Week 9 - Homework

STAT 420, Summer 2022, Ilya Andreev, iandre3@illinois.edu

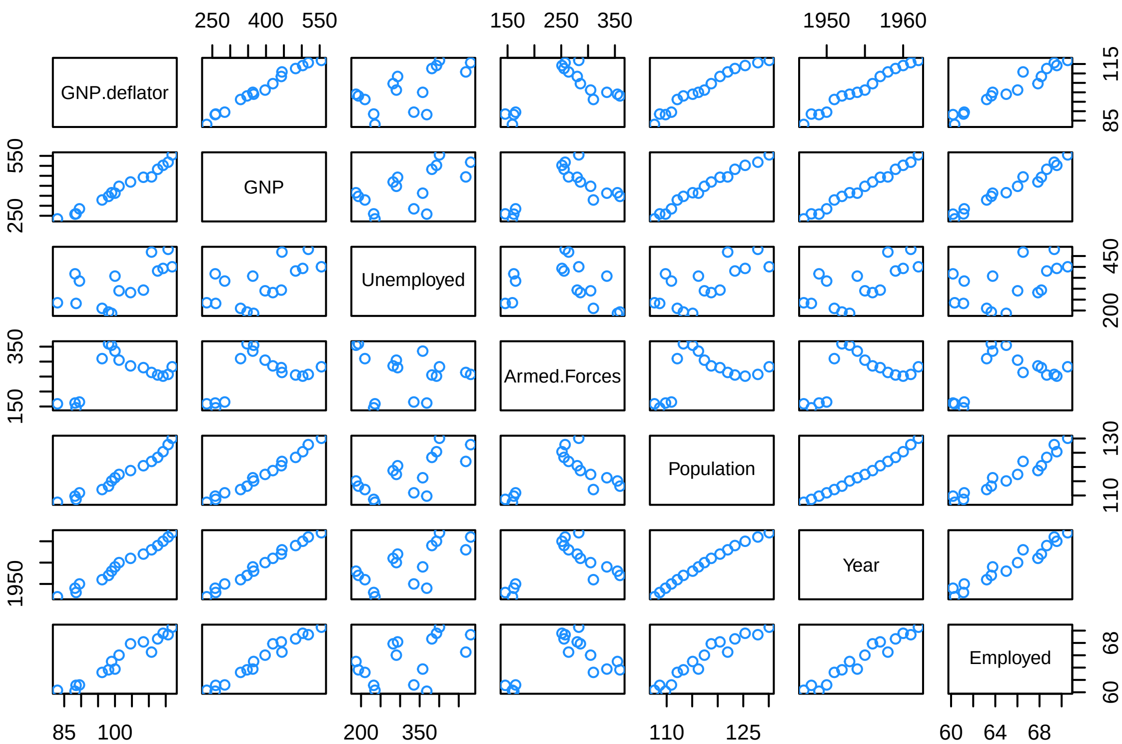
Exercise 1 (`longley` Macroeconomic Data)

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?

```
pairs(longley, col = "dodgerblue")
```




```
round(cor(longley), 3)
```

```
##              GNP.deflator   GNP Unemployed Armed.Forces Population  Year
## GNP.deflator        1.000 0.992      0.621        0.465      0.979 0.991
## GNP                 0.992 1.000      0.604        0.446      0.991 0.995
## Unemployed          0.621 0.604      1.000       -0.177      0.687 0.668
## Armed.Forces        0.465 0.446     -0.177        1.000      0.364 0.417
## Population          0.979 0.991      0.687        0.364      1.000 0.994
## Year                0.991 0.995      0.668        0.417      0.994 1.000
```

```
## Employed            0.971 0.984      0.502        0.457      0.960 0.971
##              Employed
## GNP.deflator    0.971
## GNP             0.984
## Unemployed      0.502
## Armed.Forces    0.457
## Population      0.960
## Year            0.971
## Employed        1.000
```

The highest correlation 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(car)
```

```
## Loading required package: carData
```

```
m = lm(Employed ~ ., longley)
vif(m)
```

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

```
which.max(vif(m))
```

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

```
vif(m) > 5
```

```
## GNP.deflator          GNP   Unemployed Armed.Forces   Population         Year
##         TRUE         TRUE         TRUE        FALSE         TRUE         TRUE
```

GNP has the largest VIF. VIFs of GNP.deflator, GNP, Unemployed, Population, and Year suggest colinearity.

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

```
m = lm(Population ~ . - Employed, longley)
```

0.9975 is the proportion of variation in `Population` that is explained by a linear relationship with other predictors.

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

```
full_model = lm(Employed ~ . - Population, longley)
added_predictor_model = lm(Population ~ . - Employed, longley)
```

The correlation coefficient is -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?

