Week 9 - Homework

STAT 420, Summer 2019, Sushanta Panda

- Exercise 1 (`longley` Macroeconomic Data)
- Exercise 2 (`Credit` Data)
- Exercise 3 (`Sacramento` Housing Data)
- Exercise 4 (Does It Work?)

Exercise 1 (`longley` Macroeconomic Data)

The built-in dataset `longley` contains macroeconomic data for predicting employment. We will attempt to model the `Employed` variable.

```
View(longley)
?longley
```

(a) What is the largest correlation between any pair of predictors in the dataset?

```
cor(longley)
```

```
##              GNP.deflator    GNP Unemployed Armed.Forces Population   Year
## GNP.deflator       1.0000 0.9916     0.6206       0.4647     0.9792 0.9911
## GNP                0.9916 1.0000     0.6043       0.4464     0.9911 0.9953
## Unemployed         0.6206 0.6043     1.0000      -0.1774     0.6866 0.6683
## Armed.Forces       0.4647 0.4464    -0.1774       1.0000     0.3644 0.4172
## Population         0.9792 0.9911     0.6866       0.3644     1.0000 0.9940
## Year               0.9911 0.9953     0.6683       0.4172     0.9940 1.0000
## Employed           0.9709 0.9836     0.5025       0.4573     0.9604 0.9713
##              Employed
## GNP.deflator   0.9709
## GNP            0.9836
## Unemployed     0.5025
## Armed.Forces   0.4573
## Population     0.9604
## Year           0.9713
## Employed       1.0000
```

```
max(cor(longley)[cor(longley) != 1])
```

```
## [1] 0.9953
```

```
which(cor(longley) == max(cor(longley)[cor(longley) != 1]), arr.ind = TRUE)
```

```
##      row col
## Year   6   2
## GNP    2   6
```

The largest correlation between any pair of predictors in the dataset is **0.9953** whcich is between `Year` and `GNP`

(b) Fit a model with `Employed` as the response and the remaining variables as predictors. Calculate and report the variance inflation factor (VIF) for each of the predictors. Which variable has the largest VIF? Do any of the VIFs suggest multicollinearity?

Fit the model eith `Employed` as the response and all remaining variables as predictors

```
fit_lin = lm(Employed ~ . , data = longley)
fit_lin
```

```
##
## Call:
## lm(formula = Employed ~ ., data = longley)
##
## Coefficients:
##  (Intercept)  GNP.deflator           GNP    Unemployed  Armed.Forces
##   -3482.2586        0.0151       -0.0358       -0.0202       -0.0103
##   Population          Year
##      -0.0511        1.8292
```

Derive the Varience Inflation Factor (VIF) for each variable

```
car::vif(fit_lin)
```

```
## GNP.deflator          GNP   Unemployed Armed.Forces   Population         Year
##      135.532     1788.513       33.619        3.589      399.151      758.981
```

Derive the maximim Varience Inflation Factor (VIF) for the variable

```
which.max(car::vif(fit_lin))
```

```
## GNP
##   2
```

The variable GNP **GNP** has the largest Varience Inflation Factor (VIF) and value is: **1788.5135**

From the VIF it seems that the varibale `GNP.deflator` (** > 130**), `GNP` (** > 1500**), `Population` (** > 300**) and `Year` (** > 750**) has the higher VIF , hence** there is multicolinearity exists between the variables**

(c) What proportion of the observed variation in `Population` is explained by a linear relationship with the other predictors?

```
fit_lin_population = lm(Population ~ .-Employed, data = longley)
summary(fit_lin_population)$r.squared
```

```
## [1] 0.9975
```

The Proportion of the observerd variation in `Population` is explained by a linear relationhip with other predictors is: **0.9975**

(d) Calculate the partial correlation coefficient for `Population` and `Employed` **with the effects of the other predictors removed**.

```
fit_lin_Employed = lm(Employed ~ 1, data = longley)
fit_lin_Population = lm(Population ~ 1, data = longley)
cor(resid(fit_lin_Employed), resid(fit_lin_Population))
```

```
## [1] 0.9604
```

The partial correlation coefficient for `population` and `Employed` **with the effects of the other predictors removed** is : **0.9604**

(e) Fit a new model with `Employed` as the response and the predictors from the model in **(b)** that were significant. (Use $\alpha = 0.05$.) Calculate and report the variance inflation factor for each of the predictors. Which variable has the largest VIF? Do any of the VIFs suggest multicollinearity?

Identifying predictor from the model in (b) which are significanct

```
summary(fit_lin)$coefficient[,'Pr(>|t|)'][summary(fit_lin)$coefficient[,'Pr(>|t|)'
] < 0.05]
```

```
##  (Intercept)   Unemployed Armed.Forces         Year
##    0.0035604    0.0025351    0.0009444    0.0030368
```

The predictors which are significance from the model **(b)** are **`Unemployed`** , **`Armed.Forces`** and **`Year`**

Fit a new model with these predictors which are significant

```
fit_model_sign = lm(Employed ~ Unemployed + Armed.Forces + Year, data = longley)
fit_model_sign
```

```
##
## Call:
## lm(formula = Employed ~ Unemployed + Armed.Forces + Year, data = longley)
##
## Coefficients:
##  (Intercept)    Unemployed  Armed.Forces          Year
##    -1.80e+03     -1.47e-02     -7.72e-03      9.56e-01
```

```
car::vif(fit_model_sign)
```

```
##   Unemployed Armed.Forces         Year
##        3.318        2.223        3.891
```

The variation inflation factor of `Unemployed` is: **3.3179**, `Armed.Forces` is: **2.2233**, `Year` is: **3.8909**

```
which.max(car::vif(fit_model_sign))
```

```
## Year
##    3
```

The variable which has highest variation inflation factor(VIF) is: **Year** having VIF is: **3.8909**

