Y%m%dT"), "%Y%m")
length(date.format)
[1] 21605
head(date.format)
[1] "201410" "201412" "201502" "201412" "201502" "201405"
dim(mod.variables)
[1] 21605    14
```

Then, the following lines will extract out all the observations with transactions closed in May 2015. We will use these observations for our out-of-time test set `test2`. `test2` should be 646 by 13.

```
# Create out of time test set
test2 = mod.variables[date.format %in% "201505", !(names(mod.variables) %in%
    c("date"))]

# Check dimensions
dim(test2)
[1] 646  13
```

For the rest of the observations run the following code chunk; this will split the remaining observations into 2 sets: 85% training set and 15% in-time test set.

```
tmp = mod.variables[!date.format %in% "201505", !(names(mod.variables) %in%
    c("date"))]

s = dim(tmp)[1]
s
[1] 20959

set.seed(2012018)
permu = sample(1:s)
train = tmp[permu, ][1:round(s * 0.85), ]
dim(train)
[1] 17815    13
```

```
test1 = tmp[permu, ][(round(s * 0.85) + 1):s, ]
dim(test1)
[1] 3144   13
```

Model selection

We are now ready to build our model! We will use BIC and Adjusted R^2 as our criteria for our model selection. `mod.variables` has been divided into 3 sets:

- `train.val`: The 17815 by 13 training plus validation set
- `test1`: The 3144 by 13 in-time test set
- `test2`: The 646 by 13 out-of-time test set

For the rest of the project we will build our model with the training data, and test the performance of our chosen model on the 2 test sets.

Question 5 (27 pts)

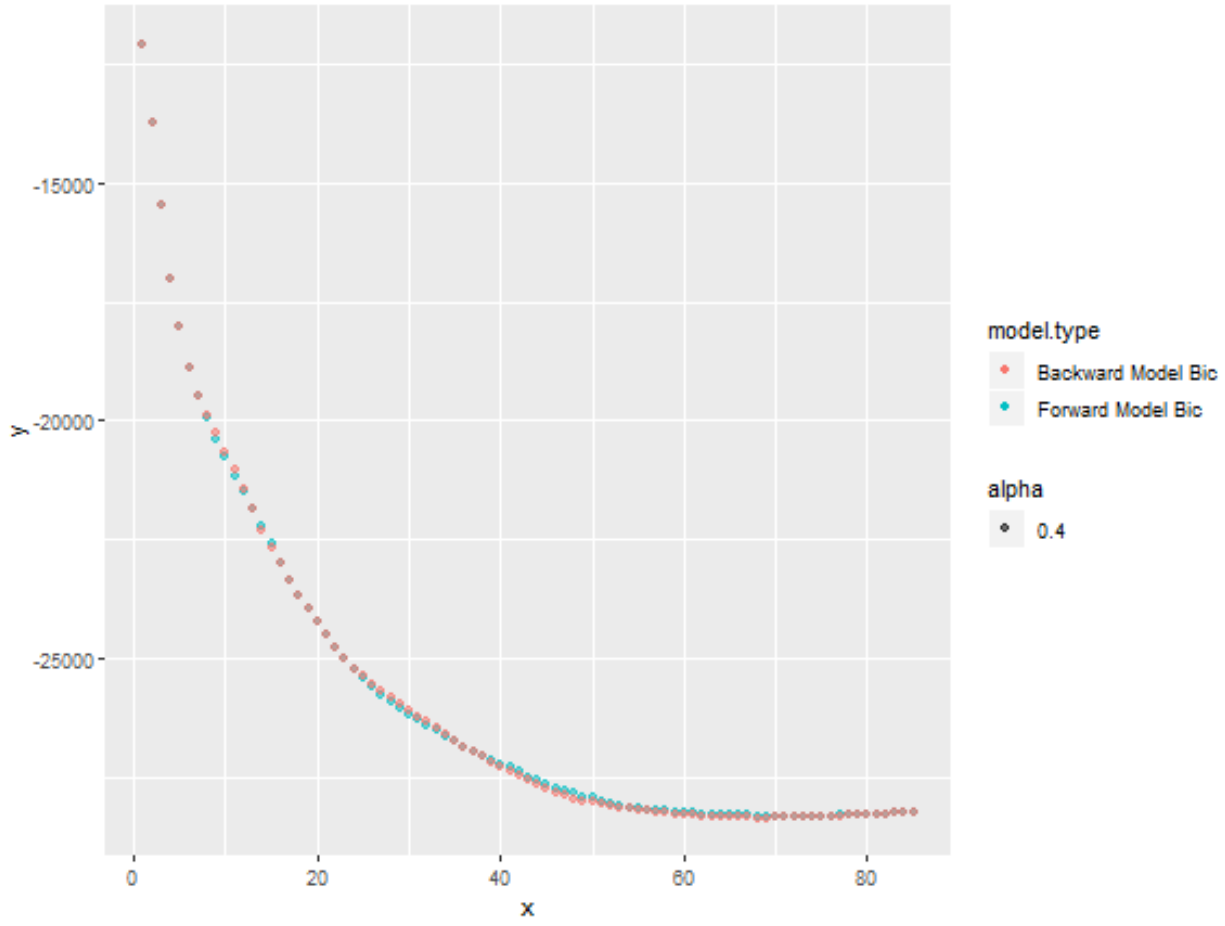
Part a (7 pts) Consider models with the original data only

In this part please consider only the predictors that already exist in the original dataset (i.e., all the variables in `train` except `f.bdrm.less.eq.8` and `trans.grade`).

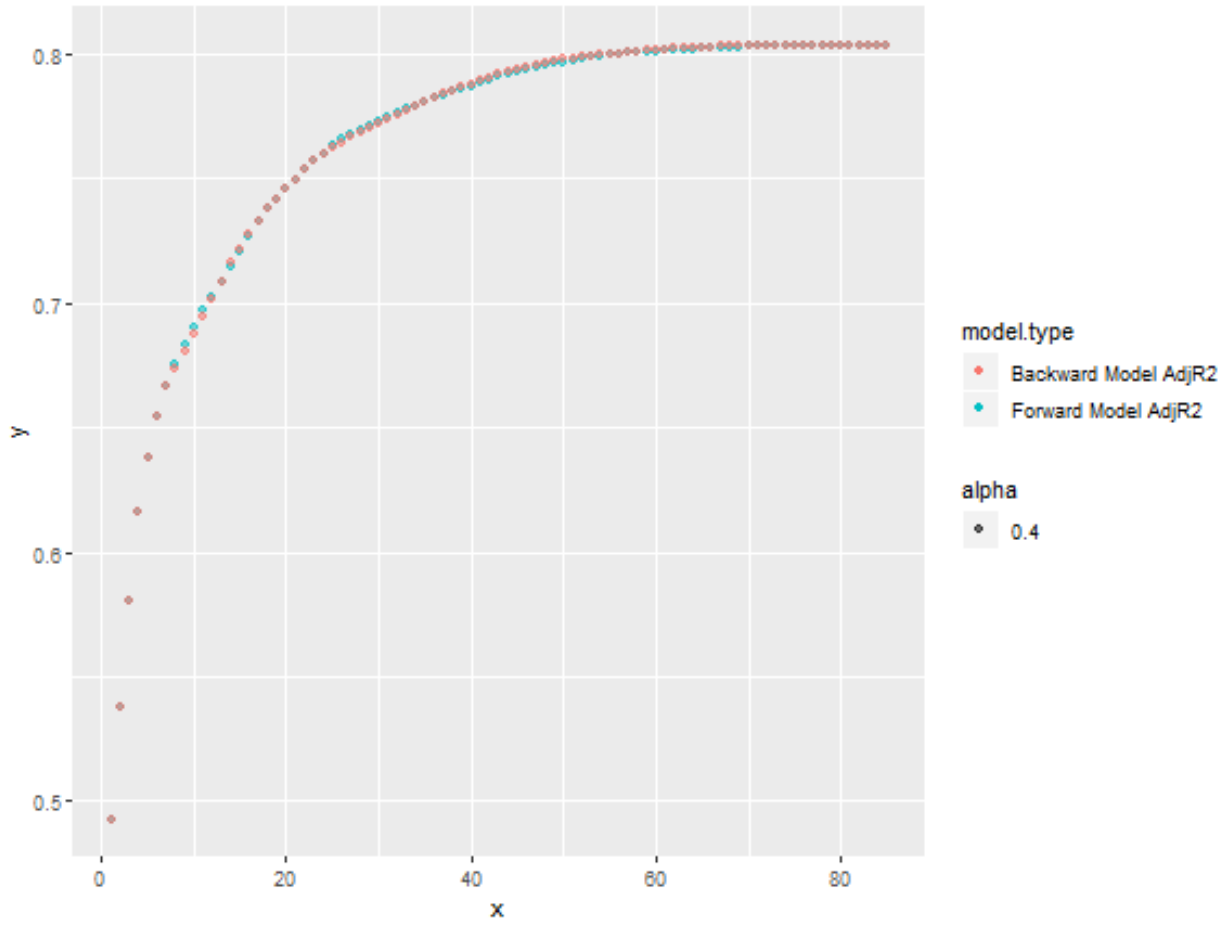
Use BIC and Adjusted R^2 as the criteria to evaluate the performance of your "best" models. Do backward and forward selection algorithms produce similar (in terms of the values for BIC and Adjusted R^2) results in this case?