```
m = lm(Employed ~ ., longley)
summary(m)
```

```
##
```

```
## Call:
## lm(formula = Employed ~ ., data = longley)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4101 -0.1577 -0.0282  0.1016  0.4554
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.48e+03   8.90e+02   -3.91  0.00356 **
## GNP.deflator  1.51e-02   8.49e-02    0.18  0.86314
## GNP          -3.58e-02   3.35e-02   -1.07  0.31268
## Unemployed   -2.02e-02   4.88e-03   -4.14  0.00254 **
## Armed.Forces -1.03e-02   2.14e-03   -4.82  0.00094 ***
## Population   -5.11e-02   2.26e-01   -0.23  0.82621
## Year          1.83e+00   4.55e-01    4.02  0.00304 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.305 on 9 degrees of freedom
## Multiple R-squared:  0.995,  Adjusted R-squared:  0.992
## F-statistic:  330 on 6 and 9 DF,  p-value: 4.98e-10
```

```
significant_model = lm(Employed ~ Unemployed + Armed.Forces + Year, longley)
vif(significant_model)
```

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

```
which.max(vif(significant_model))
```

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

This time, `Year` has the largest VIF, but even that VIF is not large enough to suggest co-linearity issues.

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

- The null hypothesis
- The test statistic
- The distribution of the test statistic under the null hypothesis
- The p-value
- A decision
- Which model you prefer, **(b)** or **(e)**

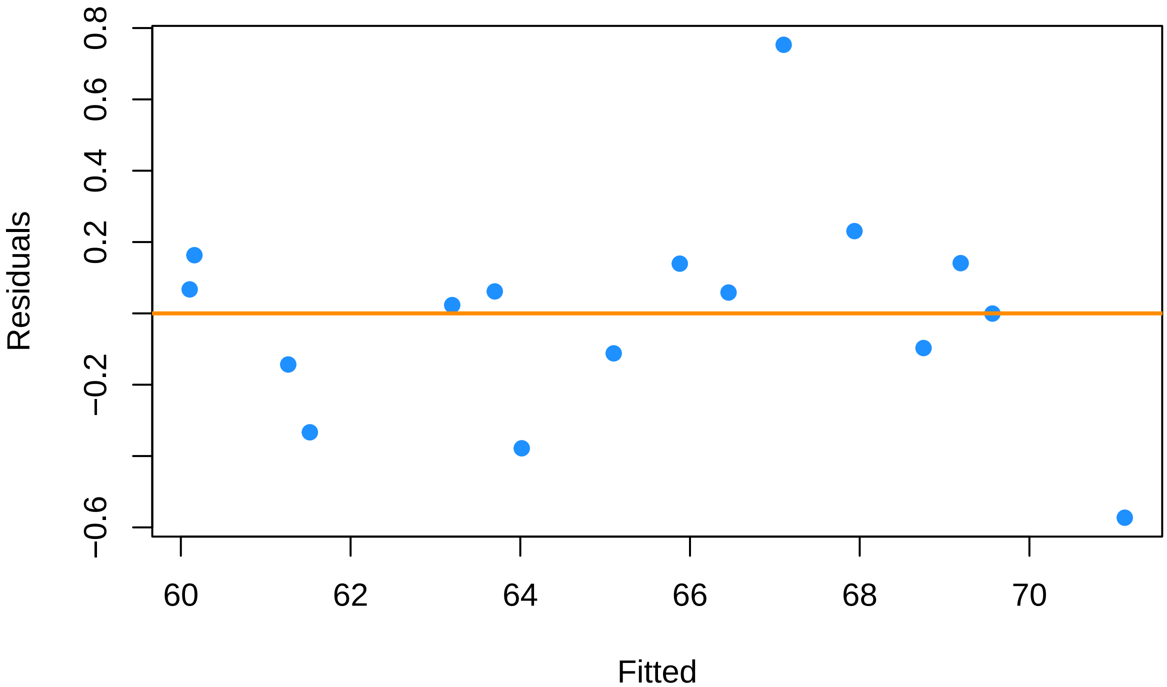
```
m1 = lm(Employed ~ ., longley)
m2 = lm(Employed ~ Unemployed + Armed.Forces + Year, longley)
res = anova(m2, m1)
res
```