(f) Use an F-test to compare the models in parts **(b)** and **(e).** Report the following:

- The null hypothesis

$$H_0 : \beta_{GNP.deflator} = \beta_{GNP} = \beta_{Population} = 0$$

- The test statistic

```
null_model = lm(Employed ~ Unemployed + Armed.Forces + Year, data = longley)
full_model = lm(Employed ~ . , data = longley)
anova(null_model,full_model)
```

```
## Analysis of Variance Table
##
## Model 1: Employed ~ Unemployed + Armed.Forces + Year
## Model 2: Employed ~ GNP.deflator + GNP + Unemployed + Armed.Forces + Popul
ation +
##     Year
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     12 1.323
## 2      9 0.836  3     0.487 1.75   0.23
```

- The distribution of the test statistic under the null hypothesis

The distribution of the test statiscs under the null hypothesis for the following parameter is as below

Distribution of the F value

The ditribution of the **F value** is **F distribution with Degree of Freedom between `3` (df1) and `12` (df2)**

$$F \sim F(p-1, n-p), \text{where n} = 16 \text{ and p} = 4$$

The ditribution of the **P value** is **Uniform distribution**

$$\text{p-value} \sim \text{Unif}(0, 1).$$

The ditribution of the **R2 value** is **Beta distribution** with parameter of shape1 = 4/2, asd shape2 = 11/2

$$R^2 \sim \text{Beta}\left(\frac{p}{2}, \frac{n-p-1}{2}\right), \text{where n} = 16 \text{ and p} = 4$$

- The p-value

```
anova(null_model,full_model)$'Pr(>F)'[2]
```

```
## [1] 0.227
```

The P value is **0.227**

- A decision

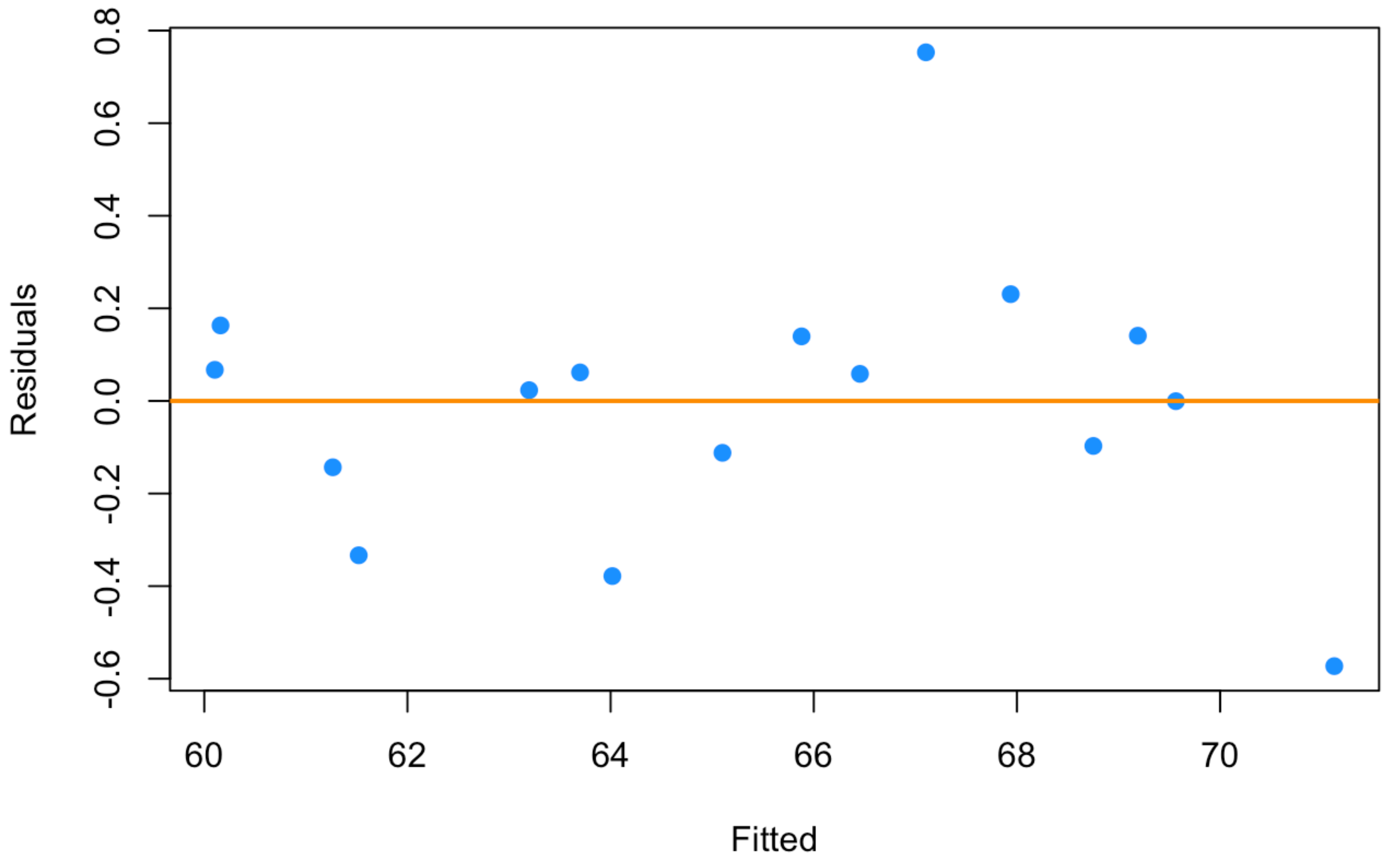
  Since the P value from the anova P test is **0.227** which is greater than for any α (0.1, 0.05), hence we **failed to reject the null hypothesis**, means we prefer the **NULL Model**