```
model.length <- length(lm(price ~ . + f.waterfront + f.floor + f.cond +
    f.renov + f.zipcode - trans.grade - f.bdrm.less.eq.8, data = train)$coef) -
    1

library("leaps")

model.forward <- regsubsets(price ~ . + f.waterfront + f.floor +
    f.cond + f.renov + f.zipcode - trans.grade - f.bdrm.less.eq.8,
    data = train, method = "forward", nvmax = model.length)

model.backward <- regsubsets(price ~ . + f.waterfront + f.floor +
    f.cond + f.renov + f.zipcode - trans.grade - f.bdrm.less.eq.8,
    data = train, method = "backward", nvmax = model.length)

df.models <- data.frame(y = c(summary(model.forward)$bic, summary(model.backward)$bic,
    summary(model.forward)$adjr2, summary(model.backward)$adjr2),
    x = c(rep(1:model.length, 4)), model.type = c(rep("Forward Model Bic",
        model.length), rep("Backward Model Bic", model.length),
        rep("Forward Model AdjR2", model.length), rep("Backward Model AdjR2",
            model.length)))
```

```
ggplot(df.models[df.models$model.type == "Backward Model Bic" |
    df.models$model.type == "Forward Model Bic", ]) + geom_point(aes(x = x,
    y = y, color = model.type, alpha = 0.4))
```




```
ggplot(df.models[df.models$model.type == "Backward Model AdjR2" |
    df.models$model.type == "Forward Model AdjR2", ]) + geom_point(aes(x = x,
    y = y, color = model.type, alpha = 0.4))
```



Looking at the plots of the Bic and Adjusted R-squared values of the forward and backward models, there seems to be little difference in terms of either metric between the 'best' models for any given number of predictors.

Part b (2 pts)

Print out the coefficient estimates for the best model with 7 predictors for each algorithm; is `f.waterfront1` included in both models?

