Attachment

Environment & Data

1. Environment Setting

```
if(!require("pacman")) install.packages("pacman")
pacman::p_load(dplyr, caret, GGally, Hmisc, broom, tidyr, car, e1071, rpart, rpart.plot,
               rattle, randomForest, h2o, forecast)
options(digits = 3)
theme_set(theme_minimal())
set.seed(123)
```

2. Data Preprocessing

a. Data Loading

```
df.0 = read.csv("apartments_for_rent.csv")
```

b. Data Type Encoding

```
df.0[, c(5:21)] = lapply(df.0[, c(5:21)], factor)
df.0[, c(22:23)] = lapply(df.0[, c(22:23)], as.character)
```

c. Missing Data Dealing

```
df.1 = na.omit(df.0)
```

d. Variable Selecting

```
nzv = nearZeroVar(df.1)
df.2 = df.1[, -nzv]
```

e. Outlier Removing

```
df.3 = df.2[, -c(16, 17)]
remove = c("319", "3828", "8775")
df.4 = df.3[row.names(df.3) %nin% remove, ]
```

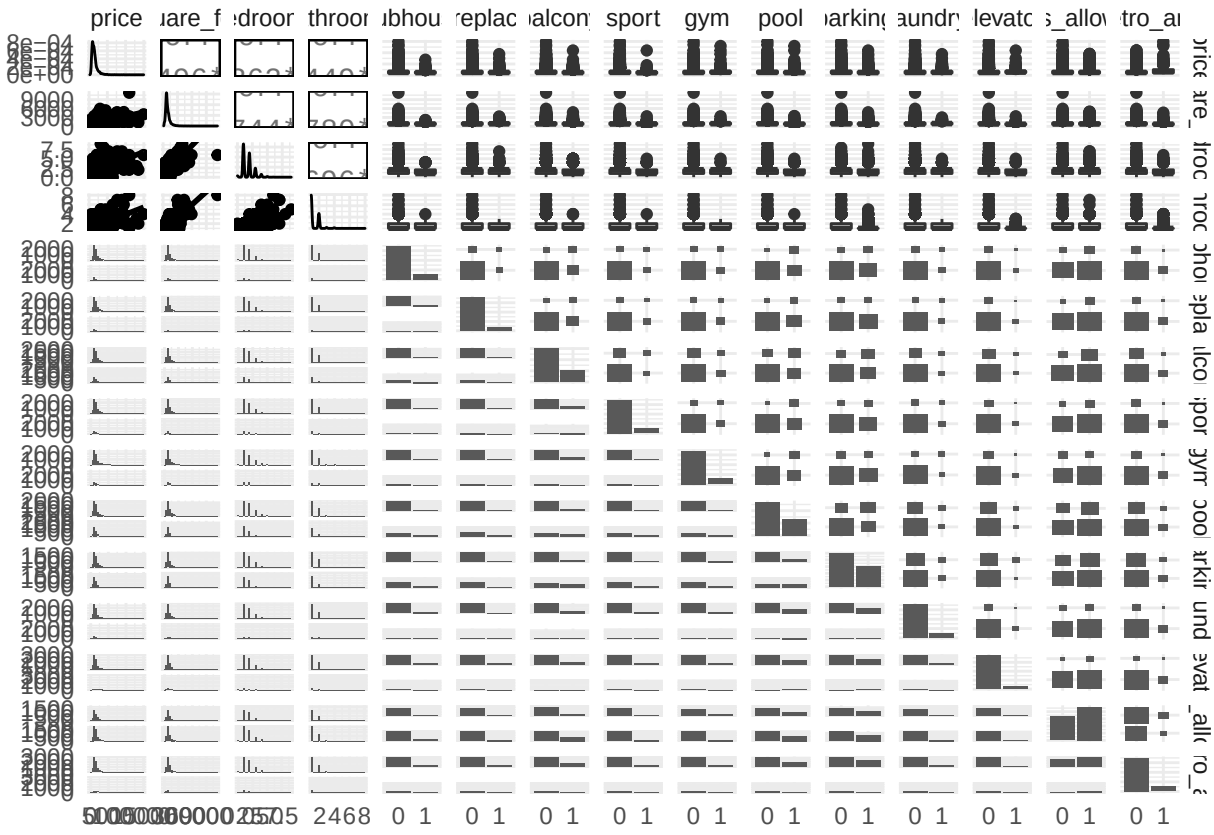
f. Data Partitioning

```
split = createDataPartition(df.4$price, p = 0.8, list = F)
train.set = df.4[split, ]
test.set = df.4[-split, ]
```

The dimension of train.set is 7968 rows and 15 cols. The dimension of test.set is 1990 rows and 15 cols.

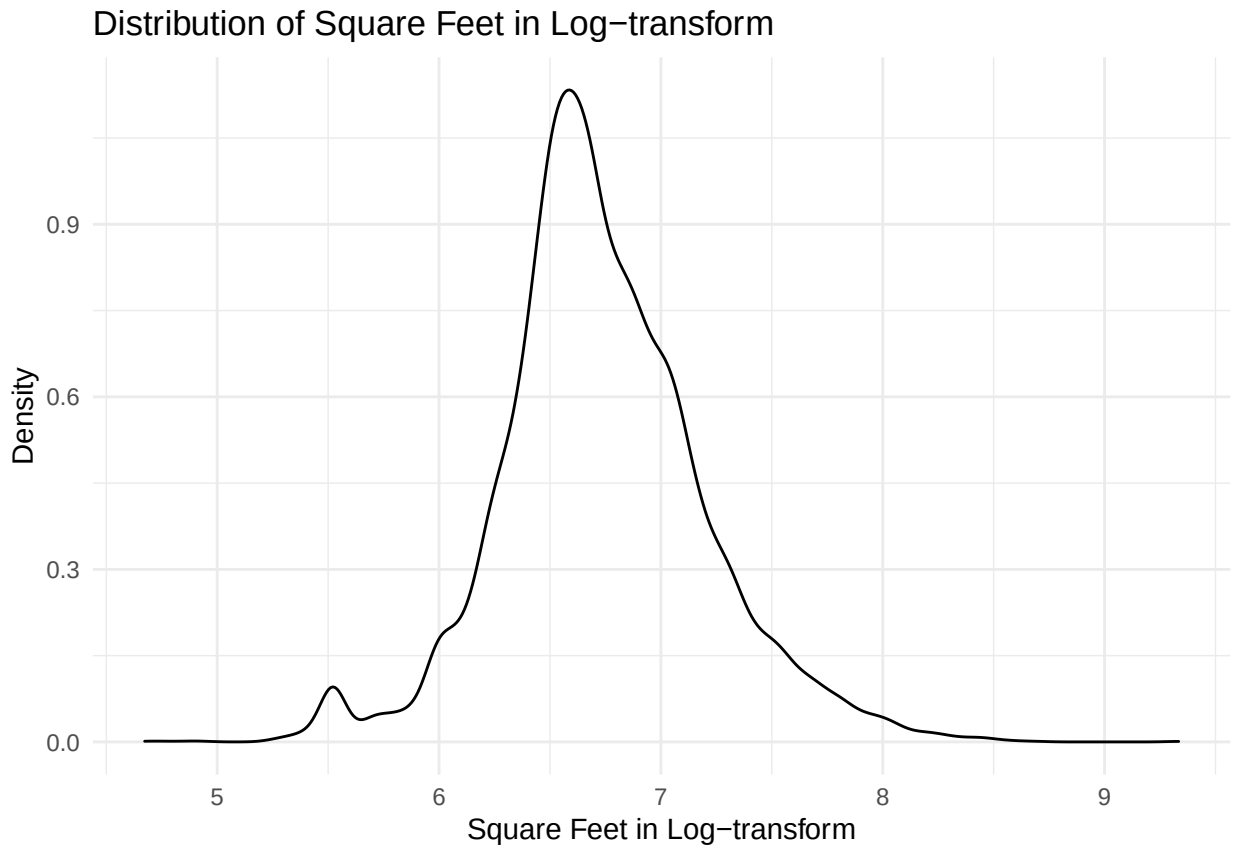
Exploring Data Analysis

```
ggpairs(train.set, lower = list(continuous = wrap("smooth", method = "lm")))
```

