Week 9 - Homework

STAT 420, Summer 2023, Adriane Yi

2023-07-17

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? **Solution:**

```
cor_matrix = cor(longley)
cor_matrix
```

```
##              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
```

```
diag(cor_matrix) = 0
largest_cor = max(abs(cor_matrix), na.rm = TRUE)
largest_cor
```

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

Solution: 0.9953 which is between `GNP` and `Year`.

(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?

```
library(faraway)
longley_mod = lm(Employed ~ ., data = longley)
vif(longley_mod)
```

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

```
idx = which.max(vif(longley_mod))
vif(longley_mod)[idx]
```

```
##  GNP
## 1789
```

Solution: Variable `GNP` has the largest VIF (1788.513) Except for Armed.Forces which is less than 5, all other predictor variables are very large and suggest multicollinearity.

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

```
pop_model = lm(Population ~ ., data = longley)
(pop_r_2 = summary(pop_model)$r.squared)
```

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

Solution: 0.9975.

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

```
pop_mod = lm(Population ~ GNP.deflator + GNP + Unemployed + Armed.Forces + Year,
             data = longley)
emp_mod = lm(Employed ~ GNP.deflator + GNP + Unemployed + Armed.Forces + Year,
             data = longley)
cor(resid(pop_mod), resid(emp_mod))
```

```
## [1] -0.07514
```

Solution: -0.0751

(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?

```
alpha = 0.05
which(summary(longley_mod)$coef[, "Pr(>|t|)"] < alpha)
```

```
##  (Intercept)   Unemployed Armed.Forces         Year
##            1            4            5            7
```

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

```r
vif(longley_mod_sig)
```

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

```r
max(vif(longley_mod_sig))
```

```
## [1] 3.891
```

Solution:
`Year` has the largest VIF. However, VIF of all three variables are less than 5. This suggests that there is no multicollinearity issue.

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

```r
anova_test = anova(longley_mod, longley_mod_sig)
anova_test
```

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

- The null hypothesis :
  Null hypothesis is that β*(GNP.deflator)* $= \beta$(GNP) $= \beta$_(Population) $= 0$

- The test statistic :
  F test statistic is 1.7465

- The distribution of the test statistic under the null hypothesis: F distribution with p-q and n-p degrees of freedom

```r
n = length(resid(longley_mod))
p = length(coef(longley_mod))
q = length(coef(longley_mod_sig))
p - q
```

```
## [1] 3
```

```r
n - p
```

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

Solution: F_3, 9 degrees of freedom

- The p-value : 0.227

```
anova_test[2,"Pr(>F)"]
```

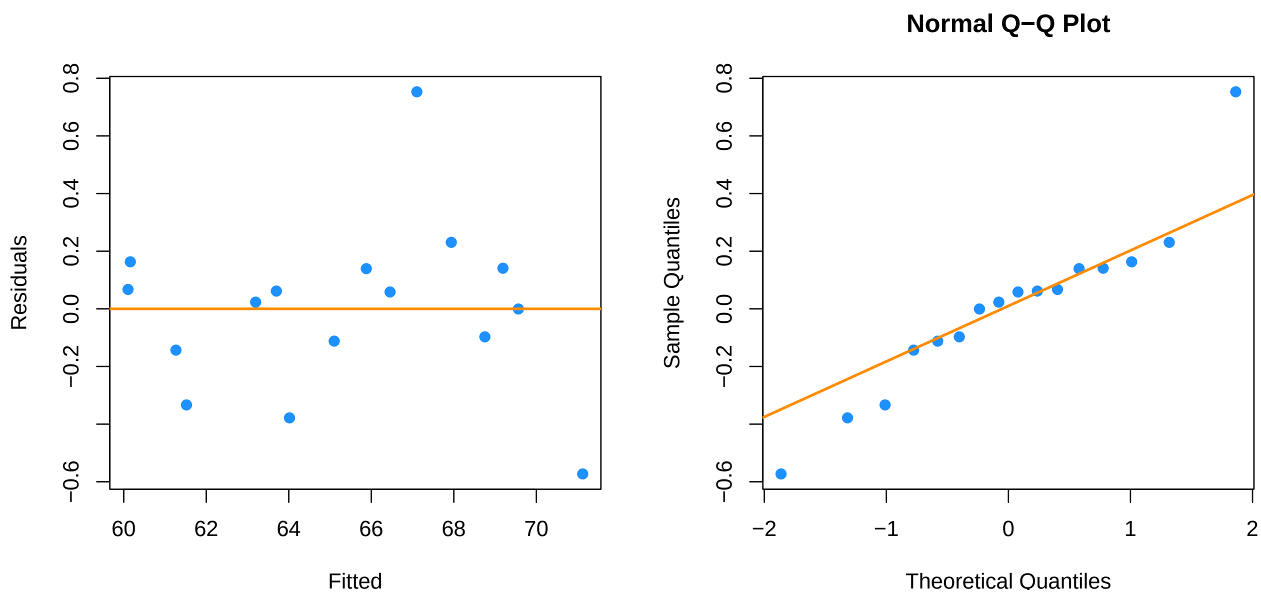
```
## [1] 0.227
```

- A decision: Based on P-value(0.227) we fail to reject null hypothesis.
- Which model you prefer, **(b)** or **(e)**: We prefer smaller model we fit from (e).