```
coef(model.forward, id = 7)
   (Intercept)    sqft_living          grade  f.waterfront1
  -494755.0373       162.1336     87229.4192    893290.1133
f.zipcode98004 f.zipcode98039 f.zipcode98040 f.zipcode98112
   605666.9138   1115386.5017    355140.8267    427776.8863
coef(model.backward, id = 7)
   (Intercept)    sqft_living          grade  f.waterfront1
  -494755.0373       162.1336     87229.4192    893290.1133
f.zipcode98004 f.zipcode98039 f.zipcode98040 f.zipcode98112
   605666.9138   1115386.5017    355140.8267    427776.8863
```

Yes, `f.waterfront1` is included in both models.

Part c (3 pts)

Go back to modify your matrices of scatterplots in question 2.a to color the points by the case whether the house has a view to a waterfront. Note that now the plots indicates that it might be good to consider the following interaction terms for your model: `f.waterfront:sqft_living`, `f.waterfront:bathrooms`. Also, from question 2.d we see that it would be good to consider `(f.bdrm.less.eq.8:bedrooms)` too.

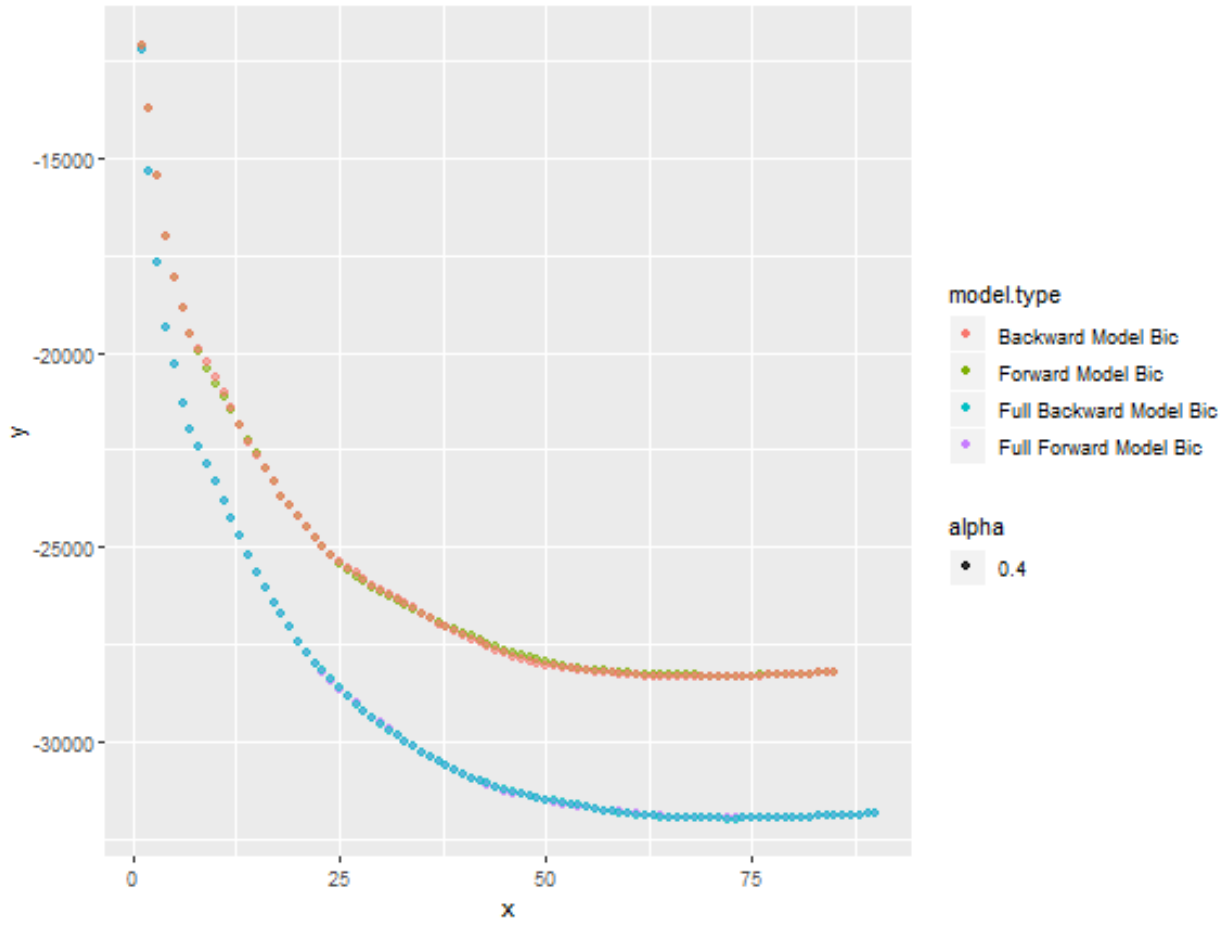
Part d (5 pts) Consider models with the additional variables

In this part we want to see if including additional variables that are not defined in the original dataset will improve the performance of the model. Repeat part a but now consider all the variables in the data frame `train` plus the interaction terms `f.waterfront:sqft_living`, `f.waterfront:bathrooms` and `f.bdrm.less.eq.8:bedrooms`.

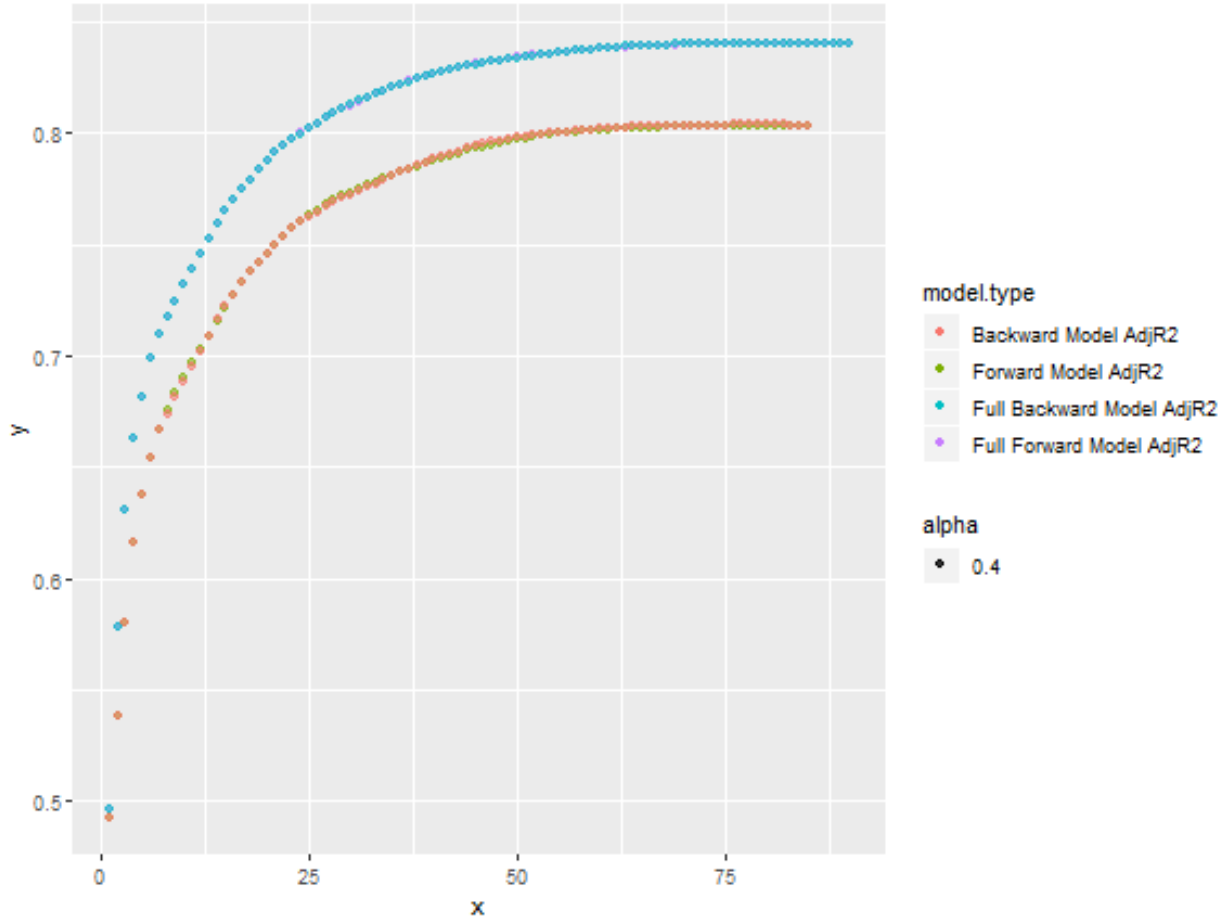
Pick the results from the better performing algorithm (if both algorithms perform similarly just pick either one) in part a and add the results to the plots for BIC and Adjusted R^2 for the current models too; this will allow you to see how much the performance of each of the best models improves by considering the additional variables.

As you can see from the graphs creating the right variables helps to improve your model significantly!