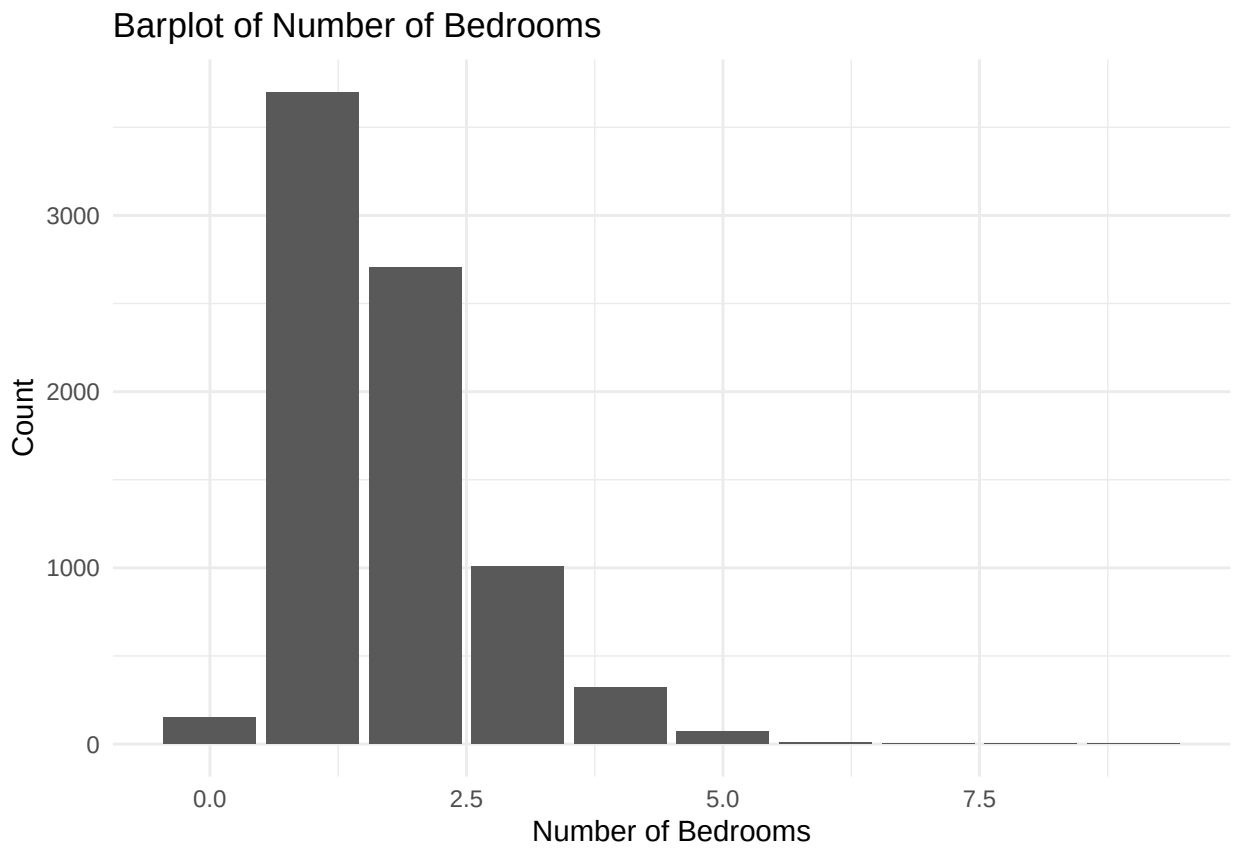


```r
# price in log
qplot(data = train.set, x = log(price), geom = c("density")) +
  labs(title = "Distribution of Price in Log-transform",
       x = "Price in Log-transform", y = "Density")
```

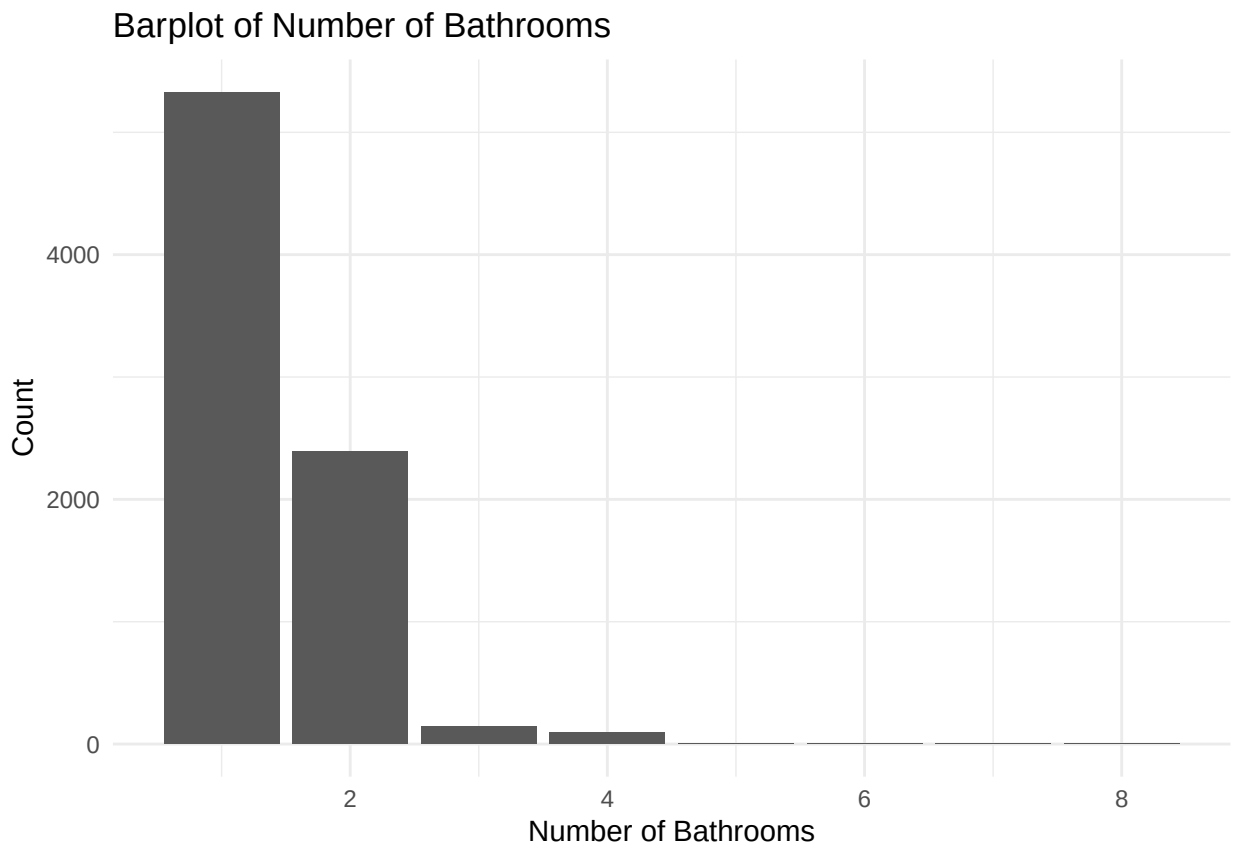
```r
# square_feet in log
qplot(data = train.set, x = log(square_feet), geom = c("density")) +
  labs(title = "Distribution of Square Feet in Log-transform",
       x = "Square Feet in Log-transform", y = "Density")
```



```r
# bedrooms
qplot(data = train.set, x = bedrooms, geom = c("bar")) +
  labs(title = "Barplot of Number of Bedrooms", x = "Number of Bedrooms", y = "Count")
```



```r
# bathrooms
qplot(data = train.set, x = bathrooms, geom = c("bar")) +
  labs(title = "Barplot of Number of Bathrooms", x = "Number of Bathrooms", y = "Count")
```