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

Solution:

```
par(mfrow = c(1,2))
plot_fitted_resid(longley_mod_sig)
plot_qq(longley_mod_sig)
```



Looking at the Fitted vs Residual plot, constant variance appears be violated Looking at the Normal Q-Q Plot, Normality assumption appears be violated

---

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:

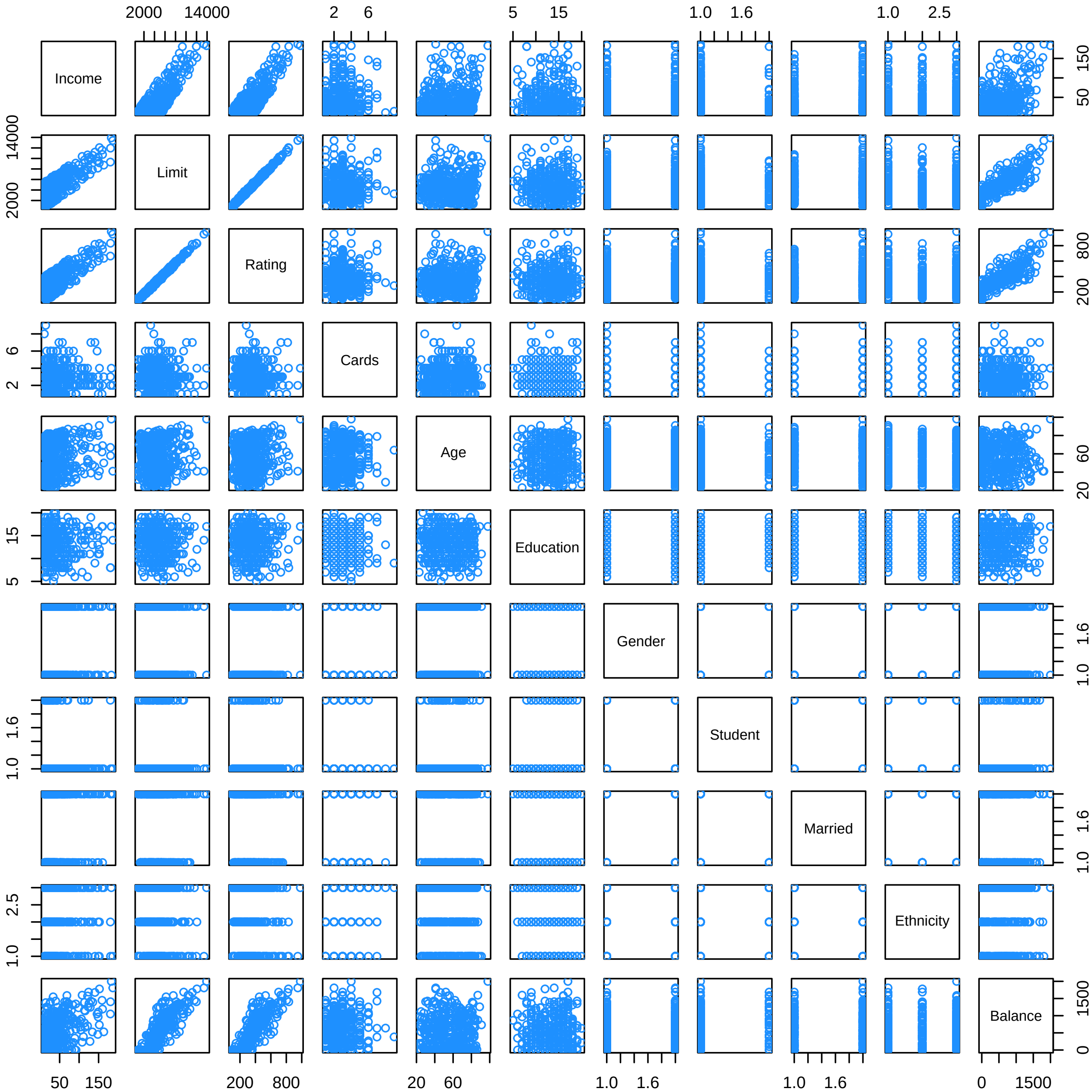
- Reach a LOOCV-RMSE below `140`
- Obtain an adjusted R^2 above `0.90`
- Fail to reject the Breusch-Pagan test with an α of 0.01
- Use fewer than 10 β parameters