```
num.var.full <- length(lm(price ~ . + f.waterfront + f.floor + f.cond +
    f.renov + f.zipcode + f.bdrm.less.eq.8 + f.waterfront:sqft_living +
    f.waterfront:bathroom + f.bdrm.less.eq.8:bedroom, data = train)$coef) -
    1

model.forward.full <- regsubsets(price ~ . + f.waterfront + f.floor +
    f.cond + f.renov + f.zipcode + f.bdrm.less.eq.8 + f.waterfront:sqft_living +
    f.waterfront:bathroom + f.bdrm.less.eq.8:bedroom, data = train,
    method = "forward", nvmax = num.var.full)

model.backward.full <- regsubsets(price ~ . + f.waterfront + f.floor +
    f.cond + f.renov + f.zipcode + f.bdrm.less.eq.8 + f.waterfront:sqft_living +
    f.waterfront:bathroom + f.bdrm.less.eq.8:bedroom, data = train,
    method = "backward", nvmax = num.var.full)

df.models.full = data.frame(y = c(summary(model.forward.full)$bic,
    summary(model.backward.full)$bic, summary(model.forward.full)$adjr2,
    summary(model.backward.full)$adjr2), x = c(rep(1:num.var.full,
    4)), model.type = c(rep("Full Forward Model Bic", num.var.full),
    rep("Full Backward Model Bic", num.var.full), rep("Full Forward Model AdjR2",
        num.var.full), rep("Full Backward Model AdjR2", num.var.full)))

ggplot(df.models.full[df.models.full$model.type == "Full Backward Model Bic" |
    df.models.full$model.type == "Full Forward Model Bic", ]) +
    geom_point(aes(x = x, y = y, color = model.type, alpha = 0.4)) +
    geom_point(data = df.models[df.models$model.type == "Backward Model Bic" |
        df.models$model.type == "Forward Model Bic", ], aes(x = x,
        y = y, color = model.type, alpha = 0.4))
```




```
ggplot(df.models.full[df.models.full$model.type == "Full Backward Model AdjR2" |
    df.models.full$model.type == "Full Forward Model AdjR2", ]) +
    geom_point(aes(x = x, y = y, color = model.type, alpha = 0.4)) +
    geom_point(data = df.models[df.models$model.type == "Backward Model AdjR2" |
        df.models$model.type == "Forward Model AdjR2", ], aes(x = x,
        y = y, color = model.type, alpha = 0.4))
```



Part e (2 pt)

What is the purpose of using measures, such as BIC or Adjusted R-square, to evaluate the performance of a model? Why don't we just use RSS instead?

The purpose of using more advances measures such as BIC and Adjusted R-square is to reduce the chance of overfitting. Both of these measurements penalize the use of too many predictors, which RSS doesn't do.

Part f (4 pts: 3pts for the choice and 1 for the print out) The champion model

Propose your champion model and justify your choice. Print out the coefficient estimates for your model.

```
min(summary(model.backward.full)$bic)
[1] -31996.21

min(summary(model.forward.full)$bic)
[1] -31984.11

which.min(summary(model.backward.full)$bic)
[1] 73

coef(model.backward.full, id = which.min(summary(model.backward.full)$bic))
```

```
          (Intercept)           sqft_living
        -3.834670e+05          1.701558e+02
         sqft_basement              bathroom
        -3.952260e+01          1.829877e+04
         f.waterfront1              f.floor2
        -2.124835e+05         -3.643444e+04
            f.floor2.5              f.floor3
         6.236239e+04         -8.321621e+04
                f.cond4              f.cond5
         3.244811e+04          7.820127e+04
                f.renov1     f.bdrm.less.eq.81
         6.983655e+04          3.198346e+05
        f.zipcode98004        f.zipcode98005
         7.384613e+05          2.905656e+05
        f.zipcode98006        f.zipcode98007
         2.472365e+05          2.334408e+05
        f.zipcode98008        f.zipcode98010
         2.545519e+05          5.057717e+04
        f.zipcode98011        f.zipcode98014
         1.200722e+05          1.030834e+05
        f.zipcode98019        f.zipcode98023
         8.217335e+04         -4.330957e+04
        f.zipcode98024        f.zipcode98027
         1.664390e+05          1.492438e+05
        f.zipcode98028        f.zipcode98029
         1.266475e+05          2.060299e+05
        f.zipcode98033        f.zipcode98034
         3.464777e+05          1.884824e+05
        f.zipcode98038        f.zipcode98039
         2.748814e+04          1.184619e+06
        f.zipcode98040        f.zipcode98045
         4.844281e+05          9.443042e+04
        f.zipcode98052        f.zipcode98053
         2.217685e+05          1.993487e+05
        f.zipcode98055        f.zipcode98056
         3.518707e+04          8.410252e+04
        f.zipcode98059        f.zipcode98065
         6.969822e+04          1.007930e+05
        f.zipcode98070        f.zipcode98072
         9.883942e+04          1.442673e+05
        f.zipcode98074        f.zipcode98075
         1.685156e+05          1.662235e+05
        f.zipcode98077        f.zipcode98092
         1.095847e+05         -3.277016e+04
        f.zipcode98102        f.zipcode98103
         5.045550e+05          3.341012e+05
        f.zipcode98105        f.zipcode98106
         4.637457e+05          1.086031e+05
        f.zipcode98107        f.zipcode98108
         3.443664e+05          1.048291e+05
        f.zipcode98109        f.zipcode98112
         5.138048e+05          5.913112e+05
        f.zipcode98115        f.zipcode98116
```