- Which model you prefer, **(b)** or **(e)**

  Since we failed to reject the NULL model, we prefer the NULL model, i.e. model comes from the **(e)**

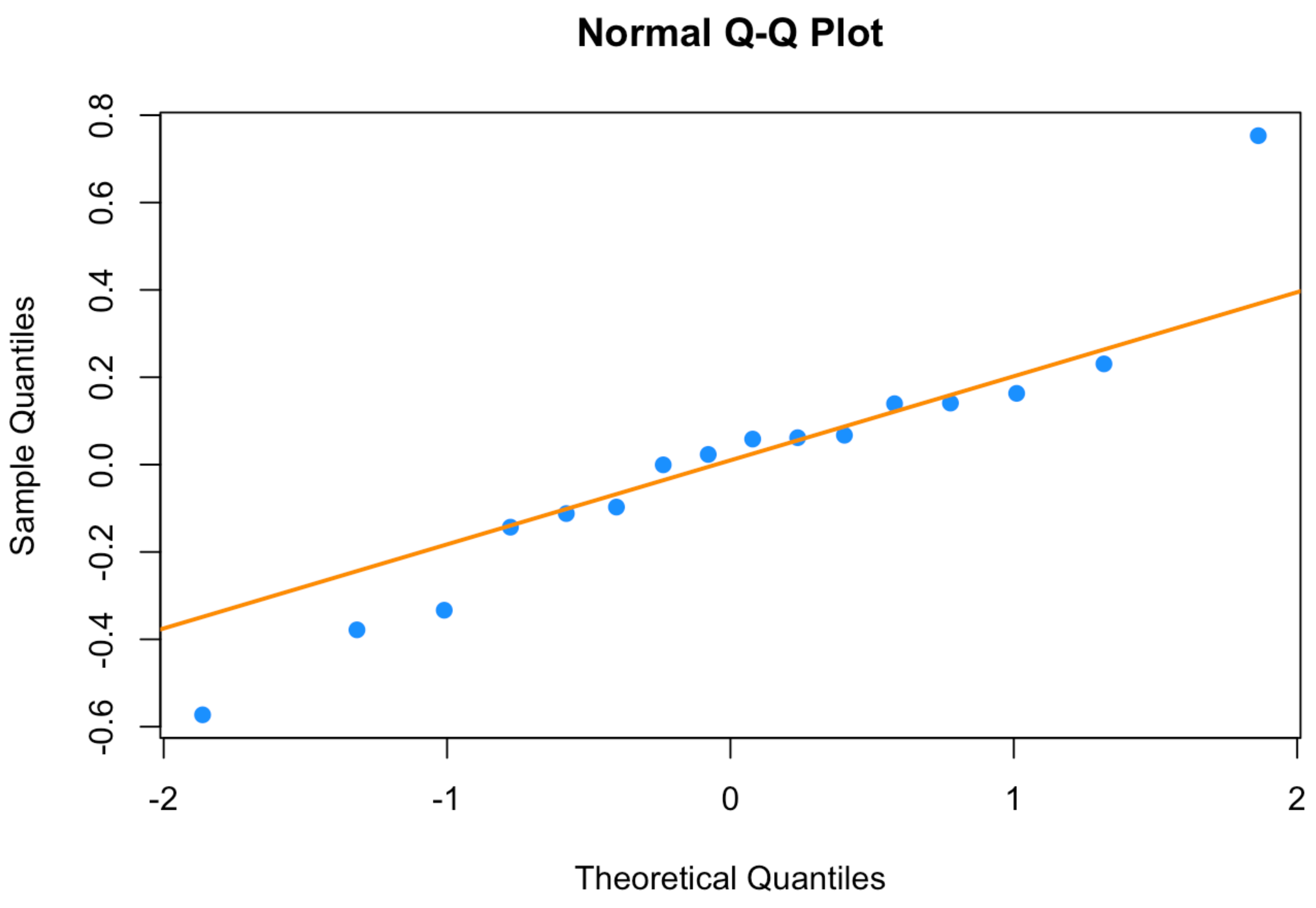
(g) Check the assumptions of the model chosen in part **(f)**. Do any assumptions appear to be violated?

```
plot_fitted_resid(model = null_model)
```



The **constant varience** assumption seems to be **suspect**, as from the above plot, it seem that the residuals are not uniformly ditributed accross the dark orange like. Hence it violates the constant varience assumption

```
plot_qq(model = null_model)
```



The **normality** assumptions seems to be **suspect**, as from the above Q-Q plot, it seems that it has a heavy tail on both higher and lower quantiles. Hence it also violates the normality assumptions

---

Exercise 2 (`Credit` Data)

For this exercise, use the `Credit` data from the `ISLR` package. Use the following code to remove the `ID` variable which is not useful for modeling.

```
library(ISLR)
data(Credit)
Credit = subset(Credit, select = -c(ID))
```

Use `?Credit` to learn about this dataset.

(a) Find a “good” model for `balance` using the available predictors. Use any methods seen in class except transformations of the response. The model should:

- Reach a LOOCV-RMSE below `135`
- Obtain an adjusted R^2 above `0.90`
- Fail to reject the Breusch-Pagan test with an α of 0.01
- Use fewer than 10 β parameters