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.

```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_a)
get_adj_r2(mod_a)
get_bp_decision(mod_a, alpha = 0.01)
get_num_params(mod_a)
```

```r
library(faraway)
pairs(Credit, col = "dodgerblue")
```




```r
credit_mod = lm(Balance ~., data = Credit)
library(faraway)
vif(credit_mod)
```

```
##              Income              Limit             Rating              Cards 
##               2.786            234.028            235.848              1.449 
##                 Age          Education       GenderFemale         StudentYes 
##               1.051              1.020              1.006              1.032 
##          MarriedYes     EthnicityAsian EthnicityCaucasian 
##               1.045              1.552              1.528
```

We see that Limit and Rating have very high vif. We have collinearity issue. Now we test if adding these variables to model is beneficial or not.

```r
credit_mod_small = lm(Balance ~ Limit + Cards + Age, data = Credit)
credit_mod_small2 = lm(Rating ~ Limit + Cards + Age, data = Credit)
cor(resid(credit_mod_small), resid(credit_mod_small2))
```

```
## [1] 0.0412
```

We see that there is very small correlation with variable Rating and the the variation of response Balance that is unexplained by Limit, cards, and age. We will try a model without Rating and keep Limit as predictor variable.

```r
credit_mod_back_aic = step(credit_mod, direction = "backward", k = 2, trace = 0)
credit_mod_back_aic
```

```
##
## Call:
## lm(formula = Balance ~ Income + Limit + Rating + Cards + Age +
##     Student, data = Credit)
##
## Coefficients:
## (Intercept)       Income        Limit       Rating        Cards          Age
##    -493.734       -7.795        0.194        1.091       18.212       -0.624
##  StudentYes
##     425.610
```

Backward AIC method found the model with 6 predictors(Income, Limit, Rating, Cards, Age, and Student)

```r
n = length(resid(credit_mod))
credit_mod_back_bic = step(credit_mod, direction = "backward", k = log(n), trace = 0)
credit_mod_back_bic
```

```
##
## Call:
## lm(formula = Balance ~ Income + Limit + Cards + Student, data = Credit)
##
## Coefficients:
## (Intercept)       Income        Limit        Cards   StudentYes
##    -499.727       -7.839        0.267       23.175      429.606
```

Backward BIC method found a model with 4 predictors(Income, Limit, Cards, Student)

We compare these to models.

```r
anova(credit_mod_back_bic, credit_mod_back_aic)
```

```
## Analysis of Variance Table
##
## Model 1: Balance ~ Income + Limit + Cards + Student
## Model 2: Balance ~ Income + Limit + Rating + Cards + Age + Student
##   Res.Df     RSS Df Sum of Sq   F Pr(>F)
## 1    395 3915058
## 2    393 3821620  2     93439 4.8 0.0087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given P-value of the F-test is small (< 0.01) so we can reject null hypothesis and we prefer bigger model. We test criteria with the model - Rating removed. As shown below, this model has issue with bp test which suggests linearity assumption is suspect.

```
mod_check = lm(Balance ~ Income + Limit + Cards + Age + Student, data = Credit)
get_loocv_rmse(mod_check)
```

```
## [1] 99.92
```

```
get_adj_r2(mod_check)
```

```
## [1] 0.9535
```

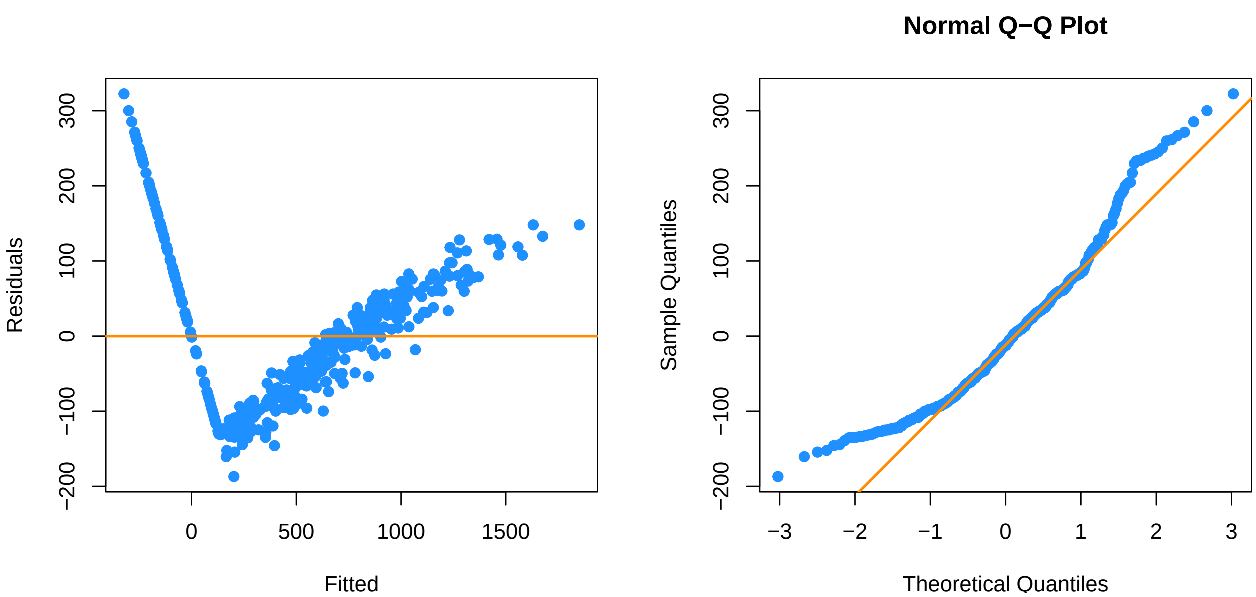
```
get_bp_decision(mod_check, alpha = 0.01)
```