```
## Analysis of Variance Table
##
## Model 1: Employed ~ Unemployed + Armed.Forces + Year
## Model 2: Employed ~ GNP.deflator + GNP + Unemployed + Armed.Forces + Population +
##     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
```

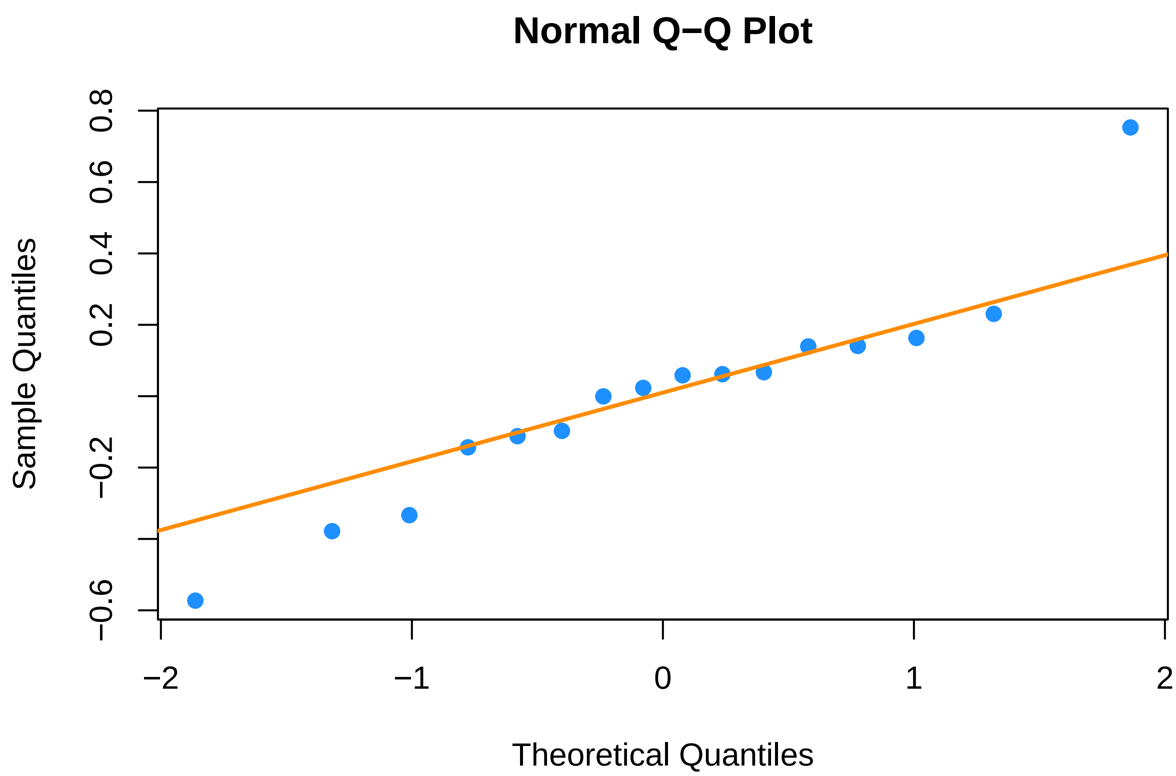
The null hypothesis states that the true coefficients of `GNP.deflator`, `GNP`, `Population` are 0 when `Unemployed`, `Armed.Forces`, and `Year` are already present in the model. The test statistic is 1.7465. The distribution of the test statistic under the null hypothesis is the F distribution with 12 and 9 degrees of freedom. The p-value is 0.227. The decision is to fail to reject the null hypothesis. The model from **(e)** is the preferred model.

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

```
m = lm(Employed ~ Unemployed + Armed.Forces + Year, longley)
plot_fitted_resid(m)
```




```
plot_qq(m)
```



The fitted-residuals plot suggests no violation of the linearity and equal variance assumptions. The Q-Q plot suggests no violation of the normality assumption.

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))
```

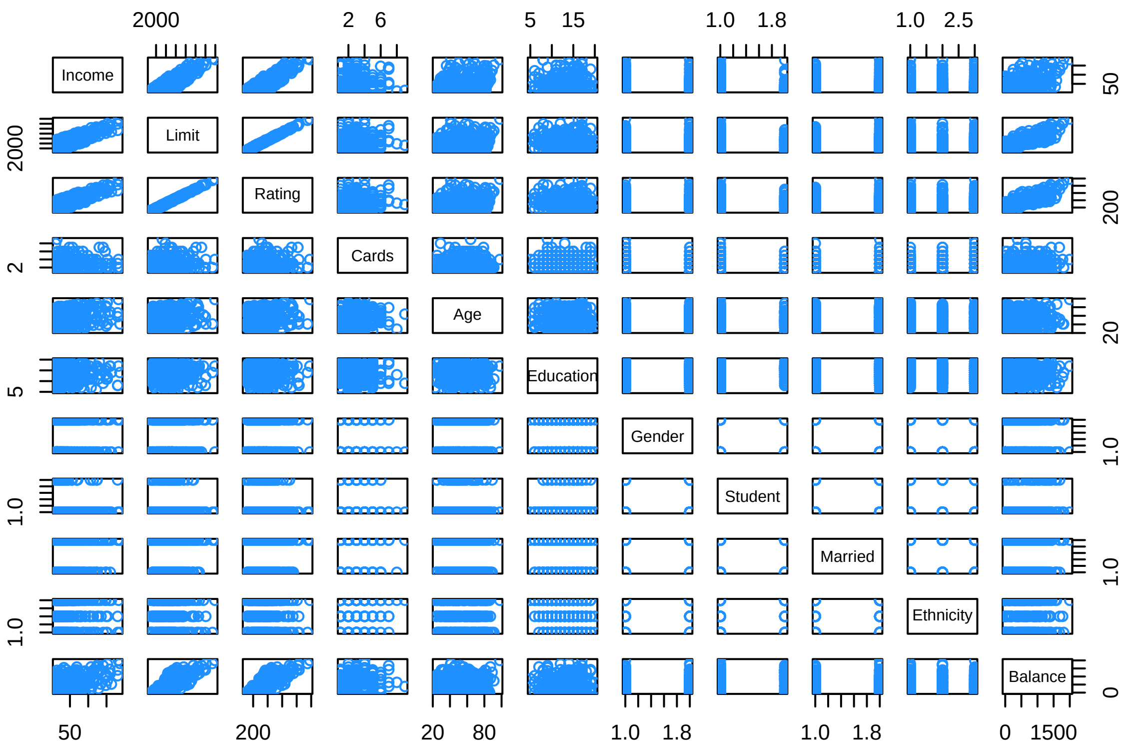
Use `?Credit` to learn about this dataset.