```
             3.339462e+05                3.084508e+05
          f.zipcode98117              f.zipcode98118
             3.163935e+05                1.601251e+05
          f.zipcode98119              f.zipcode98122
             4.974673e+05                3.298756e+05
          f.zipcode98125              f.zipcode98126
             2.031170e+05                1.904924e+05
          f.zipcode98133              f.zipcode98136
             1.474208e+05                2.645969e+05
          f.zipcode98144              f.zipcode98146
             2.890333e+05                1.075490e+05
          f.zipcode98155              f.zipcode98166
             1.296301e+05                5.454169e+04
          f.zipcode98168              f.zipcode98177
             4.488097e+04                2.384070e+05
          f.zipcode98178              f.zipcode98199
             4.819452e+04                3.938892e+05
              trans.grade sqft_living:f.waterfront1
             1.020992e-01                2.720194e+02
  bathroom:f.waterfront1 bedroom:f.bdrm.less.eq.81
             6.801565e+04               -1.241958e+04
```

Fix this

Part g (4 pts)

Let k be the number of predictors that you chose for your champion model. Compare the best models with k predictors selected by the backward and forward algorithms. Do the algorithms pick the same set of predictors? Are there any predictors that are selected by the forward algorithm but not by the backward? List these variables if there are any. Similarly, are there any predictors that are selected by the backward algorithm but not by the forward? List the variables too if there are any.

```
# Establish K
k = as.numeric(which.min(summary(model.backward.full)$bic))
k
[1] 73

# Variables in backward model not in forward model
names(coef(model.backward.full, id = k))[!names(coef(model.backward.full,
    id = k)) %in% names(coef(model.forward.full, id = k))]
[1] "f.zipcode98010"            "f.zipcode98038"
[3] "f.zipcode98055"            "bedroom:f.bdrm.less.eq.81"