```
## [1] "Reject"
```

```
get_num_params(mod_check)
```

```
## [1] 6
```

```
par(mfrow = c (1, 2))
plot_fitted_resid(mod_check)
plot_qq(mod_check)
```



We can visually confirm that the fitted vs residual plot shows that the constant variance assumption is suspect. and QQ plot show the tails that are going off.

So we try predictor transformation. First we try log on one of the predictors with smaller model.

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

```r
mod_check = cred_mod_log_Income
get_loocv_rmse(mod_check)
```

```
## [1] 130.9
```

```r
get_adj_r2(mod_check)
```

```
## [1] 0.9206
```

```r
get_bp_decision(mod_check, alpha = 0.01)
```

```
## [1] "Reject"
```

```r
get_num_params(mod_check)
```

```
## [1] 6
```

Did not improve bp test result. So we try predictor transformation. First we try log on one of the predictors with larger model.

```r
cred_mod_log_Income_all = lm(Balance ~ log(Income) + Limit + Cards + Age + Student + Education + Gender
```

```r
mod_a = cred_mod_log_Income_all
get_loocv_rmse(mod_a)
```

```
## [1] 131.5
```

```r
get_adj_r2(mod_a)
```

```
## [1] 0.9206
```

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

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

```r
get_num_params(mod_a)
```

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

This led to much better result. Passed **Goal:** .

- Reach a LOOCV-RMSE below `140` : Satisfied with 131.5 .
- Obtain an adjusted R^2 above `0.90` : Satisfied with 0.9206.
- Fail to reject the Breusch-Pagan test with an α of 0.01 : Satisfied with "Failed to Reject".

- Use fewer than 10 β parameters: Satisfied with 9.

(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 `130`
- Obtain an adjusted R^2 above `0.85`
- 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.

```
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_b)
get_adj_r2(mod_b)
get_sw_decision(mod_b, alpha = 0.01)
get_num_params(mod_b)
```

```
credit_mod_add = lm(Balance ~., data = Credit)
vif(credit_mod_add)
```

```
##            Income            Limit           Rating            Cards
##             2.786          234.028          235.848            1.449
##               Age        Education     GenderFemale       StudentYes
##             1.051            1.020            1.006            1.032
##        MarriedYes  EthnicityAsian EthnicityCaucasian
##             1.045            1.552            1.528
```

```r
get_loocv_rmse(credit_mod_add)
```

```
## [1] 100.4
```

```r
get_adj_r2(credit_mod_add)
```

```
## [1] 0.9538
```

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

```
## [1] "Reject"
```

```r
get_num_params(credit_mod_add)
```

```
## [1] 12
```

- Reach a LOOCV-RMSE below `130`
- Obtain an adjusted R^2 above `0.85`
- Fail to reject the Shapiro-Wilk test with an α of 0.01
- Use fewer than 25 β parameters The additive model satisfies other categories but fails Shapiro-Wilk test – this suggests that the model has Normality assumption violated. We also check the model selected from part a.