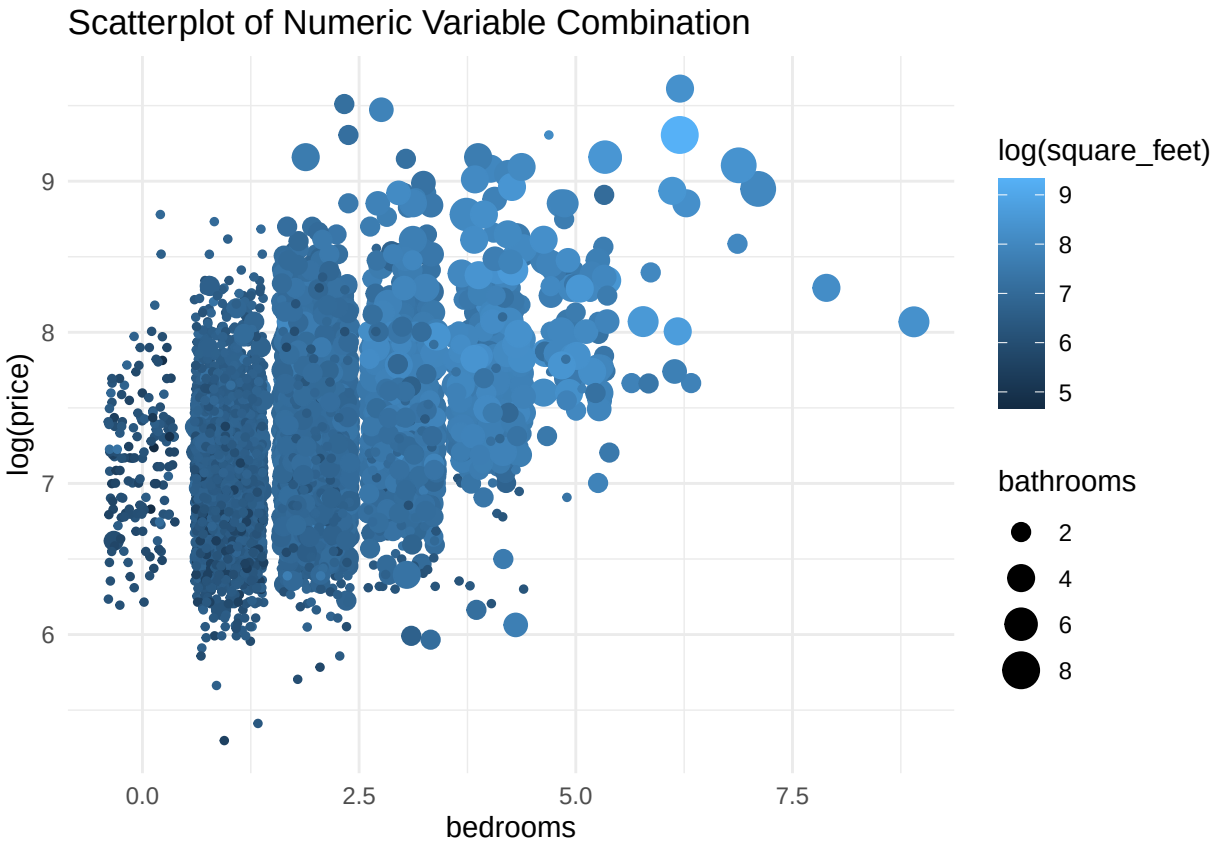


```r
# clubhouse
temp = t.test(price ~ clubhouse, data = train.set)
clubhouse = temp$p.value

# fireplace
temp = t.test(price ~ fireplace, data = train.set)
fireplace = temp$p.value

# balcony
temp = t.test(price ~ balcony, data = train.set)
balcony = temp$p.value

# sport
temp = t.test(price ~ sport, data = train.set)
sport = temp$p.value

# gym
temp = t.test(price ~ gym, data = train.set)
gym = temp$p.value

# pool
temp = t.test(price ~ pool, data = train.set)
pool = temp$p.value

# parking
temp = t.test(price ~ parking, data = train.set)
parking = temp$p.value

# laundry
temp = t.test(price ~ laundry, data = train.set)
laundry = temp$p.value

# elevator
temp = t.test(price ~ elevator, data = train.set)
elevator = temp$p.value

# pets_allowed
temp = t.test(price ~ pets_allowed, data = train.set)
pets_allowed = temp$p.value

# metro_area
temp = t.test(price ~ metro_area, data = train.set)
metro_area = temp$p.value
```

```r
# p-value result
temp = data.frame(clubhouse, fireplace, balcony, sport, gym, pool, parking,
                  laundry, elevator, pets_allowed, metro_area) %>%
  mutate(across(is.numeric, ~ round(., 3))) %>%
  t() %>% data.frame() %>% arrange(desc(.))
colnames(temp) = "p-value"
temp
```

```
##              p-value
## laundry        0.955
## gym            0.231
## elevator       0.002
## pets_allowed   0.002
## clubhouse      0.000
## fireplace      0.000
## balcony        0.000
## sport          0.000
## pool           0.000
## parking        0.000
## metro_area     0.000
```

```
qplot(data = train.set, x = bedrooms, y = log(price), size = bathrooms,
      col = log(square_feet), geom = c("jitter")) +
  labs(title = "Scatterplot of Numeric Variable Combination")
```