# Variables in forward model not in backword model
names(coef(model.forward.full, id = k))[!names(coef(model.forward.full,
    id = k)) %in% names(coef(model.backward.full, id = k))]
[1] "bedroom"        "f.cond3"        "f.zipcode98003"
[4] "f.zipcode98042"
```

The variables in the backward that are not in the forward model are “f.zipcode98010”, “f.zipcode98038”, “f.zipcode98055”, and “bedroom:f.bdrm.less.eq.81”. The variables in the forward model that are not in the backward model are “f.cond3”, “f.zipcode98003”, “f.zipcode98042”, and “bedroom”.

Question 6 (10 pts)

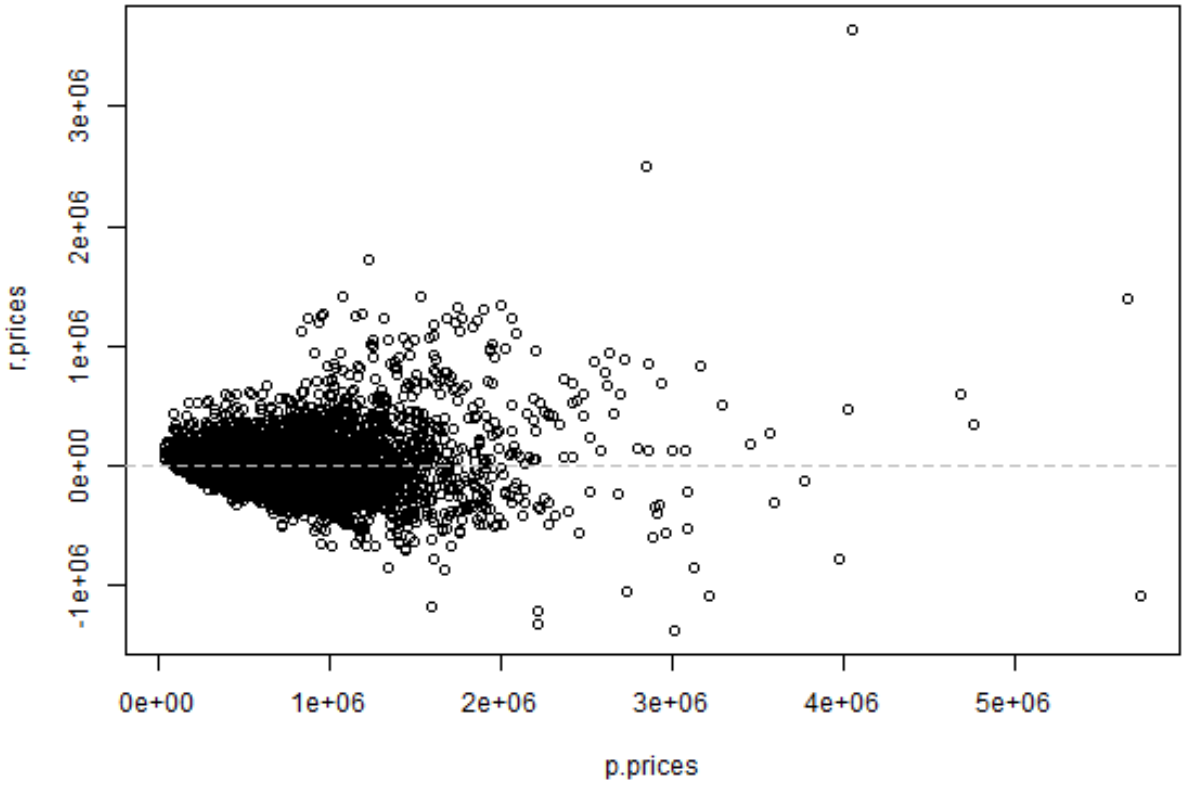
Part a (6 pts) Checking assumptions

For your champion model plot the residual plot, the qq plot and the histogram for the distribution of the residuals. What are the assumptions on the distribution of the errors for a linear model? Do your plots suggest any possible violation of the assumptions? Explain.

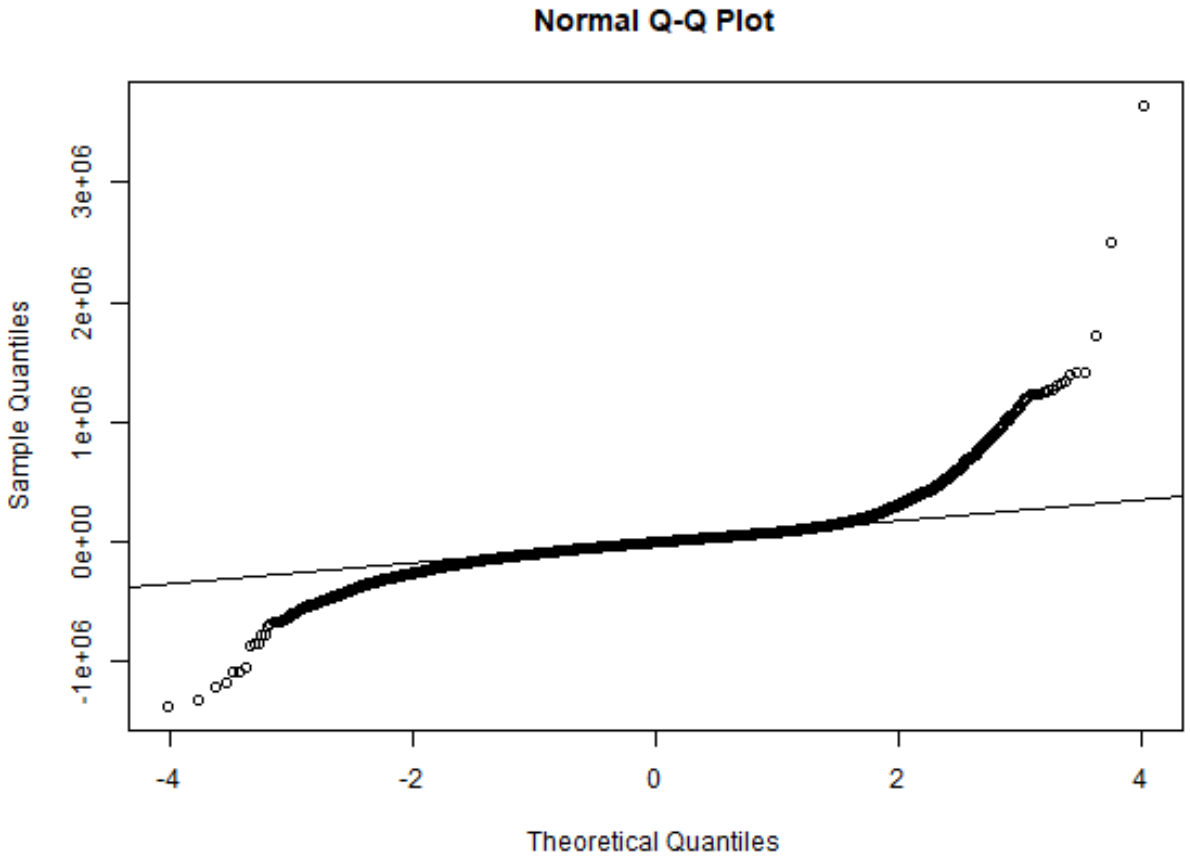