(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

```
Credit$Gender = as.factor(Credit$Gender)
Credit$Student = as.factor(Credit$Student)
Credit$Married = as.factor(Credit$Married)
Credit$Ethnicity = as.factor(Credit$Ethnicity)
mod_a = lm(Balance ~
             poly(Income, 2) +
             poly(Limit, 2) +
             poly(Rating, 2) +
             poly(Cards, 2) +
```

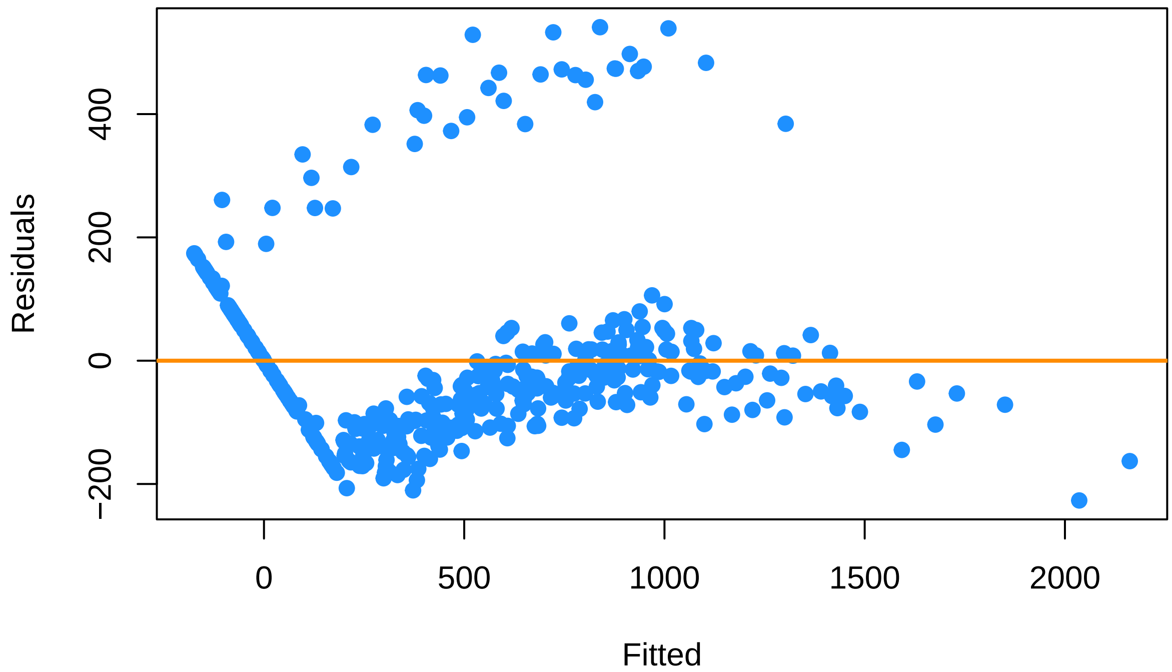
```
            poly(Age, 2) +
            poly(Education, 2), Credit)

pairs(Credit, col = "dodgerblue")
```