Data Analysis

1. Model Fitting

a. Multiple Linear Regression

```
# model fit
# all in
mod.a.1 = lm(price ~ ., data = train.set)
sc.a.1 = glance(mod.a.1)

# backward elimination
mod.all = lm(price ~ ., data = train.set)
mod.a.2 = step(mod.all, direction = "backward", trace = F)
sc.a.2 = glance(mod.a.2)

# foward selection
mod.nul = lm(price ~ 1, data = train.set)
mod.a.3 = step(mod.nul, direction = "forward", trace = F,
               scope = list(lower = mod.nul, upper = mod.all))
sc.a.3 = glance(mod.a.3)

# stepwise combination
mod.a.4 = step(mod.all, direction = "both", trace = F)
sc.a.4 = glance(mod.a.4)

# result compare
temp = data.frame("type" = c("all in", "backward elimination",
                             "foward selection", "stepwise combination"),
                  "adj.r.squared" = c(sc.a.1$adj.r.squared, sc.a.2$adj.r.squared,
                                      sc.a.3$adj.r.squared, sc.a.4$adj.r.squared),
                  "aic" = c(sc.a.1$AIC, sc.a.2$AIC, sc.a.3$AIC, sc.a.4$AIC),
                  "bic" = c(sc.a.1$BIC, sc.a.2$BIC, sc.a.3$BIC, sc.a.4$BIC))
temp
```

```
##                   type adj.r.squared    aic    bic
## 1               all in         0.338 127804 127915
## 2 backward elimination         0.338 127799 127890
## 3     foward selection         0.338 127799 127890
## 4 stepwise combination         0.338 127799 127890
```

```
# final model
mod.a = lm(price ~ square_feet + bedrooms + bathrooms + fireplace + sport + gym +
             parking + elevator + pets_allowed + metro_area, data = train.set)
```

b. Polynomial Non Linear Regression

```
# model fit
# polynomial term
mod.b.1 = lm(price ~ square_feet + I(square_feet^2) + bedrooms + bathrooms +
               fireplace + sport + gym + parking + elevator + pets_allowed + metro_area,
             data = train.set)
sc.b.1 = glance(mod.b.1)

# polynomial & interative term
```

```
mod.b.2 = lm(price ~ square_feet + I(square_feet^2) + bedrooms + bathrooms +
               fireplace + sport + gym + parking + elevator + pets_allowed +
               metro_area + square_feet:bedrooms + square_feet:bathrooms +
               square_feet:pets_allowed + square_feet:metro_area + bedrooms:bathrooms,
             data = train.set)
sc.b.2 = glance(mod.b.2)

# result compare
temp = data.frame("type" = c("polynomial", "polynomial & interative"),
                  "adj.r.squared" = c(sc.b.1$adj.r.squared, sc.b.2$adj.r.squared),
                  "aic" = c(sc.b.1$AIC, sc.b.2$AIC),
                  "bic" = c(sc.b.1$BIC, sc.b.2$BIC))
temp
```

```
##                       type adj.r.squared    aic    bic
## 1               polynomial         0.339 127796 127887
## 2 polynomial & interative         0.363 127504 127630
```

```
# final model
mod.b = mod.b.2
```

c. Support Vector Regression

```
# model fit
mod.c = svm(price ~ ., data = train.set, type = "eps-regression", kernel = "linear")
```

d. Kernel Support Vector Regression

```
# model fit
mod.d = svm(price ~ ., data = train.set, type = "eps-regression", kernel = "radial")
```

e. Decision Tree Regression

```
# model fit
mod.e = train(price ~ ., data = train.set, method = "rpart",
              trControl = trainControl(method = "cv", number = 3, verboseIter = F))
```

f. Random Forest Regression

```
# model fit
set.seed(123)
mod.f = train(price ~ ., data = train.set, method = "rf",
              trControl = trainControl(method = "cv", number = 3, verboseIter = F))
```

g. Artificial Neural Network Regression

```
# server connect
h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         2 hours 42 minutes
```

```
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.30.0.1
##     H2O cluster version age:    7 months and 20 days !!!
##     H2O cluster name:           H2O_started_from_R_andy_iry509
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.94 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.2 (2020-06-22)
```

```
# model fit
mod.g = h2o.deeplearning(y = "price", training_frame = as.h2o(train.set),
                         activation = "Rectifier", hidden = c(6, 6),
                         epochs = 100, train_samples_per_iteration = -2)
```

```
##   |                                                                              |
##   |                                                                              |
```

```
# server disconect
pred.g = h2o.predict(mod.g, newdata = as.h2o(train.set)) %>% as.vector() %>% as.numeric()
```

```
##   |                                                                              |
##   |                                                                              |
```

```
h2o.shutdown()
```

```
## Are you sure you want to shutdown the H2O instance running at http://localhost:54321/ (Y/N)?
```

2. Model Evaluating

```
# a. multiple linear regression
pred.a = predict(mod.a, newdata = test.set)
error = accuracy(pred.a, test.set$price)
rmse.a = format(round(error[2], 2), nsmall = 2) %>% as.numeric()

# b. polynomial non linear regression
pred.b = predict(mod.b, newdata = test.set)
error = accuracy(pred.b, test.set$price)
rmse.b = format(round(error[2], 2), nsmall = 2) %>% as.numeric()

# c. support vector regression
pred.c = predict(mod.c, newdata = test.set)
error = accuracy(pred.c, test.set$price)
rmse.c = format(round(error[2], 2), nsmall = 2) %>% as.numeric()

# d. kernel support vector regression
pred.d = predict(mod.d, newdata = test.set)
error = accuracy(pred.d, test.set$price)
rmse.d = format(round(error[2], 2), nsmall = 2) %>% as.numeric()
```

```r
# e. decision tree regression
pred.e = predict(mod.e, newdata = test.set)
error = accuracy(pred.e, test.set$price)
rmse.e = format(round(error[2], 2), nsmall = 2) %>% as.numeric()

# f. random forest regression
pred.f = predict(mod.f, newdata = test.set)
error = accuracy(pred.f, test.set$price)
rmse.f = format(round(error[2], 2), nsmall = 2) %>% as.numeric()

# g. artificial neural network regression
error = accuracy(pred.g, test.set$price)
rmse.g = format(round(error[2], 2), nsmall = 2) %>% as.numeric()

# rmse result
temp = data.frame(rmse.a, rmse.b, rmse.c, rmse.d, rmse.e, rmse.f, rmse.g) %>%
  t() %>% as.data.frame()
rownames(temp) = c("MLR", "PNLR", "SVR", "KSVR", "DT", "RF", "ANN")
colnames(temp) = "RMSE"
temp[order(temp$RMSE), , drop = F]
```

```
##      RMSE
## PNLR  705
## MLR   705
## RF    706
## KSVR  717
## SVR   742
## DT    784
## ANN   995
```