Store your model in a variable called `mod_a`. Run the two given chunks to verify your model meets the requested criteria. If you cannot find a model that meets all criteria, partial credit will be given for meeting at least some of the criteria.

```
mod_a = lm(Balance ~ log(Income) + Limit + Cards + Age + Gender +
                   Student, data = Credit)
```

```
library(lmtest)

get_bp_decision = function(model, alpha) {
  decide = unname(bptest(model)$p.value < alpha)
  ifelse(decide, "Reject", "Fail to Reject")
}

get_sw_decision = function(model, alpha) {
  decide = unname(shapiro.test(resid(model))$p.value < alpha)
  ifelse(decide, "Reject", "Fail to Reject")
}

get_num_params = function(model) {
  length(coef(model))
}

get_loocv_rmse = function(model) {
  sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
}

get_adj_r2 = function(model) {
  summary(model)$adj.r.squared
}
```

```
get_loocv_rmse(mod_a)
```

```
## [1] 131.2
```

```
get_adj_r2(mod_a)
```

```
## [1] 0.9204
```

```
get_bp_decision(mod_a, alpha = 0.01)
```

```
## [1] "Fail to Reject"
```

```
get_num_params(mod_a)
```

```
## [1] 7
```

```
get_loocv_rmse(mod_a) < 135
```

```
## [1] TRUE
```

```
get_adj_r2(mod_a) > 0.90
```

```
## [1] TRUE
```

```
get_bp_decision(mod_a, alpha = 0.01) == "Fail to Reject"
```

```
## [1] TRUE
```

```
get_num_params(mod_a) < 10
```

```
## [1] TRUE
```

(b) Find another "good" model for `balance` using the available predictors. Use any methods seen in class except transformations of the response. The model should:

- Reach a LOOCV-RMSE below `125`
- Obtain an adjusted R^2 above `0.91`
- Fail to reject the Shapiro-Wilk test with an α of 0.01
- Use fewer than 25 β parameters

Store your model in a variable called `mod_b`. Run the two given chunks to verify your model meets the requested criteria. If you cannot find a model that meets all criteria, partial credit will be given for meeting at least some of the criteria.

```
mod_b = lm(Balance ~ log(Income) + Limit + Cards + Age + Gender + Student +
                           I(log(Income)^2) + I(Limit^2), data = Credit)
```

```r
library(lmtest)

get_bp_decision = function(model, alpha) {
  decide = unname(bptest(model)$p.value < alpha)
  ifelse(decide, "Reject", "Fail to Reject")
}

get_sw_decision = function(model, alpha) {
  decide = unname(shapiro.test(resid(model))$p.value < alpha)
  ifelse(decide, "Reject", "Fail to Reject")
}

get_num_params = function(model) {
  length(coef(model))
}

get_loocv_rmse = function(model) {
  sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
}

get_adj_r2 = function(model) {
  summary(model)$adj.r.squared
}
```

```r
get_loocv_rmse(mod_b)
```

```
## [1] 92.79
```

```r
get_adj_r2(mod_b)
```

```
## [1] 0.9612
```

```r
get_sw_decision(mod_b, alpha = 0.01)
```

```
## [1] "Fail to Reject"
```

```r
get_num_params(mod_b)
```

```
## [1] 9
```

```r
get_loocv_rmse(mod_b) < 125
```

```
## [1] TRUE
```

```r
get_adj_r2(mod_b) > 0.90
```

```
## [1] TRUE
```

```
get_sw_decision(mod_b, alpha = 0.01) == "Fail to Reject"
```

```
## [1] TRUE
```

```
get_num_params(mod_b) < 25
```

```
## [1] TRUE
```

Exercise 3 (`Sacramento` Housing Data)

For this exercise, use the `Sacramento` data from the `caret` package. Use the following code to perform some preprocessing of the data.

```
library(lattice)
library(caret)
```

```
## Loading required package: ggplot2
```

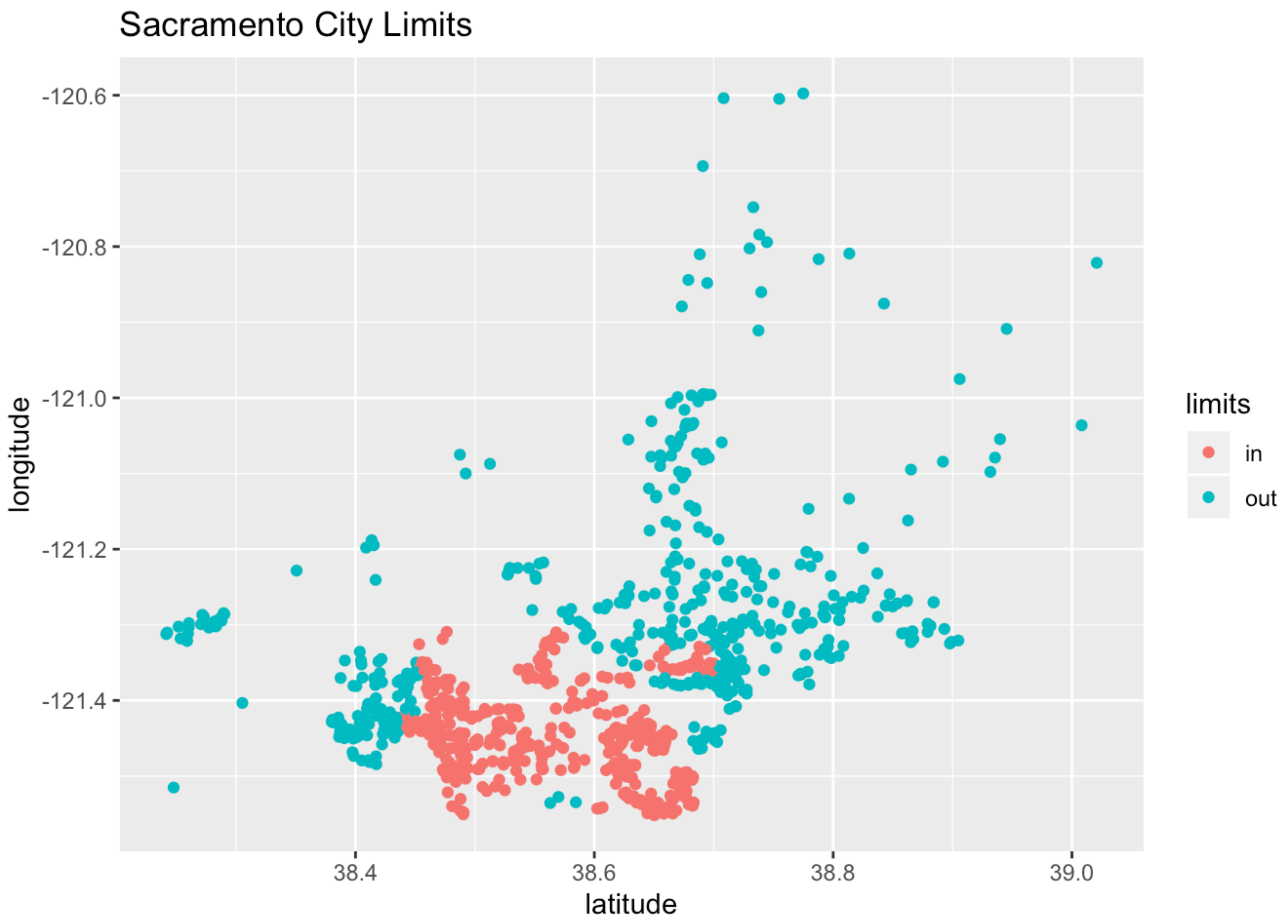
```
library(ggplot2)

data(Sacramento)
sac_data = Sacramento
sac_data$limits = factor(ifelse(sac_data$city == "SACRAMENTO", "in", "out"))
sac_data = subset(sac_data, select = -c(city, zip))
```