```r
get_loocv_rmse(mod_a)
```

```
## [1] 131.5
```

```r
get_adj_r2(mod_a)
```

```
## [1] 0.9206
```

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

```
## [1] "Reject"
```

```r
get_num_params(mod_a)
```

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

```r
credit_mod_2_back_aic = step(mod_a, direction = "backward",
                             k = 2, trace = 0)
credit_mod_2_back_aic
```

```
##
## Call:
## lm(formula = Balance ~ log(Income) + Limit + Cards + Age + Student,
##     data = Credit)
##
## Coefficients:
## (Intercept)  log(Income)        Limit        Cards          Age   StudentYes
##     424.103     -303.805        0.239       20.578       -0.993      418.439
```

This model is same as one of the model we tried in part a. cred_mod_log_Income = lm(Balance ~ log(Income) + Limit + Cards + Age + Student, data = Credit)

```r
get_loocv_rmse(cred_mod_log_Income)
```

```
## [1] 130.9
```

```r
get_adj_r2(cred_mod_log_Income)
```

```
## [1] 0.9206
```

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

```
## [1] "Reject"
```

```r
get_num_params(cred_mod_log_Income)
```

```
## [1] 6
```

So we will try to further modify the predictor variables

```r
cred_mod_log_Income_2 = lm(Balance ~ (log(Income)+ Limit + Cards + Age + Student)^2, data = Credit)
```

```r
cred_mod_log_Income_2_back_aic = step(cred_mod_log_Income_2, direction = "backward", k = 2, trace = 0)
cred_mod_log_Income_2_back_aic
```

```
##
## Call:
## lm(formula = Balance ~ log(Income) + Limit + Cards + Age + Student +
##     log(Income):Limit + log(Income):Age + log(Income):Student +
##     Limit:Cards + Limit:Student + Age:Student, data = Credit)
##
## Coefficients:
##           (Intercept)            log(Income)                  Limit
##            -259.33225             -109.36811                0.32467
##                 Cards                    Age             StudentYes
##               3.44206                3.84885              713.47635
##     log(Income):Limit        log(Income):Age  log(Income):StudentYes
##              -0.02454               -1.19012             -104.31912
##           Limit:Cards       Limit:StudentYes         Age:StudentYes
##               0.00329                0.05382               -3.18707
```

```r
get_loocv_rmse(cred_mod_log_Income_2_back_aic)
get_adj_r2(cred_mod_log_Income_2_back_aic)
get_sw_decision(cred_mod_log_Income_2_back_aic, alpha = 0.01)
get_num_params(cred_mod_log_Income_2_back_aic)
```

Solution: Found the model that satisfied tests below. lm(formula = Balance ~ log(Income) + Limit + Cards + Age + Student + log(Income):Limit + log(Income):Age + log(Income):Student + Limit:Cards + Limit:Student + Age:Student, data = Credit)

```r
mod_b = cred_mod_log_Income_2_back_aic
get_loocv_rmse(mod_b)
```

```
## [1] 118.8
```

```r
get_adj_r2(mod_b)
```

```
## [1] 0.9356
```

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

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

```r
get_num_params(mod_b)
```

```
## [1] 12
```

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.

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:faraway':
##
##     melanoma
```

```r
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))
```

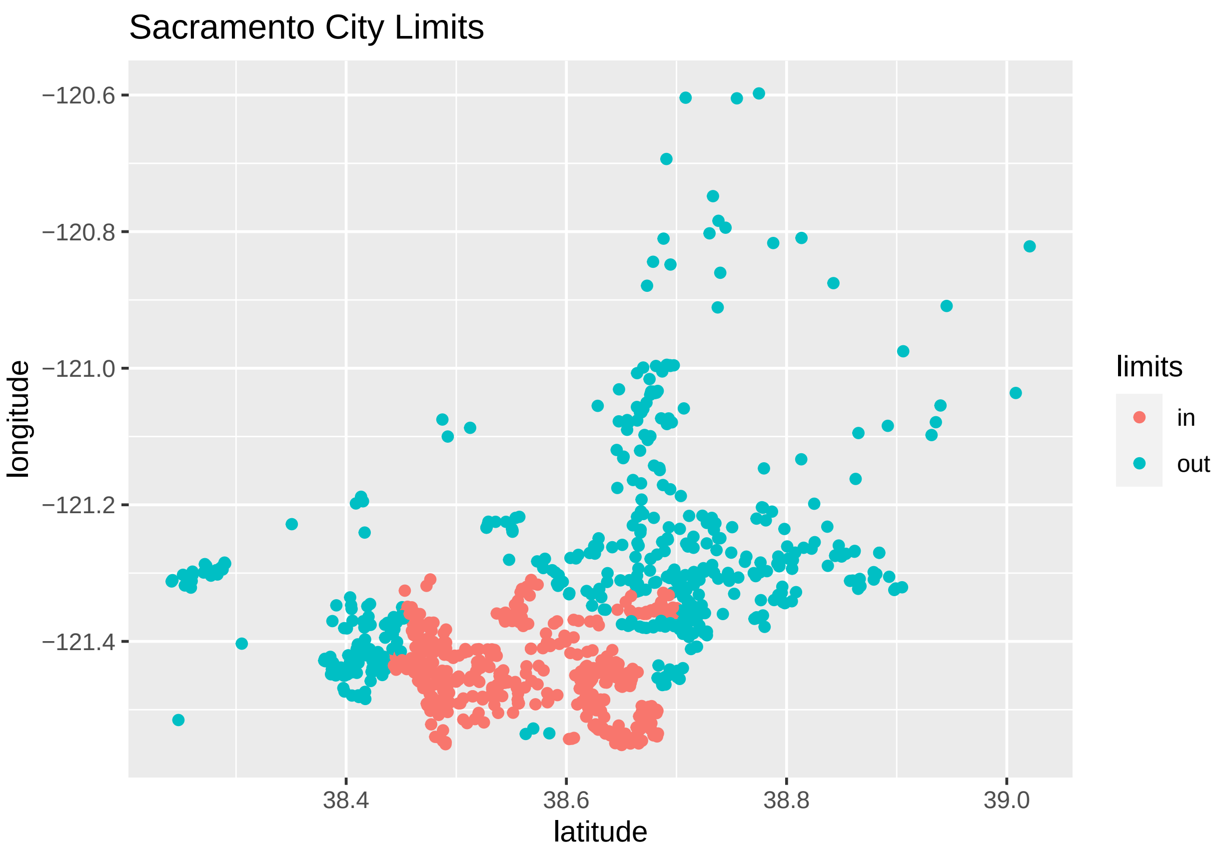
Instead of using the `city` or `zip` variables that exist in the dataset, we will simply create a variable (`limits`) indicating whether or not a house is technically within the city limits of Sacramento. (We do this because they would both be factor variables with a **large** number of levels. This 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.