Model Inference

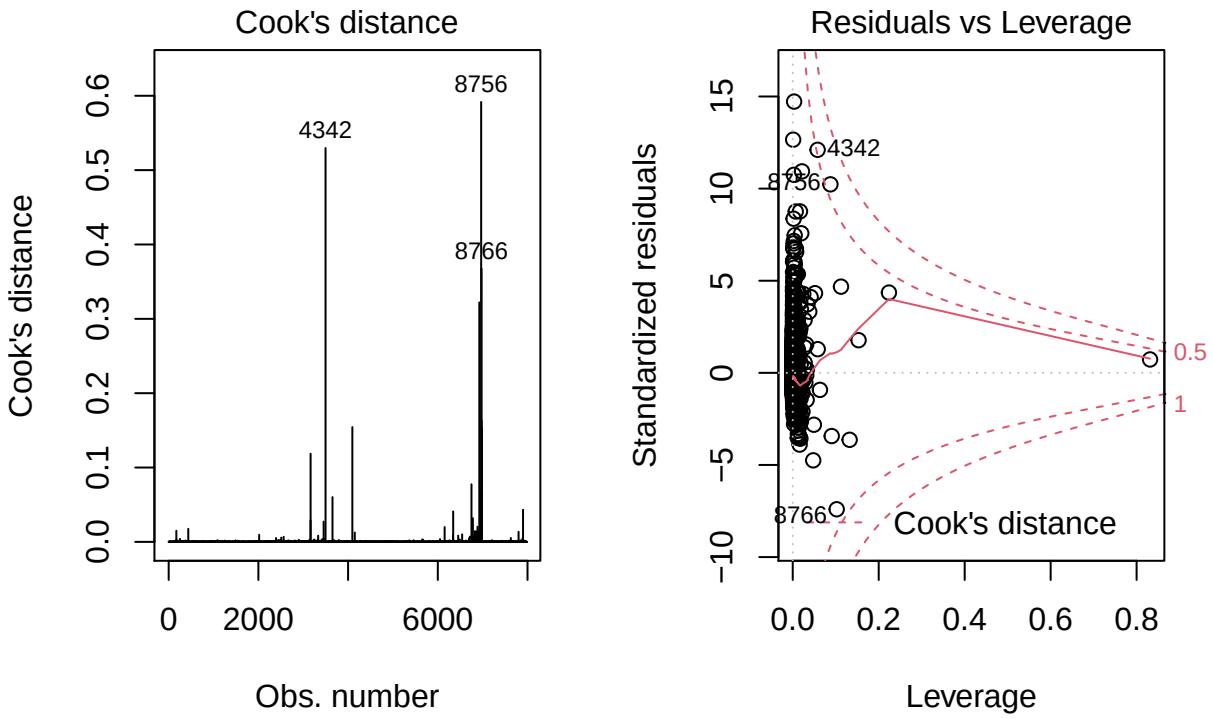
1. Assumption Checking & Remodeling

a. Model Building

```r
# model original
fit.1 = lm(formula = price ~
             square_feet + bedrooms + bathrooms +
             fireplace + sport + gym + parking + elevator + pets_allowed + metro_area +
             I(square_feet^2) +
             square_feet:bedrooms + square_feet:bathrooms +
             square_feet:pets_allowed + square_feet:metro_area + bedrooms:bathrooms,
           data = train.set)
```

b. Assumption Checking

```
# assumption 1 - outlier
par(mfrow = c(1, 2))
plot(fit.1, which = c(4, 5))
```



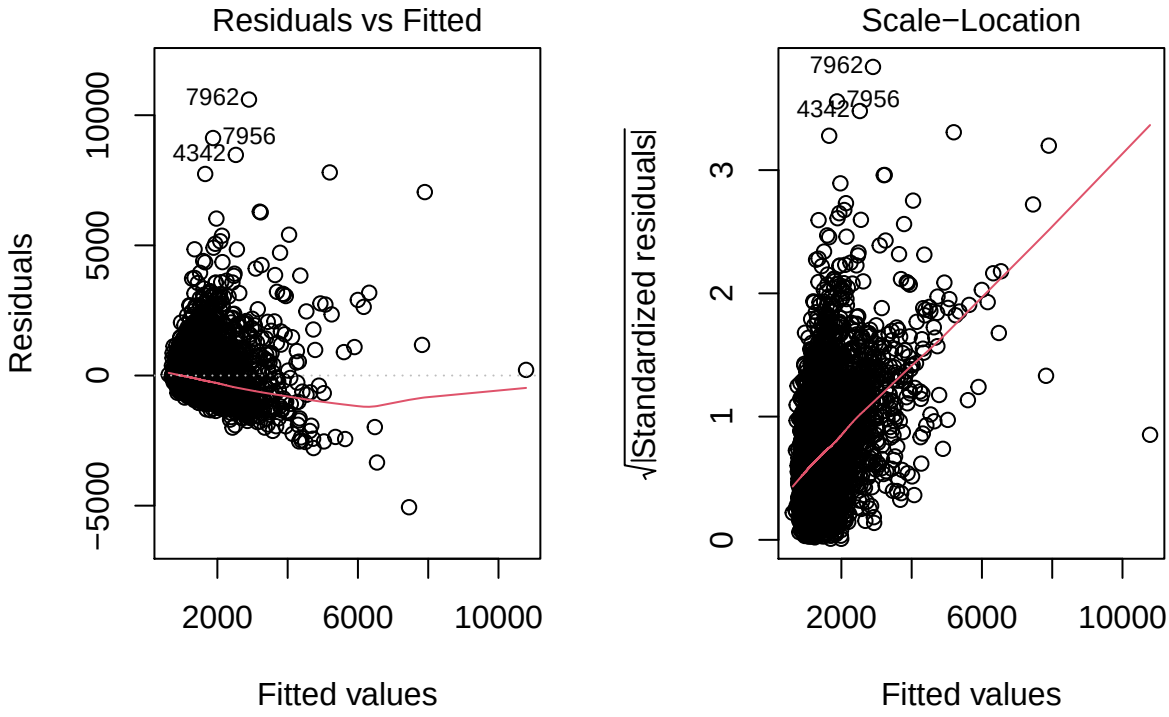

```
# solution
# 1. remove outlier
# 2. transform variable
```

```r
# assumption 2 - collinearity
temp = fit.1 %>% vif() %>% sort(decreasing = T) %>%  data.frame()
colnames(temp) = "VIF"
temp %>%
  mutate("Variable" = row.names(temp)) %>%
  mutate("Collinearity Issue" = ifelse((VIF > 10), "Yes", "No")) %>%
  select(c("Variable", "VIF", "Collinearity Issue"))
```

```
##                    Variable    VIF Collinearity Issue
## 1     square_feet:bathrooms 129.04                Yes
## 2        bedrooms:bathrooms 101.25                Yes
## 3      square_feet:bedrooms  72.19                Yes
## 4          I(square_feet^2)  70.52                Yes
## 5               square_feet  20.04                Yes
## 6                 bathrooms  15.48                Yes
## 7                  bedrooms  10.43                Yes
## 8  square_feet:pets_allowed   5.88                 No
## 9                 metro_area   4.92                 No
## 10   square_feet:metro_area   4.84                 No
## 11             pets_allowed   4.62                 No
## 12                      gym   1.18                 No
## 13                    sport   1.15                 No
## 14                  parking   1.13                 No
## 15                fireplace   1.11                 No
## 16                 elevator   1.08                 No
```