Instead of using the `city` or `zip` variables that exist in the dataset, we will simply create a variable (`limits`) indicating whether or not a house is technically within the city limits of Sacramento. (We do this because they would both be factor variables with a **large** number of levels. This is a choice that is made due to laziness, not necessarily because it is justified. Think about what issues these variables might cause.)

Use `?Sacramento` to learn more about this dataset.

A plot of longitude versus latitude gives us a sense of where the city limits are.

```
qplot(y = longitude, x = latitude, data = sac_data,
      col = limits, main = "Sacramento City Limits ")
```



After these modifications, we test-train split the data.

```
set.seed(420)
sac_trn_idx  = sample(nrow(sac_data), size = trunc(0.80 * nrow(sac_data)))
sac_trn_data = sac_data[sac_trn_idx, ]
sac_tst_data = sac_data[-sac_trn_idx, ]
```

The training data should be used for all model fitting. Our goal is to find a model that is useful for predicting home prices.

(a) Find a “good” model for `price`. Use any methods seen in class. The model should reach a LOOCV-RMSE below 77,500 in the training data. Do not use any transformations of the response variable.

```
good_model = lm(price ~ beds + baths + sqft + type + latitude +
    longitude + limits + beds:sqft + beds:longitude + baths:limits +
    sqft:longitude + sqft:limits + type:latitude + latitude:longitude +
    longitude:limits, data = sac_trn_data)
good_model
```

```
##
## Call:
## lm(formula = price ~ beds + baths + sqft + type + latitude +
##     longitude + limits + beds:sqft + beds:longitude + baths:limits +
##     sqft:longitude + sqft:limits + type:latitude + latitude:longitude +
##     longitude:limits, data = sac_trn_data)
##
## Coefficients:
##              (Intercept)                      beds
##                -1.32e+09                  1.33e+07
##                    baths                      sqft
##                -1.61e+04                  7.34e+03
##         typeMulti_Family           typeResidential
##                -1.28e+07                  1.28e+06
##                 latitude                 longitude
##                 3.27e+07                 -1.09e+07
##                limitsout                 beds:sqft
##                 2.23e+07                 -1.50e+01
##           beds:longitude           baths:limitsout
##                 1.10e+05                  2.14e+04
##           sqft:longitude            sqft:limitsout
##                 5.86e+01                 -3.66e+01
## typeMulti_Family:latitude   typeResidential:latitude
##                 3.31e+05                 -3.27e+04
##       latitude:longitude        longitude:limitsout
##                 2.69e+05                  1.83e+05
```

```
sqrt(mean((resid(good_model) / (1 - hatvalues(good_model)))^2))
```

```
## [1] 77287
```

The LOOCV-RMSE is **77287.446**

```
sqrt(mean((resid(good_model) / (1 - hatvalues(good_model)))^2)) < 77500
```

```
## [1] TRUE
```

(b) Is a model that achieves a LOOCV-RMSE below 77,500 useful in this case? That is, is an average error of 77,500 low enough when predicting home prices? To further investigate, use the held-out test data and your model from part **(a)** to do two things:

- Calculate the average percent error:

$$\frac{1}{n}\sum_i \frac{|\text{predicted}_i - \text{actual}_i|}{\text{predicted}_i} \times 100$$