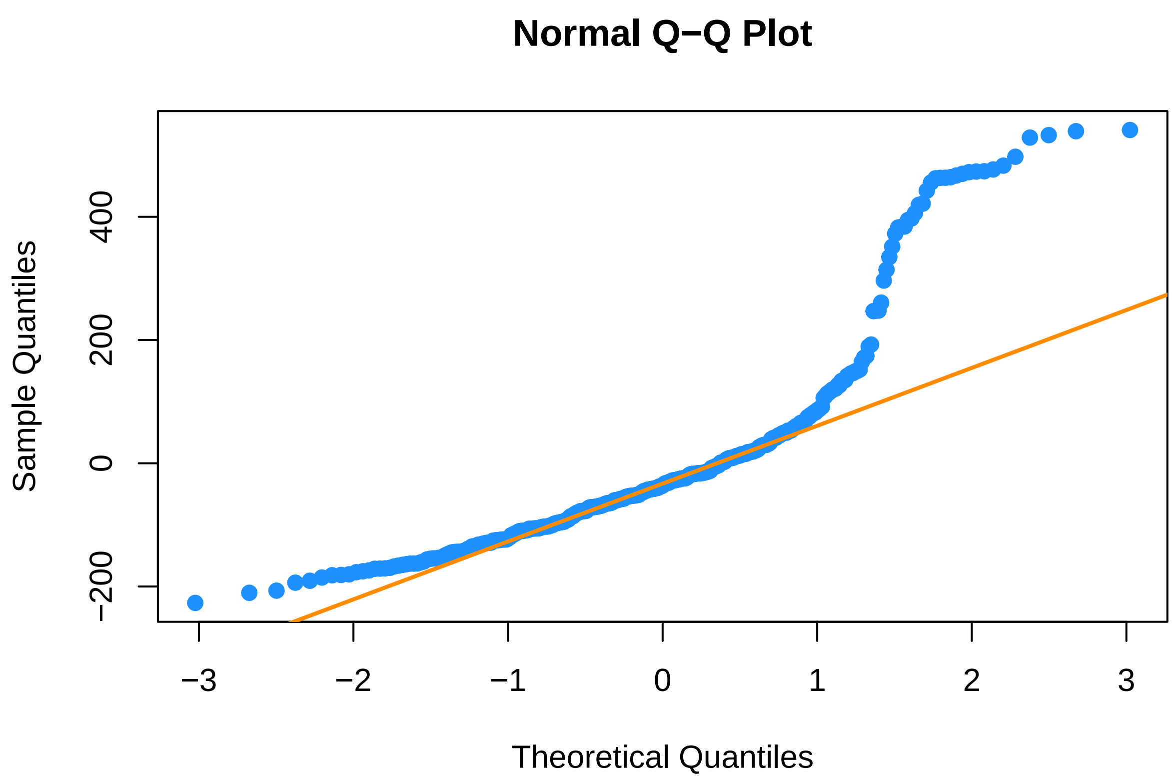

```
round(cor(Credit[ , -which(names(Credit) %in% c("Gender","Student", "Married", "Ethnicity"))]), 3)
```

```
##           Income  Limit Rating  Cards   Age Education Balance
## Income     1.000  0.792  0.791 -0.018 0.175    -0.028   0.464
## Limit      0.792  1.000  0.997  0.010 0.101    -0.024   0.862
## Rating     0.791  0.997  1.000  0.053 0.103    -0.030   0.864
## Cards     -0.018  0.010  0.053  1.000 0.043    -0.051   0.086
## Age        0.175  0.101  0.103  0.043 1.000     0.004   0.002
## Education -0.028 -0.024 -0.030 -0.051 0.004     1.000  -0.008
## Balance    0.464  0.862  0.864  0.086 0.002    -0.008   1.000
```

```
res = step(mod_a, direction = "both", trace = 0)
mod_a = res
plot_fitted_resid(mod_a)
```




```
plot_qq(mod_a)
```



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.

```
library(lmtest)

get_bp_decision = function(model, alpha) {
```

```r
  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
}
```

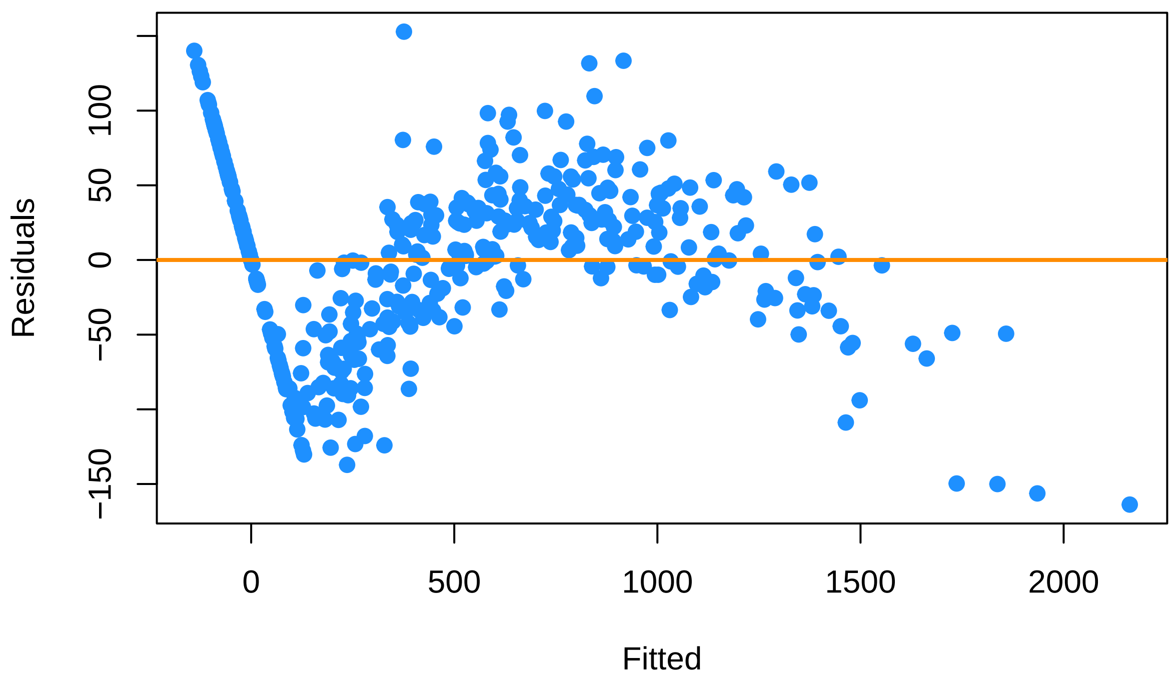
```r
get_loocv_rmse(mod_a)
get_adj_r2(mod_a)
get_bp_decision(mod_a, alpha = 0.01)
get_num_params(mod_a)
```