You can use the following function `predict.reg` to make prediction on a dataset with a model produced by `regsubsets()`. It works similarly to the `predict()` function except that it has an extra input argument `id` that specifies which `best` model you would like to use for making the predictions. E.g., say `g` is an output of `regsubsets()`, then `predict.reg(object = g,newdata = d,id = 5)` predicts for the data in dataset `d` by using the best model in `g` for 5 predictors.

You do not need to understand how the function is being coded

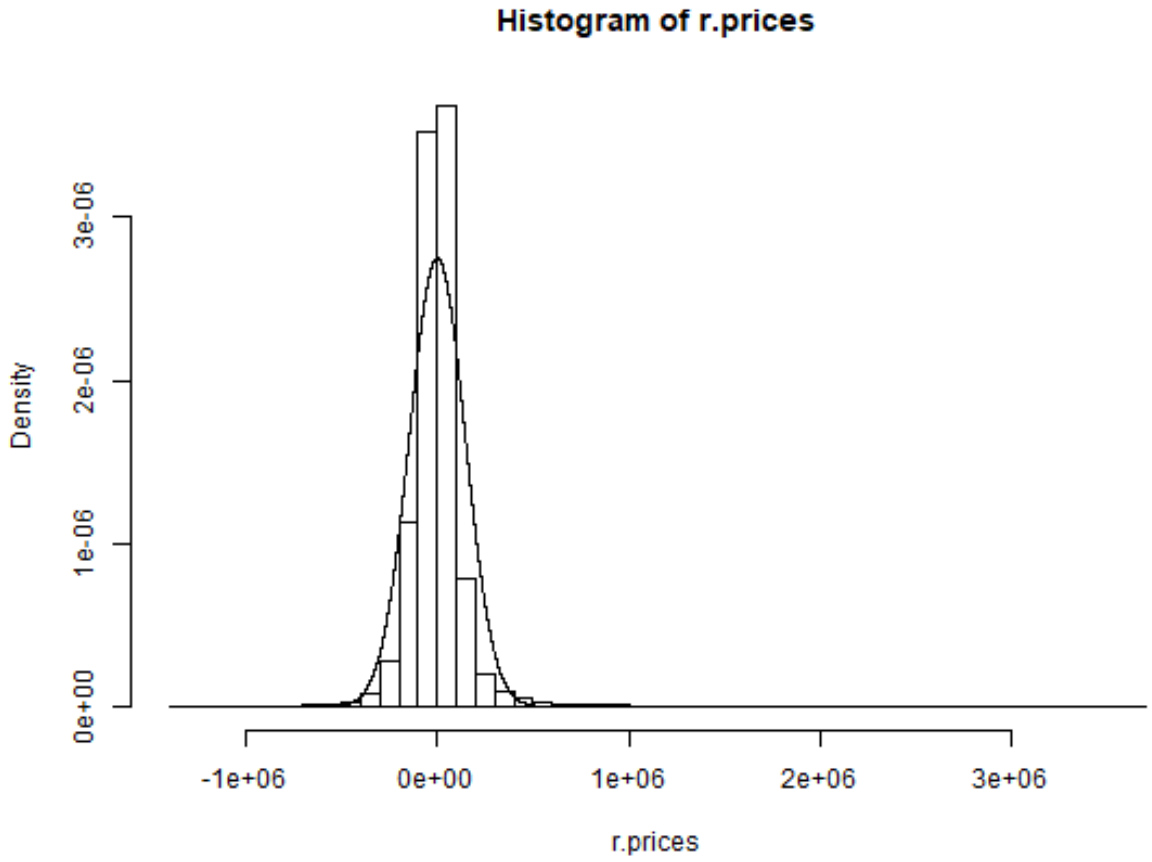
```
predict.reg = function(object, newdata, id) {
    form = as.formula(object$call[[2]])  # Extract the formula used when we called regsubsets()
    mat = model.matrix(form, newdata)  # Build the model matrix
    coefi = coef(object, id = id)  # Extract the coefficients of the id'th model
    xvars = names(coefi)  # Pull out the names of the predictors used in the ith model
    mat[, xvars] %*% coefi  # Make predictions using matrix multiplication
}
p.prices <- predict.reg(object = model.backward.full, newdata = train,
    id = as.numeric(which.min(summary(model.backward.full)$bic)))