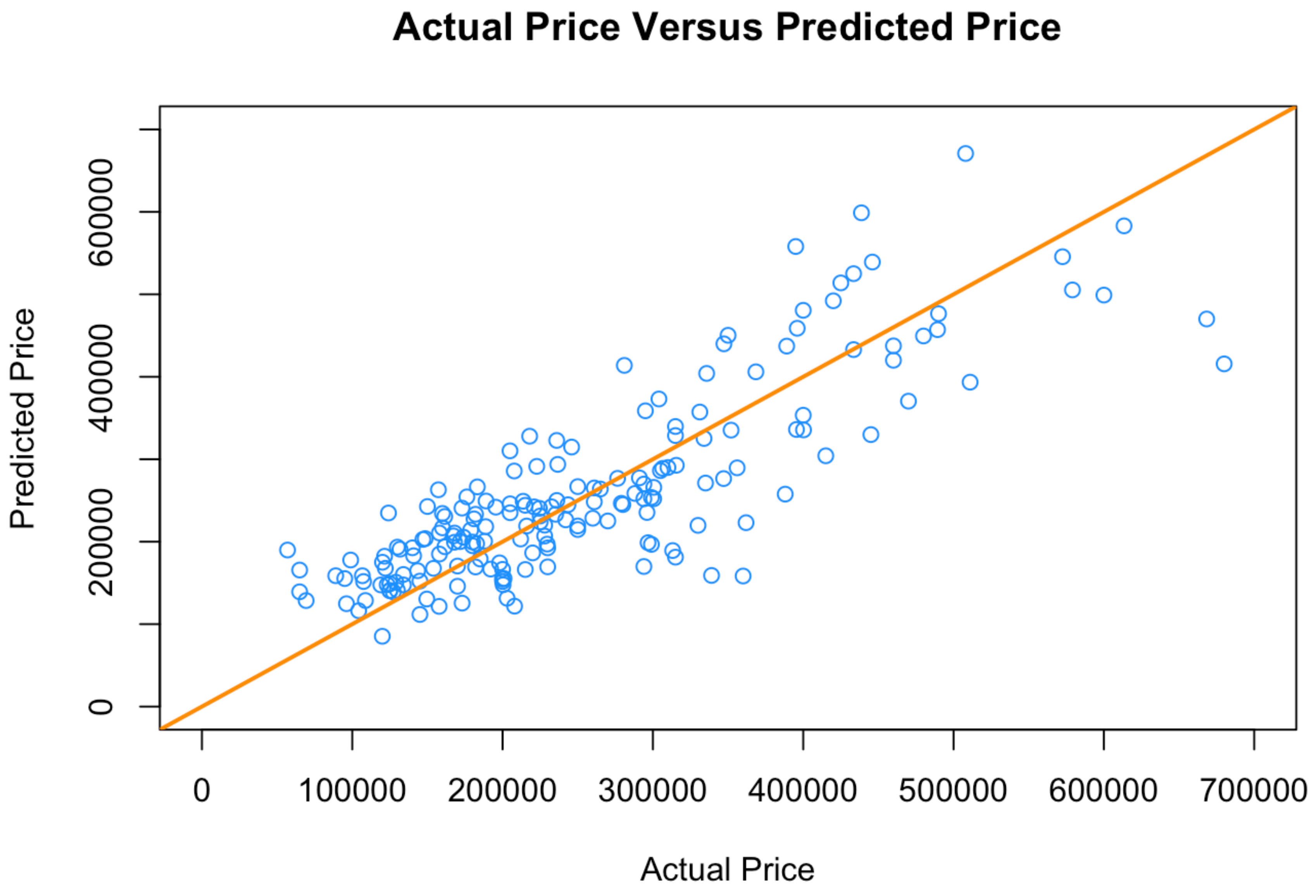
```
predict_price = predict(good_model, newdata = sac_tst_data[,c("beds","baths","sqft
","type","latitude","longitude","limits")])

average_percent_error = (mean(abs(predict_price - sac_tst_data$price) / predict_pr
ice)) * 100
average_percent_error
```

```
## [1] 23.17
```

The average percent error is around **23.1668**

- Plot the predicted versus the actual values and add the line $y = x$.

```
xplot = seq(0,10,by=0.1)
plot(predict_price~sac_tst_data$price, col = "dodgerblue", xlim=c(0,700000), ylim=
c(0,700000),pch = 1, cex = 1, xlab = "Actual Price", ylab = "Predicted Price", mai
n = "Actual Price Versus Predicted Price")
abline(a = 0, b = 1, col = "darkorange", lwd = 2)
```



Based on all of this information, argue whether or not this model is useful.

The model seems to be usefull for lower price value, howver not that great, but is not at all usefull for higher price value. The reason because of the fact that, the Total Percent of error is around 23% and from the above plots it also sees that, the predicted price is not close to the orange line (where the predicted value should match with the actual value) for the lower value. As the price goes high, the difference between the predicted price versus actual price seems wider.

Exercise 4 (Does It Work?)

In this exercise, we will investigate how well backwards AIC and BIC actually perform. For either to be "working" correctly, they should result in a low number of both **false positives** and **false negatives**. In model selection,

- **False Positive**, FP: Incorrectly including a variable in the model. Including a *non-significant* variable
- **False Negative**, FN: Incorrectly excluding a variable in the model. Excluding a *significant* variable

Consider the **true** model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 + \beta_9 x_9 + \beta_{10} x_{10} + \epsilon$$

where $\epsilon \sim N(0, \sigma^2 = 4)$. The true values of the β parameters are given in the `R` code below.

```
beta_0  = 1
beta_1  = -1
beta_2  = 2
beta_3  = -2
beta_4  = 1
beta_5  = 1
beta_6  = 0
beta_7  = 0
beta_8  = 0
beta_9  = 0
beta_10 = 0
sigma = 2
```

Then, as we have specified them, some variables are significant, and some are not. We store their names in `R` variables for use later.

```
not_sig  = c("x_6", "x_7", "x_8", "x_9", "x_10")
signif = c("x_1", "x_2", "x_3", "x_4", "x_5")
```

We now simulate values for these `x` variables, which we will use throughout part **(a)**.

```
set.seed(420)
n = 100
x_1  = runif(n, 0, 10)
x_2  = runif(n, 0, 10)
x_3  = runif(n, 0, 10)
x_4  = runif(n, 0, 10)
x_5  = runif(n, 0, 10)
x_6  = runif(n, 0, 10)
x_7  = runif(n, 0, 10)
x_8  = runif(n, 0, 10)
x_9  = runif(n, 0, 10)
x_10 = runif(n, 0, 10)
```