(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:

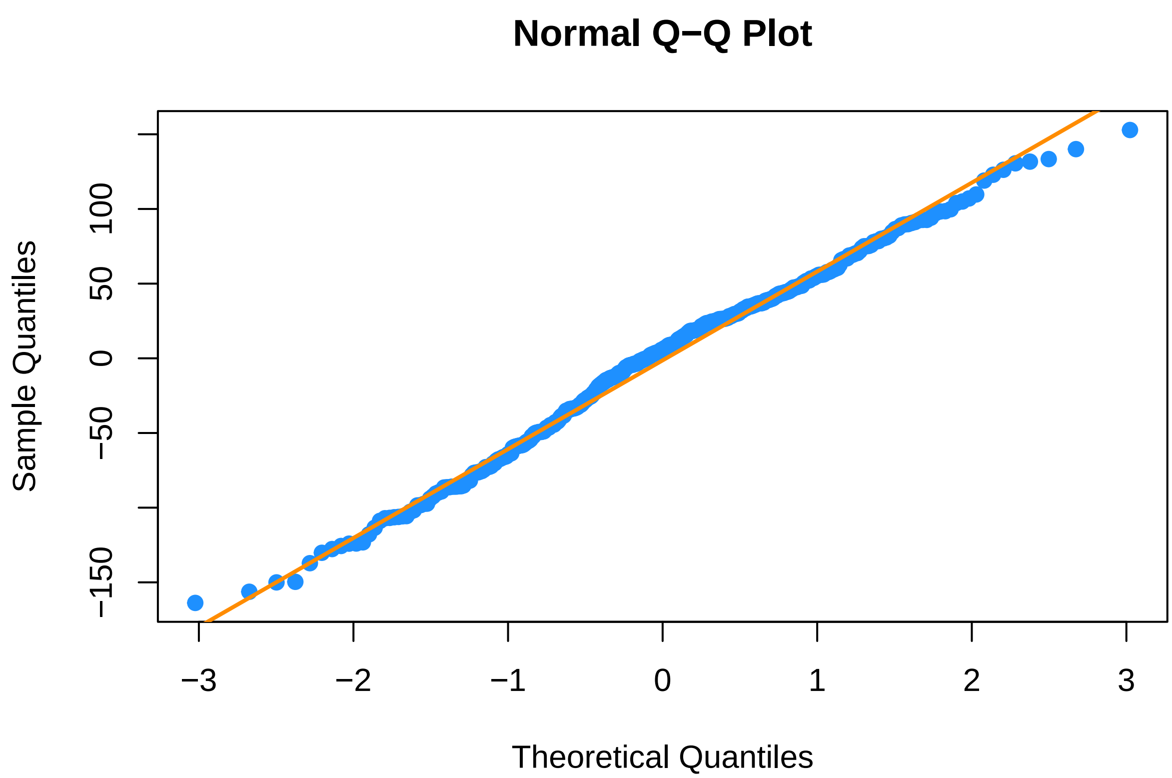
- Reach a LOOCV-RMSE below `130`
- Obtain an adjusted R^2 above `0.85`
- Fail to reject the Shapiro-Wilk test with an α of 0.01
- Use fewer than 25 β parameters

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.

```r
mod_b = lm(Balance ~ . ^ 2 - Cards - Education - Ethnicity - Gender - Student, Credit)
res = step(mod_b, direction = "both", trace = 0)
mod_b = res
plot_fitted_resid(mod_b)
```




```
plot_qq(mod_b)
```




```
library(lmtest)

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

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

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
```