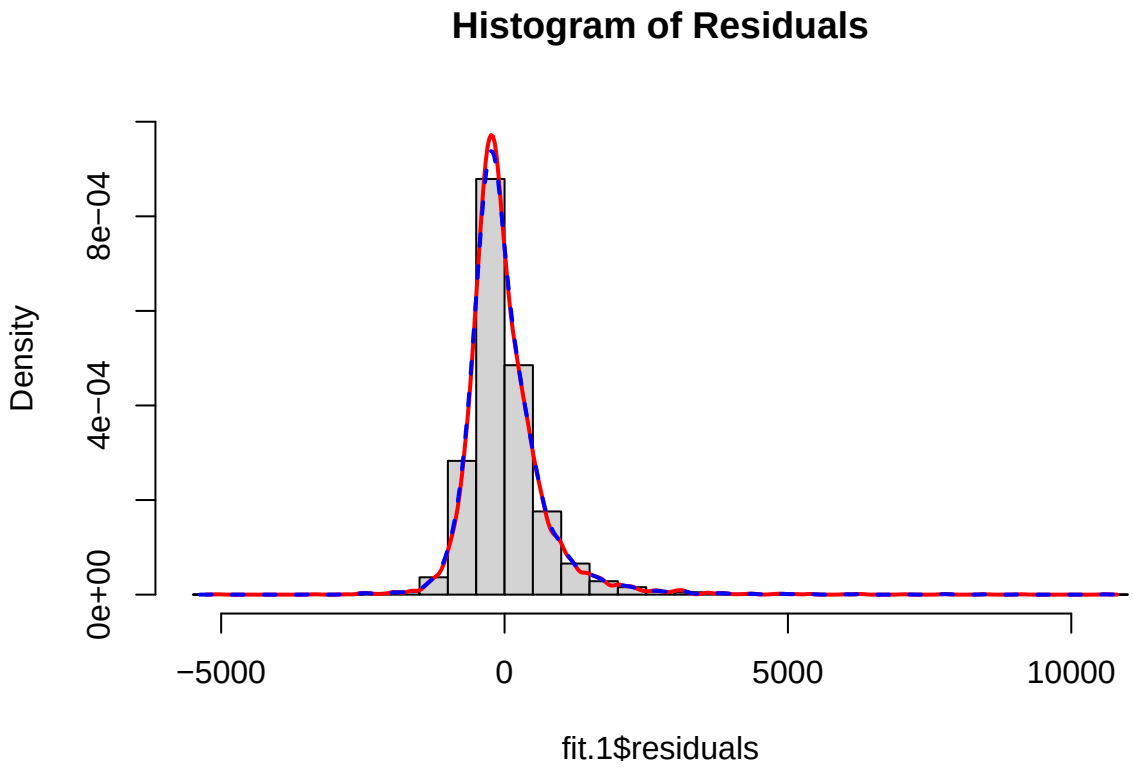
```r
# solution
# 1. combine correlated variable by PCA - no easy to interpretate
# 2. increase data size - impossible now
# 3. remove correlated variable
```

```r
# assumption 3 - heteroscedasticity
par(mfrow = c(1, 2))
plot(fit.1, which = c(1, 3))
```




```r
# solution
# 1. robust SE
# 2. model segmentation
# 3. variable transformation
```

```r
# assumption 4 - normality
par(mfrow = c(1, 1))
hist(fit.1$residuals, probability = T, ylim = c(0, 0.001),
     breaks = 50, main = "Histogram of Residuals")
lines(density(fit.1$residuals), lwd = 2, col = "red") # sample
lines(density(fit.1$residuals, adjust = 1.5), lwd = 2, lty = 2, col = "blue") # normal
```



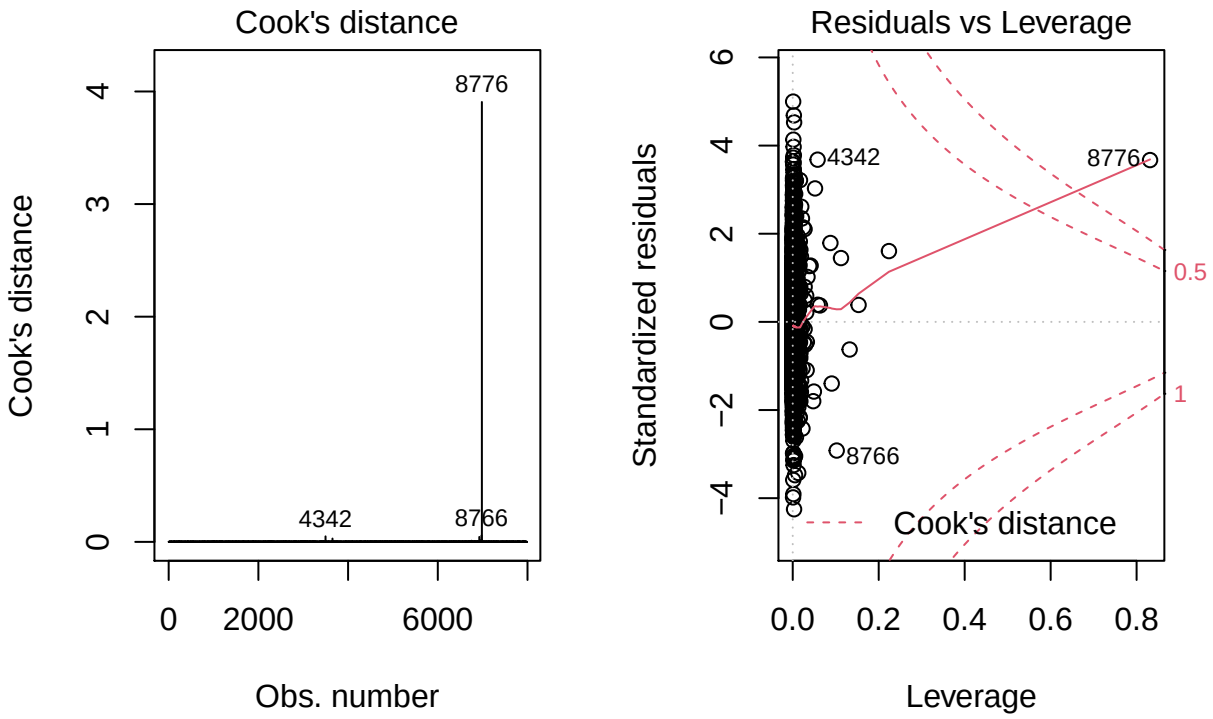

```r
# solution
# 1. remove outlier
# 2. transform variable
```

c. Model Rebuilding

```r
# model transformed
fit.2 = lm(formula = log(price) ~
             square_feet + bedrooms + bathrooms +
             fireplace + sport + gym + parking + elevator + pets_allowed + metro_area +
             I(square_feet^2) +
             square_feet:bedrooms + square_feet:bathrooms +
             square_feet:pets_allowed + square_feet:metro_area + bedrooms:bathrooms,
           data = train.set)
```

d. Assumption Rechecking

```
# assumption 1 - outlier
par(mfrow = c(1, 2))
plot(fit.2, which = c(4, 5))
```