We then combine these into a data frame and simulate `y` according to the true model.

```
sim_data_1 = data.frame(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_10,
  y = beta_0 + beta_1 * x_1 + beta_2 * x_2 + beta_3 * x_3 + beta_4 * x_4 +
      beta_5 * x_5 + rnorm(n, 0 , sigma)
)
```

We do a quick check to make sure everything looks correct.

```
head(sim_data_1)
```

```
##     x_1   x_2    x_3    x_4    x_5   x_6    x_7    x_8   x_9   x_10       y
## 1 6.055 4.088 8.7894 1.8180 0.8198 8.146 9.7305 9.6673 6.915 4.5523 -11.627
## 2 9.703 3.634 5.0768 5.5784 6.3193 6.033 3.2301 2.6707 2.214 0.4861  -0.147
## 3 1.745 3.899 0.5431 4.5068 1.0834 3.427 3.2223 5.2746 8.242 7.2310  15.145
## 4 4.758 5.315 7.6257 0.1287 9.4057 6.168 0.2472 6.5325 2.102 4.5814   2.404
## 5 7.245 7.225 9.5763 3.0398 0.4194 5.937 9.2169 4.6228 2.527 9.2349  -7.910
## 6 8.761 5.177 1.7983 0.5949 9.2944 9.392 1.0017 0.4476 5.508 5.9687   9.764
```

Now, we fit an incorrect model.

```
fit = lm(y ~ x_1 + x_2 + x_6 + x_7, data = sim_data_1)
coef(fit)
```

```
## (Intercept)         x_1         x_2         x_6         x_7
##     -1.3758     -0.3572      2.1040      0.1344     -0.3367
```

Notice, we have coefficients for `x_1`, `x_2`, `x_6`, and `x_7`. This means that `x_6` and `x_7` are false positives, while `x_3`, `x_4`, and `x_5` are false negatives.

To detect the false negatives, use:

```
# which are false negatives?
!(signif %in% names(coef(fit)))
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE
```

To detect the false positives, use:

```
# which are false positives?
names(coef(fit)) %in% not_sig
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE
```

Note that in both cases, you could `sum()` the result to obtain the number of false negatives or positives.

(a) Set a seed equal to your birthday; then, using the given data for each `x` variable above in `sim_data_1`, simulate the response variable `y` 300 times. Each time,

- Fit an additive model using each of the `x` variables.

- Perform variable selection using backwards AIC.
- Perform variable selection using backwards BIC.
- Calculate and store the number of false negatives for the models chosen by AIC and BIC.
- Calculate and store the number of false positives for the models chosen by AIC and BIC.

Calculate the rate of false positives and negatives for both AIC and BIC. Compare the rates between the two methods. Arrange your results in a well formatted table.

Run the Simulations, fitting the model, calculate the AIC and BIC

```r
library(knitr)
birthday = 19770411
set.seed(birthday)
num_sim = 300
mod1_false_negative_aic = 0
mod1_false_postive_aic = 0
mod1_false_negative_bic = 0
mod1_false_postive_bic = 0

for(i in 1:num_sim){
  sim_data_1 = data.frame(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_10,
  y = beta_0 + beta_1 * x_1 + beta_2 * x_2 + beta_3 * x_3 + beta_4 * x_4 +
      beta_5 * x_5 + rnorm(n, 0 , sigma)
  )

  mod1_fit_add = lm(y ~ ., data = sim_data_1)

  mod1_fit_add_aic_back = step(mod1_fit_add, direction = "backward",trace=0)
  mod1_false_negative_aic = mod1_false_negative_aic + sum(!(signif %in% names(coef
(mod1_fit_add_aic_back)))) # which are false negatives?
  mod1_false_postive_aic = mod1_false_postive_aic + sum(names(coef(mod1_fit_add_ai
c_back)) %in% not_sig) # which are false positives?

  mod1_fit_add_bic_back = step(mod1_fit_add, direction = "backward", k = log(n),tr
ace=0)
  mod1_false_negative_bic = mod1_false_negative_bic + sum(!(signif %in% names(coef
(mod1_fit_add_bic_back)))) # which are false negatives?
  mod1_false_postive_bic = mod1_false_postive_bic + sum(names(coef(mod1_fit_add_bi
c_back)) %in% not_sig) # which are false positives?
}
```

Derive the Rate of the AIC and BIC for False Negetive, False Positive

```
mod1_rate_false_negative_aic = mod1_false_negative_aic / num_sim
mod1_rate_false_postive_aic = mod1_false_postive_aic / num_sim

mod1_rate_false_negative_bic = mod1_false_negative_bic / num_sim
mod1_rate_false_postive_bic = mod1_false_postive_bic / num_sim

mod1_results = data.frame(
  FN = c(mod1_rate_false_negative_aic,mod1_rate_false_negative_bic),
  FP = c(mod1_rate_false_postive_aic,mod1_rate_false_postive_bic)
)
rownames(mod1_results) = c("AIC", "BIC")

mod1_results
```

```
##     FN     FP
## AIC  0 0.9433
## BIC  0 0.1833
```

Results of the AIC, BIC in the form of a table for Model 1

```
kable(mod1_results, format = "pandoc",padding = 2,caption = "Model 1 - Compare AIC
BIC against False Positive (FP), False negative(FN)")
```

Model 1 - Compare AIC BIC against False Positive (FP), False negative(FN)

	FN	FP
AIC	0	0.9433
BIC	0	0.1833

(b) Set a seed equal to your birthday; then, using the given data for each `x` variable below in `sim_data_2`, simulate the response variable `y` 300 times. Each time,

- Fit an additive model using each of the `x` variables.
- Perform variable selection using backwards AIC.
- Perform variable selection using backwards BIC.
- Calculate and store the number of false negatives for the models chosen by AIC and BIC.
- Calculate and store the number of false positives for the models chosen by AIC and BIC.