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

```
## Warning: ‘qplot()‘ was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



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

```r
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.

Solution:.

```r
sac_model = lm(price ~ ., data = sac_trn_data)
get_loocv_rmse(sac_model)
```

```
## [1] 77534
```

This model's LOOCV-RMSE close but over 77,500.

```
library(leaps)
all_sac_mod = summary(regsubsets(price ~ ., data = sac_trn_data))
p = length(coef(sac_model))
n = length(resid(sac_model))
```

```
sac_mod_aic = n * log(all_sac_mod$rss / n) + 2 * (2:p)
(best_aic_ind = which.min(sac_mod_aic))
```

```
## [1] 5
```

```
all_sac_mod$which[best_aic_ind,]
```

```
##      (Intercept)             beds            baths             sqft
##             TRUE             TRUE            FALSE             TRUE
## typeMulti_Family  typeResidential         latitude        longitude
##            FALSE             TRUE             TRUE             TRUE
##        limitsout
##            FALSE
```

```
(best_r2_idx = which.max(all_sac_mod$adjr2))
```

```
## [1] 6
```

```
all_sac_mod$which[best_r2_idx, ]
```

```
##      (Intercept)             beds            baths             sqft
##             TRUE             TRUE            FALSE             TRUE
## typeMulti_Family  typeResidential         latitude        longitude
##            FALSE             TRUE             TRUE             TRUE
##        limitsout
##             TRUE
```

```
sac_mod_back_aic = step(sac_model, direction = "backward", k = 2, trace = 0)
sac_mod_back_aic
```

```
##
## Call:
## lm(formula = price ~ beds + sqft + type + latitude + longitude,
##     data = sac_trn_data)
##
## Coefficients:
##      (Intercept)             beds             sqft  typeMulti_Family
##         13662797           -24868              152             22191
##  typeResidential         latitude        longitude
##            35022            56620           130246
```

```r
n = length(resid(sac_model))
sac_mod_back_bic = step(sac_model, direction = "backward", k = log(n), trace = 0)
sac_mod_back_bic
```

```
##
## Call:
## lm(formula = price ~ beds + sqft + longitude, data = sac_trn_data)
##
## Coefficients:
## (Intercept)         beds         sqft    longitude
##    18208818       -21659          151       149505
```

```r
get_loocv_rmse(sac_mod_back_aic)
```

```
## [1] 77393
```

```r
get_loocv_rmse(sac_mod_back_bic)
```

```
## [1] 77629
```

Where we see that the model found using Backward AIC method is just satisfying `LOOCV RMSE below 77500` requirement. lm(formula = price ~ beds + sqft + type + latitude + longitude, data = sac_trn_data)

```r
vif(sac_mod_back_aic)
```

```
##             beds             sqft typeMulti_Family  typeResidential
##            2.374            2.190            1.238            1.337
##         latitude        longitude
##            1.167            1.241
```

We will proceed with this model.

```r
sac_model_a = sac_mod_back_aic
```

(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:

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

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

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

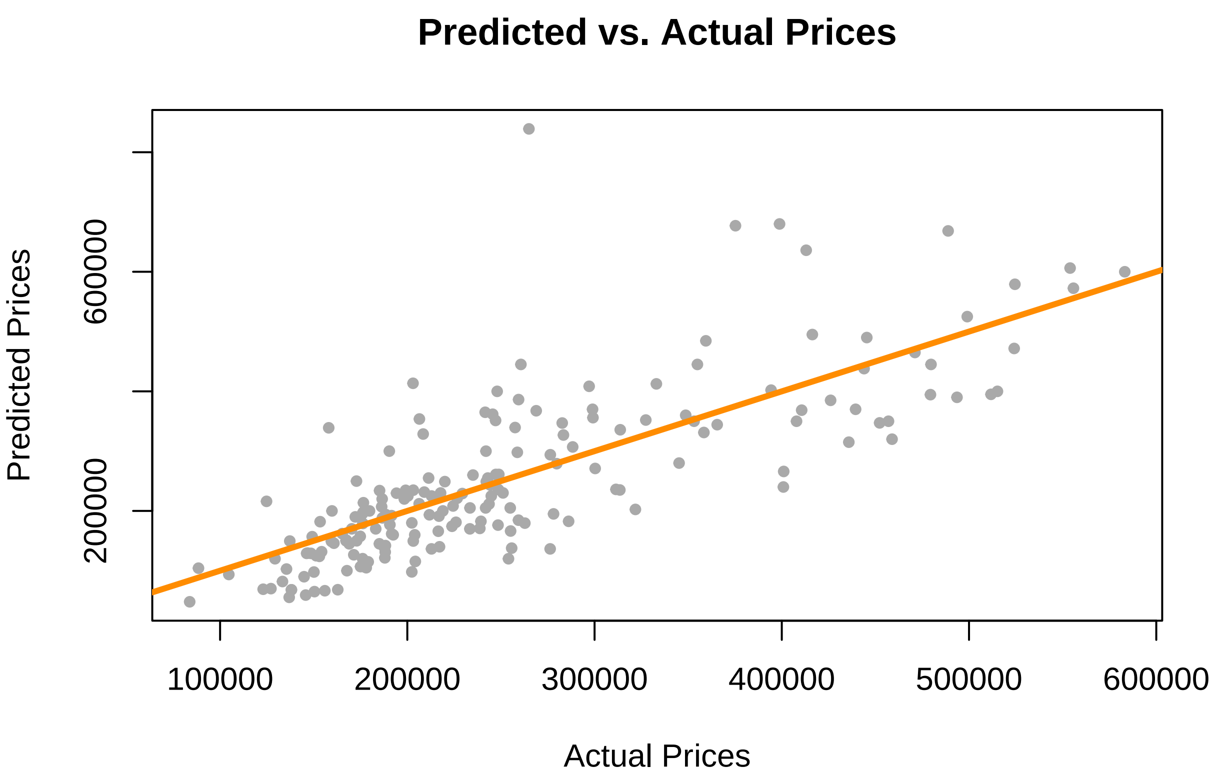
To find out if LOOCV-RMSE below 77,500 is enough for predicting home prices, we further investigate. Calculate the average percent error.

```r
actual = sac_tst_data$price
predicted = predict(sac_model_a, newdata = sac_tst_data)
```

```r
100 * mean(abs(predicted - actual) / predicted)
```

```
## [1] 24.83
```

```r
plot(actual ~ predicted, col = "darkgrey", pch = 20,
     xlab = "Actual Prices",
     ylab = "Predicted Prices",
     main = "Predicted vs. Actual Prices")
abline(a = 0, b = 1, lwd = 3, col = "darkorange")
```



From the plot and the average percentage error value calculated above, we can see that the usefulness of LOOCV-RMSE of 77,500 is highly depending on the price. If we are to use this LOOCV-RMSE for prediction that falls in the range between roughly 100,000 to 250,000, error of 77,500 is huge percentage whereas if we are talking about houses priced above 500,000, 77,500 is much small percentage. So the usefulness of error of 77,500 might be more acceptable towards higher priced property market. The average percentage of around 25% might not be very useful in this case. And we do see that there are few outliers that are predicted way higher than actual prices.

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.

```r
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.

```r
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:

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

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

To detect the false positives, use:

```r
# 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.

Solution:

```r
set.seed(19870503)
num_sims = 300

fal_pos_aic = rep(0, num_sims)
fal_pos_bic = rep(0, num_sims)
fal_neg_aic = rep(0, num_sims)
fal_neg_bic = rep(0, num_sims)

for(i in 1:num_sims){
  # simulate the response variable `y` 300 times
   sim_data_1$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)

  # Fit an additive model using each of the `x` variables
   fit_model_1 = lm(y ~ . , data = sim_data_1)

  # Perform variable selection using backwards AIC
   fit_model_1_back_aic = step(fit_model_1, direction = "backward", k = 2, trace = 0)
  # Perform variable selection using backwards BIC.
   n = length(resid(fit_model_1))
   fit_model_1_back_bic = step(fit_model_1, direction = "backward", k = log(n), trace = 0)