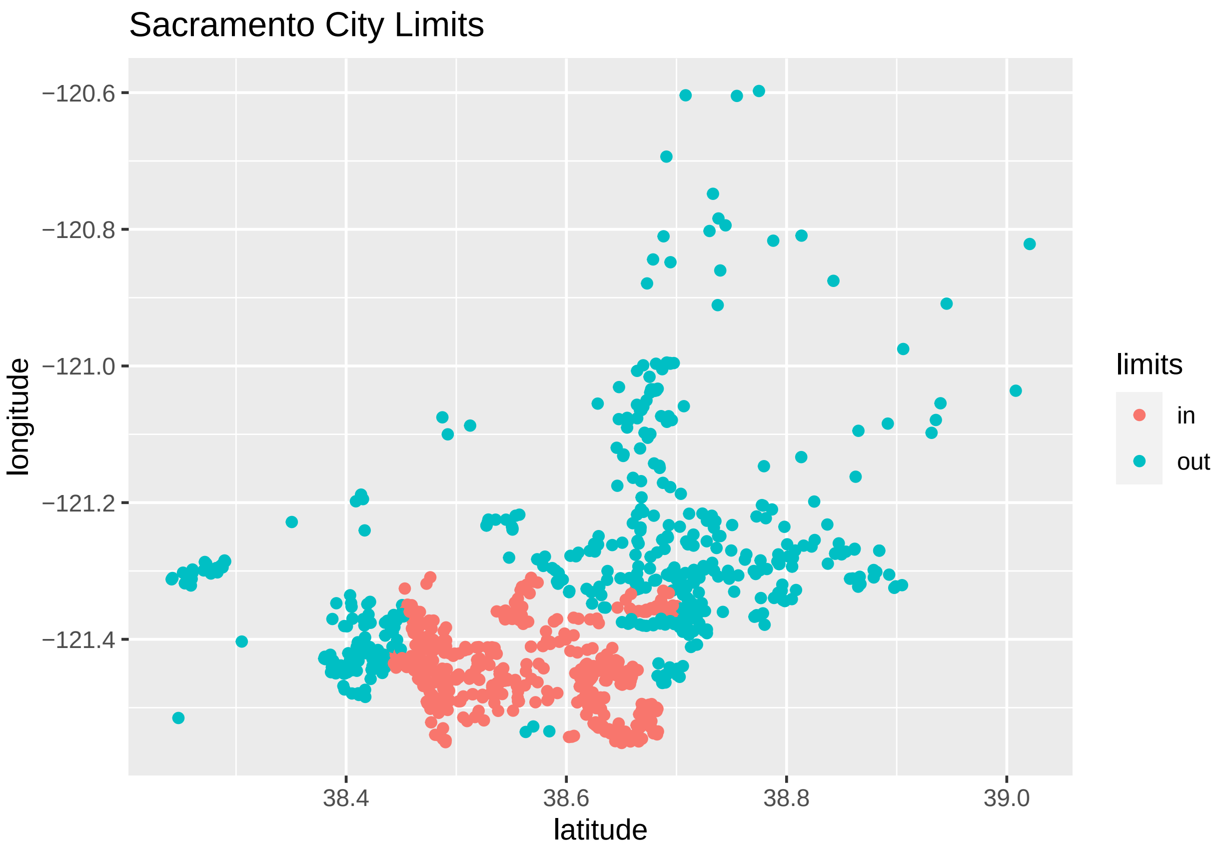
```r
library(ggplot2)
data(Sacramento)
sac_data = Sacramento
sac_data$limits = factor(ifelse(sac_data$city == "SACRAMENTO", "in", "out"))
sac_data = subset(sac_data, select = -c(city, zip))
```

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 ")
```



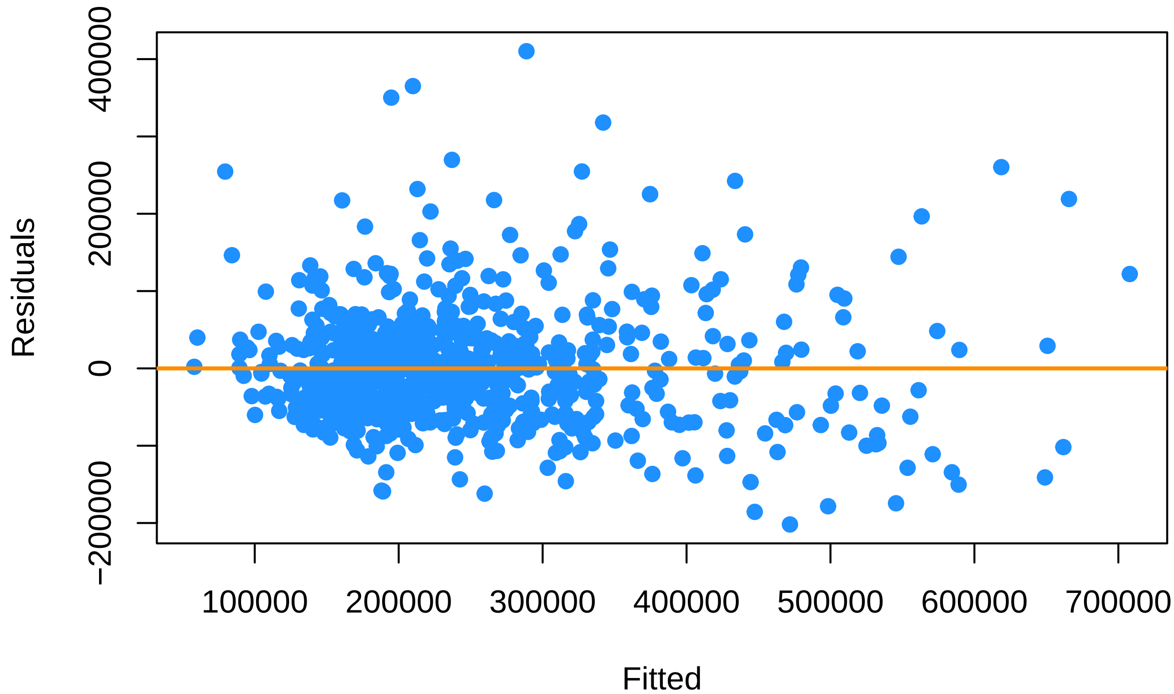
After these modifications, we test-train split the data.