r.prices <- train$price - p.prices

plot(x = p.prices, y = r.prices)
abline(a = 0, b = 0, lty = 2, col = "grey")
```




```
qqnorm(y = r.prices)
qqline(y = r.prices)
```




```
hist(r.prices, breaks = 60, freq = F)
r.prices.seq = seq(from = min(r.prices), to = max(r.prices), length = length(r.prices))
lines(y = dnorm(r.prices.seq, mean = mean(r.prices), sd = sd(r.prices)),
    x = r.prices.seq)
```



For a linear model, we assume that the errors are independent, and that the errors are normally distributed with a constand standard deviation. The plot of the residuals suggests that there is a non-constant standard deviation, which would indicate there is a violation of our assumptions. The qq-plot and histogram reaffirm this observation, as we do not observe a straight line with a slope of 1 in the qq-plot and we can see that the distribution of the residuals is especially concentrated in the middle in the histogram.

Part b (4 pts)

Estimate the mean squared errors for your champion model with the in-time and out-of-time test sets. Is the MSE estimated with the out-of-time test set bigger or smaller than the MSE estimated with the in-time test set? Is this result expected?

```
p.intime <- predict.reg(object = model.backward.full, newdata = test1,
    id = as.numeric(which.min(summary(model.backward.full)$bic)))

p.outtime <- predict.reg(object = model.backward.full, newdata = test2,
    id = as.numeric(which.min(summary(model.backward.full)$bic)))

mse.intime <- mean(p.intime^2)

mse.outtime <- mean(p.outtime^2)
```

```
mse.intime/mse.outtime
[1] 1.11938
```

The out of time test has a smaller mean squared error than the in time test. This is not expected, for you would expect that the data sampled during the same time as the training set would work better in the model than the data sampled after.

Model interpretation

Question 7 (9 pts; 3 pts for each part)

For the following model answer TRUE or FALSE for the questions below; no explanation is required and you will only be graded based on the TRUE and FALSE part of your answer even if you provided explanations.

```
Call:
lm(formula = price ~ f.waterfront * sqft_living, data = mod.variables)

Coefficients:
              (Intercept)              f.waterfront1
                 -16206.9                  -267588.0
              sqft_living  f.waterfront1:sqft_living
                    264.4                      348.6
```

(i) **TRUE or FALSE** The model above gives different y-intercepts (for predicting the house price) for houses with and without a view to a waterfront.

True

(ii) **TRUE or FALSE** The model above gives different slopes depending on if the house has a view to a waterfront or not.
True

(iii) **TRUE or FALSE** Using the model shown above to predict the `price` of a 1000 sqft house that has a view to a waterfront we calculate the estimated price this way:

Predicted price $= -16206.9 - 267588 + 264.4 + 348.6 \times 1000$.

False