  # Calculate and store the number of false negatives for the models chosen by AIC and BIC.
   fal_neg_aic[i] = sum(!(signif %in% names(coef(fit_model_1_back_aic))))
   fal_neg_bic[i] = sum(!(signif %in% names(coef(fit_model_1_back_bic))))
  # Calculate and store the number of false positives for the models chosen by AIC and BIC.
   fal_pos_aic[i] = sum(names(coef(fit_model_1_back_aic)) %in% not_sig)
   fal_pos_bic[i] = sum(names(coef(fit_model_1_back_bic)) %in% not_sig)
}
```

```r
 # Calculate the rate of false positives and negatives for both AIC and BIC
fal_pos_aic_rate = sum(fal_pos_aic) / num_sims
fal_pos_bic_rate = sum(fal_pos_bic) / num_sims
fal_neg_aic_rate = sum(fal_neg_aic) / num_sims
fal_neg_bic_rate = sum(fal_neg_bic) / num_sims
```

```r
  # Compare the rates between the two methods. Arrange in a well formatted table.

comp_table = data.frame(Method = c("AIC", "BIC"),
                        False_Positive_Rate = c(fal_pos_aic_rate, fal_pos_bic_rate),
                        False_Negative_Rate = c(fal_neg_aic_rate, fal_neg_bic_rate)
)

print(comp_table)
```

```
##   Method False_Positive_Rate False_Negative_Rate
## 1    AIC              0.9133                   0
## 2    BIC              0.1600                   0
```

False-Negative for both AIC and BIC methods are 0. That is, the models are likely bigger than being smaller. AIC method seems producing more False-Positive results, as expected. BIC method produces smaller models in general compared to AIC method.

(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(19870503)
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)
)
```

```
set.seed(19870503)
num_sims = 300

fal_pos_aic_2 = rep(0, num_sims)
fal_pos_bic_2 = rep(0, num_sims)
fal_neg_aic_2 = rep(0, num_sims)
fal_neg_bic_2 = rep(0, num_sims)

for(i in 1:num_sims){
  # simulate the response variable `y` 300 times
   sim_data_2$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)

  # Fit an additive model using each of the `x` variables
   fit_model_2 = lm(y ~ . , data = sim_data_2)

  # Perform variable selection using backwards AIC
   fit_model_2_back_aic = step(fit_model_2, direction = "backward", k = 2, trace = 0)
  # Perform variable selection using backwards BIC.
   n = length(resid(fit_model_1))
   fit_model_2_back_bic = step(fit_model_2, direction = "backward", k = log(n), trace = 0)

  # Calculate and store the number of false negatives for the models chosen by AIC and BIC.
```

```r
  fal_neg_aic_2[i] = sum(!(signif %in% names(coef(fit_model_2_back_aic))))
  fal_neg_bic_2[i] = sum(!(signif %in% names(coef(fit_model_2_back_bic))))
 # Calculate and store the number of false positives for the models chosen by AIC and BIC.
  fal_pos_aic_2[i] = sum(names(coef(fit_model_2_back_aic)) %in% not_sig)
  fal_pos_bic_2[i] = sum(names(coef(fit_model_2_back_bic)) %in% not_sig)
}
```

```r
 # Calculate the rate of false positives and negatives for both AIC and BIC
fal_pos_aic_rate_2 = sum(fal_pos_aic_2) / num_sims
fal_pos_bic_rate_2 = sum(fal_pos_bic_2) / num_sims
fal_neg_aic_rate_2 = sum(fal_neg_aic_2) / num_sims
fal_neg_bic_rate_2 = sum(fal_neg_bic_2) / num_sims
```

```r
  # Compare the rates between the two methods. Arrange in a well formatted table.

comp_table_2 = data.frame(Method = c("AIC", "BIC"),
                        False_Positive_Rate = c(fal_pos_aic_rate_2, fal_pos_bic_rate_2),
                        False_Negative_Rate = c(fal_neg_aic_rate_2, fal_neg_bic_rate_2)
)

print(comp_table_2)
```

```
##   Method False_Positive_Rate False_Negative_Rate
## 1    AIC               1.603              0.8400
## 2    BIC               1.127              0.9133
```

The results show that there are False-Negative for both AIC and BIC methods this time. Also, we see increase in False_positive rate in both AIC and BIC.

There are some changes in the variable configurations for simulation in `part b` x_8, x_9, and x_10 values depend on the x_1 and x_2 values which means there are added correlation. Multicolinearity, when predictor variables are highly correlated, can result highly variable estimates, increasing variability. Also, it can lead to also increased chance of selecting redundant effect variable in the model which can lead to in false positive which we are seeing from the results. Multicollinearity issue is also contribute to the increased chance of leaving out significant variables and increase in False-positive rate or including non-significant variables and increase in False-negative rate.