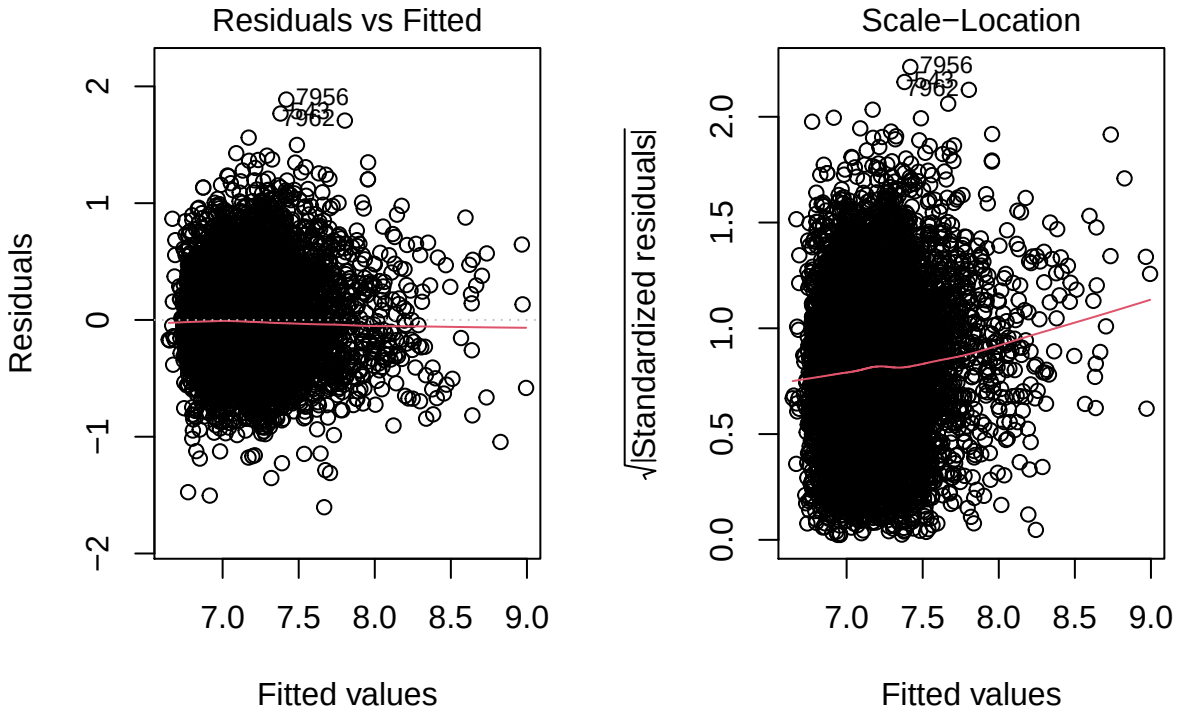

```
# solution
# 1. remove outlier - implement
# 2. transform variable - implement
```

```r
# assumption 2 - collinearity
temp = fit.2 %>% vif() %>% sort(decreasing = T) %>% data.frame()
colnames(temp) = "VIF"
temp %>%
  mutate("Variable" = row.names(temp)) %>%
  mutate("Collinearity Issue" = ifelse((VIF > 10), "Yes", "No")) %>%
  select(c("Variable", "VIF", "Collinearity Issue"))
```

```
##                    Variable    VIF Collinearity Issue
## 1     square_feet:bathrooms 129.04                Yes
## 2        bedrooms:bathrooms 101.25                Yes
## 3      square_feet:bedrooms  72.19                Yes
## 4          I(square_feet^2)  70.52                Yes
## 5               square_feet  20.04                Yes
## 6                 bathrooms  15.48                Yes
## 7                  bedrooms  10.43                Yes
## 8  square_feet:pets_allowed   5.88                 No
## 9                 metro_area   4.92                 No
## 10   square_feet:metro_area   4.84                 No
## 11             pets_allowed   4.62                 No
## 12                      gym   1.18                 No
## 13                    sport   1.15                 No
## 14                  parking   1.13                 No
## 15                fireplace   1.11                 No
## 16                 elevator   1.08                 No
```

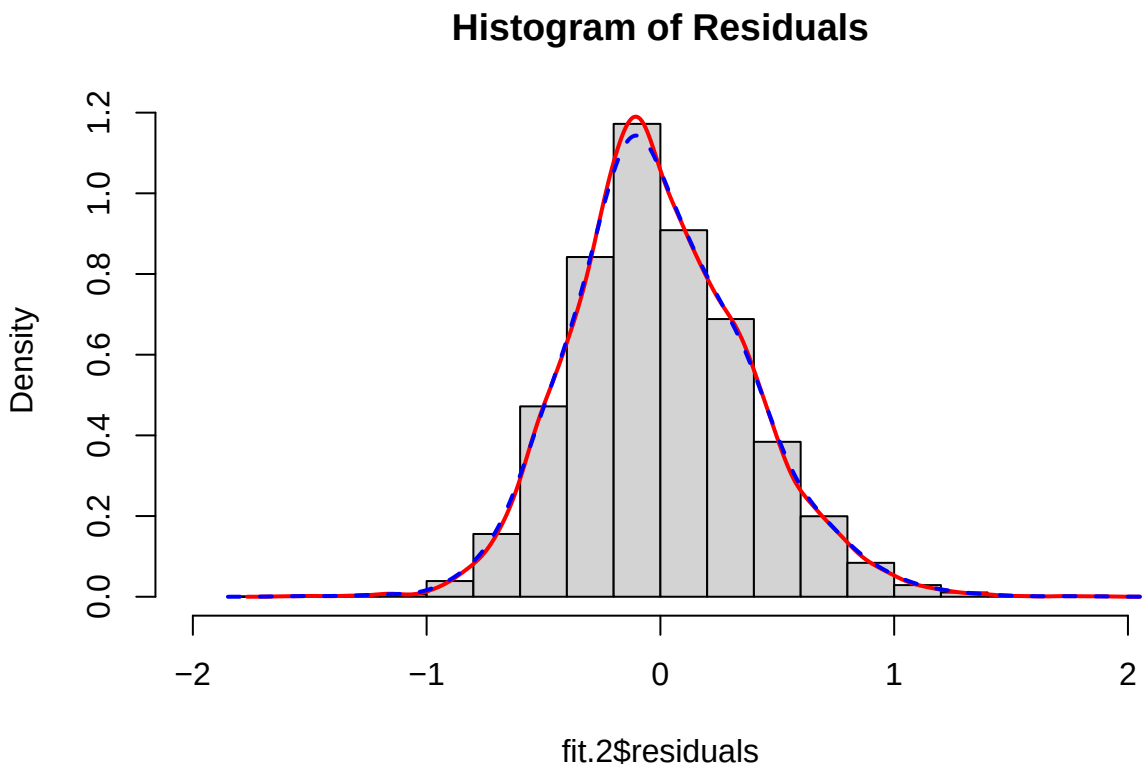
```r
# solution
# 1. combine correlated variable by PCA - no easy to interpretate
# 2. increase data size - impossible now
# 3. remove correlated variable - implement
```

```r
# assumption 3 - heteroscedasticity
par(mfrow = c(1, 2))
plot(fit.2, which = c(1, 3))
```




```r
# solution
# 1. robust SE
# 2. model segmentation
# 3. variable transformation - implement
```