Calculate the rate of false positives and negatives for both AIC and BIC. Compare the rates between the two methods. Arrange your results in a well formatted table. Also compare to your answers in part **(a)** and suggest a reason for any differences.

```
set.seed(420)
x_1  = runif(n, 0, 10)
x_2  = runif(n, 0, 10)
x_3  = runif(n, 0, 10)
x_4  = runif(n, 0, 10)
x_5  = runif(n, 0, 10)
x_6  = runif(n, 0, 10)
x_7  = runif(n, 0, 10)
x_8  = x_1 + rnorm(n, 0, 0.1)
x_9  = x_1 + rnorm(n, 0, 0.1)
x_10 = x_2 + rnorm(n, 0, 0.1)

sim_data_2 = data.frame(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_10,
  y = beta_0 + beta_1 * x_1 + beta_2 * x_2 + beta_3 * x_3 + beta_4 * x_4 +
      beta_5 * x_5 + rnorm(n, 0 , sigma)
)
```

Run the Simulations, fitting the model, calculate the AIC and BIC

```
library(knitr)
birthday = 19770411
set.seed(birthday)
num_sim = 300

mod2_false_negative_aic = 0
mod2_false_postive_aic = 0
mod2_false_negative_bic = 0
mod2_false_postive_bic = 0



for(i in 1:num_sim){
  sim_data_2 = data.frame(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_10,
  y = beta_0 + beta_1 * x_1 + beta_2 * x_2 + beta_3 * x_3 + beta_4 * x_4 +
      beta_5 * x_5 + rnorm(n, 0 , sigma)
  )

  mod2_fit_add = lm(y ~ ., data = sim_data_2)

  mod2_fit_add_aic_back = step(mod2_fit_add, direction = "backward",trace=0)
  mod2_false_negative_aic = mod2_false_negative_aic + sum(!(signif %in% names(coef
(mod2_fit_add_aic_back)))) # which are false negatives?
  mod2_false_postive_aic = mod2_false_postive_aic + sum(names(coef(mod2_fit_add_ai
c_back)) %in% not_sig) # which are false positives?

  mod2_fit_add_bic_back = step(mod2_fit_add, direction = "backward", k = log(n),tr
ace=0)
  mod2_false_negative_bic = mod2_false_negative_bic + sum(!(signif %in% names(coef
(mod2_fit_add_bic_back)))) # which are false negatives?
  mod2_false_postive_bic = mod2_false_postive_bic + sum(names(coef(mod2_fit_add_bi
c_back)) %in% not_sig) # which are false positives?
}
```

Derive the Rate of the AIC and BIC for False Negetive, False Positive

```
mod2_rate_false_negative_aic = mod2_false_negative_aic / num_sim
mod2_rate_false_postive_aic = mod2_false_postive_aic / num_sim

mod2_rate_false_negative_bic = mod2_false_negative_bic / num_sim
mod2_rate_false_postive_bic = mod2_false_postive_bic / num_sim

mod2_results = data.frame(
  FN = c(mod2_rate_false_negative_aic,mod2_rate_false_negative_bic),
  FP = c(mod2_rate_false_postive_aic,mod2_rate_false_postive_bic)
)
rownames(mod2_results) = c("AIC", "BIC")
mod2_results
```

```
##         FN   FP
## AIC 0.7967 1.58
## BIC 0.8667 1.02
```

Results of the AIC, BIC in the form of a table for Model 2

```
kable(mod2_results, format = "pandoc",padding = 2,caption = "Model 2 - Compare AIC
BIC against False Positive (FP), False negative(FN)")
```

Model 2 - Compare AIC BIC against False Positive (FP), False negative(FN)

	FN	FP
AIC	0.7967	1.58
BIC	0.8667	1.02

Creating data frame of the output of the Model 1 and Model 2

```
final_Results = data.frame(
  Model1_FN = c(mod1_rate_false_negative_aic,mod1_rate_false_negative_bic),
  Model2_FN = c(mod2_rate_false_negative_aic,mod2_rate_false_negative_bic),
  Model1_FP = c(mod1_rate_false_postive_aic,mod1_rate_false_postive_bic),
  Model2_FP = c(mod2_rate_false_postive_aic,mod2_rate_false_postive_bic)
)
rownames(final_Results) = c("AIC","BIC")
final_Results
```

```
##     Model1_FN Model2_FN Model1_FP Model2_FP
## AIC         0    0.7967    0.9433      1.58
## BIC         0    0.8667    0.1833      1.02
```

Results of the AIC, BIC in the form of a table for Model 1 & Model 2

```
kable(final_Results, format = "pandoc",padding = 2,caption = "Model 1 & Model 2 Co
mpare AIC BIC against False Positive (FP), False negative(FN)")
```

Model 1 & Model 2 Compare AIC BIC against False Positive (FP), False negative(FN)

	Model1_FN	Model2_FN	Model1_FP	Model2_FP
AIC	0	0.7967	0.9433	1.58
BIC	0	0.8667	0.1833	1.02

In the `sim_data_2` , the variable x_8, x_9 and x_10 has `exact collinearity` with the x_1, x_2 and x_2 respectively, where as this collinearity doesn't exists in `sim_data_1` . Hence False Negetive (FN) in the model 1 (`sim_data_1`) is zero(0), means none of the significance parameters have moved out from the model (via AIC or BIC). On the contrary, it happened in the Model 2 (`sim_data_2`) because AIC/BIC picks x_8 / x_9 / x_10 instead of x_1 or x_2 in the model because of collinearity.

Since the Model 2 (`sim_data_2`) picks sometime the x_8 / x_9 or x_10 instead of x_1 / x_2 (because of collinearity), the False positive have increased in Moddel 2 (`sim_data_2`) as compared to the Model 1 (`sim_data_1`).