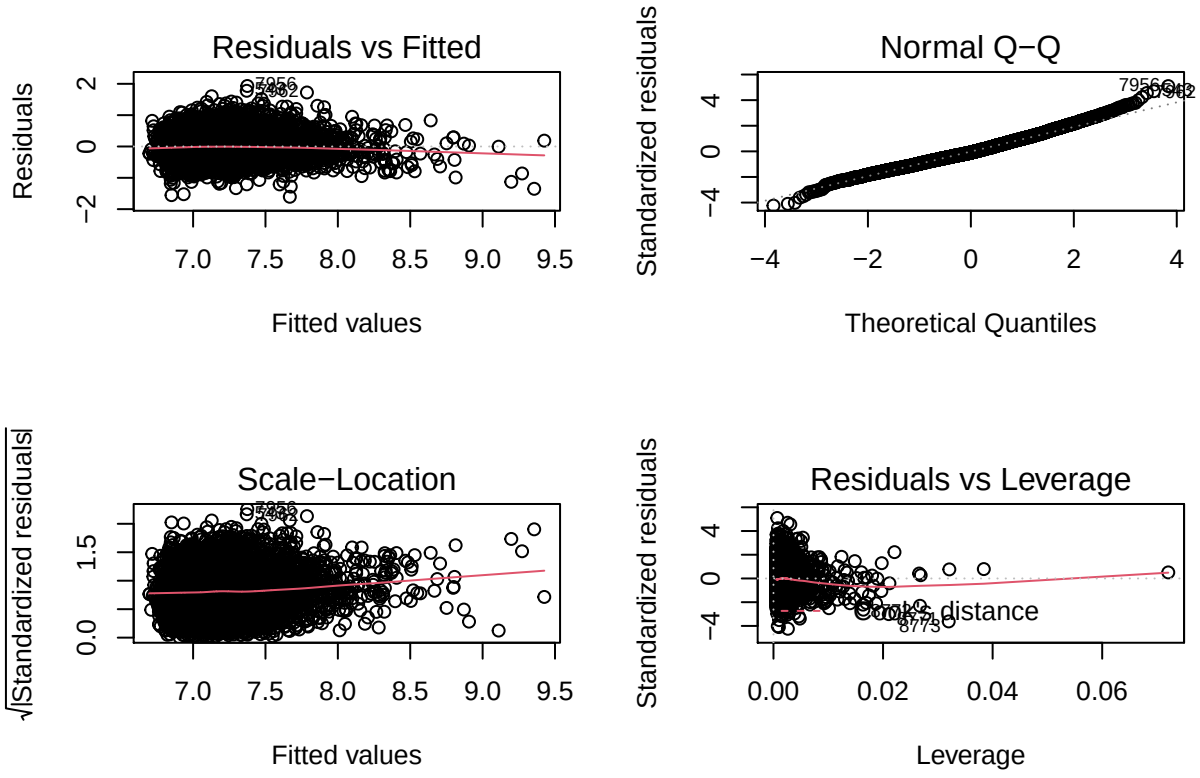
```r
# assumption 4 - normality
par(mfrow = c(1, 1))
hist(fit.2$residuals, probability = T, xlim = c(-2, 2), main = "Histogram of Residuals")
lines(density(fit.2$residuals), lwd = 2, col = "red") # sample
lines(density(fit.2$residuals, adjust = 1.5), lwd = 2, lty = 2, col = "blue") # normal
```




```r
# solution
# 1. remove outlier - implement
# 2. transform variable - implement
```

e. Model Tuning

```
# outlier remove
remove = c("8776", "8766", "4342",
           "4550", "8769", "8701")
train.set = train.set[row.names(train.set) %nin% remove, ]

# model rebuild
fit.3 = lm(formula = log(price) ~
             square_feet + bedrooms + bathrooms +
             fireplace + sport + gym + parking + elevator + pets_allowed + metro_area +
             square_feet:pets_allowed + square_feet:metro_area,
           data = train.set)

# assumption: outlier (fix), heteroscedasticity (fix), normality (fix)
par(mfrow = c(2, 2)); plot(fit.3)
```



```r
# assumption: collinearity (fix)
temp = fit.3 %>% vif() %>% sort(decreasing = T) %>% data.frame()
colnames(temp) = "VIF"
temp %>%
  mutate("Variable" = row.names(temp)) %>%
  mutate("Collinearity Issue" = ifelse((VIF > 10), "Yes", "No")) %>%
  select(c("Variable", "VIF", "Collinearity Issue"))
```

```
##                    Variable  VIF Collinearity Issue
## 1  square_feet:pets_allowed 5.76                 No
## 2                metro_area 5.13                 No
## 3    square_feet:metro_area 5.05                 No
## 4              pets_allowed 4.62                 No
## 5               square_feet 4.28                 No
## 6                 bathrooms 2.77                 No
## 7                  bedrooms 2.51                 No
## 8                       gym 1.17                 No
## 9                     sport 1.15                 No
## 10                  parking 1.13                 No
## 11                fireplace 1.11                 No
## 12                 elevator 1.08                 No
```

2. Causality Establishing

```
temp = summary(fit.3)
coef = temp$coefficients %>%
  data.frame(confint(fit.3, level = 0.95)) %>%
  mutate("Variable" = row.names(temp$coefficients))
colnames(coef) = c("estimate", "SE", "t-value", "p-value", "lcl", "ucl", "Variable")
coef %>%
  mutate(across(is.numeric, ~ round(., 4))) %>%
  select(c("estimate", "lcl", "ucl", "p-value", "Variable"))
```

```
##    estimate     lcl     ucl p-value                 Variable
## 1    6.6388  6.6082  6.6694  0.0000              (Intercept)
## 2    0.0004  0.0003  0.0004  0.0000              square_feet
## 3    0.0021 -0.0120  0.0163  0.7673                 bedrooms
## 4    0.1152  0.0919  0.1385  0.0000                bathrooms
## 5   -0.0368 -0.0653 -0.0083  0.0115               fireplace1
## 6   -0.1351 -0.1623 -0.1080  0.0000                   sport1
## 7    0.1167  0.0915  0.1420  0.0000                     gym1
## 8   -0.0437 -0.0620 -0.0255  0.0000                 parking1
## 9    0.2231  0.1881  0.2581  0.0000                elevator1
## 10   0.0301 -0.0062  0.0664  0.1042            pets_allowed1
## 11   0.2304  0.1742  0.2866  0.0000              metro_area1
## 12  -0.0001 -0.0001  0.0000  0.0001 square_feet:pets_allowed1
## 13   0.0001  0.0001  0.0002  0.0000   square_feet:metro_area1
```

3. Prediction Using

```
# title: 1B1B apartment in Richardson TX $1340/month
# link: https://tinyurl.com/y59wtrzx
# ref: Sublet.com
rentee = data.frame(square_feet = 808, bedrooms = 1, bathrooms = 1,
                    fireplace = factor(0), sport = factor(0), gym = factor(1),
                    parking = factor(1), elevator = factor(0), pets_allowed = factor(1),
                    metro_area = factor(0))
pred.rentee = predict(fit.3, newdata = rentee)
data.frame("Actual" = 1340,
           "Predict" = round(exp(pred.rentee), 0)) %>%
  mutate("Description" = "1B1B apartment in Richardson TX")
```

```
##   Actual Predict                    Description
## 1   1340    1225 1B1B apartment in Richardson TX
```