```
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, ]
```

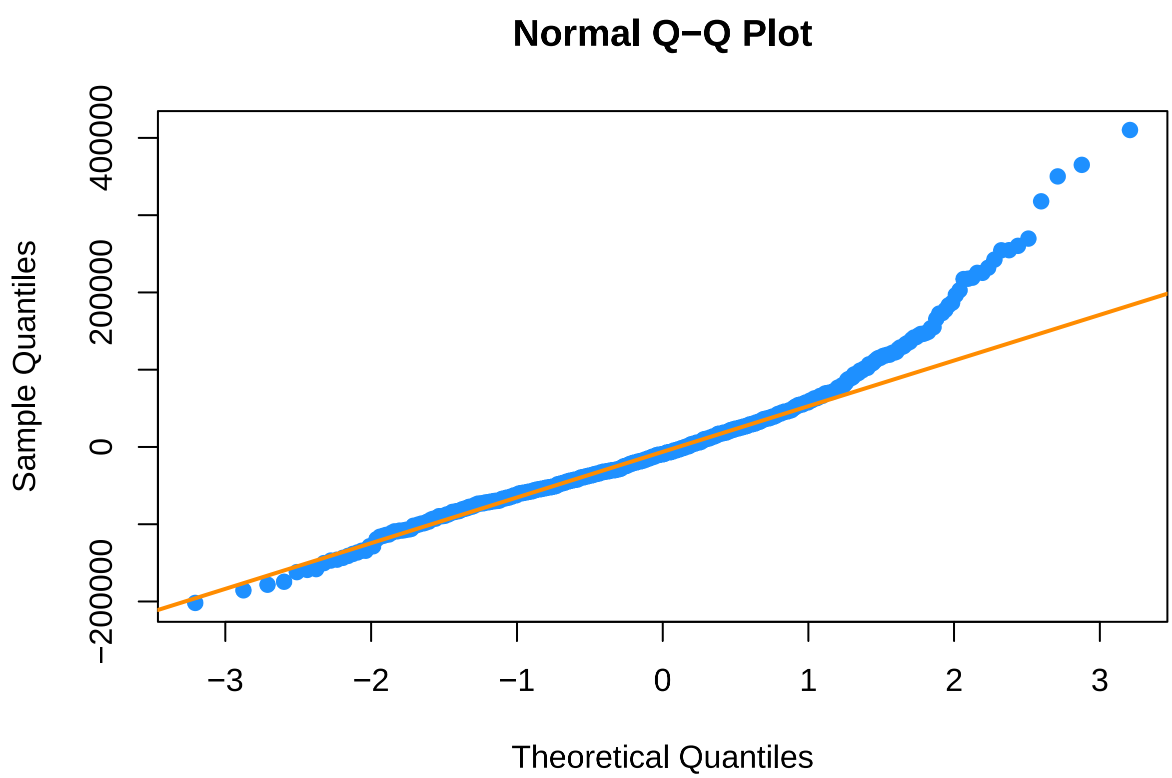
The training data should be used for all model fitting. Our goal is to find a model that is useful for predicting home prices.

(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.

```
mod_a = lm(price ~ . ^ 2, sac_trn_data)
res = step(mod_a, direction = "both", trace = 0)
mod_a = res
plot_fitted_resid(mod_a)
```




```
plot_qq(mod_a)
```




```
get_loocv_rmse(mod_a)
```

```
## [1] 75375
```

```r
get_adj_r2(mod_a)
```

```
## [1] 0.6639
```

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

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

```r
get_num_params(mod_a)
```

```
## [1] 14
```

(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:

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

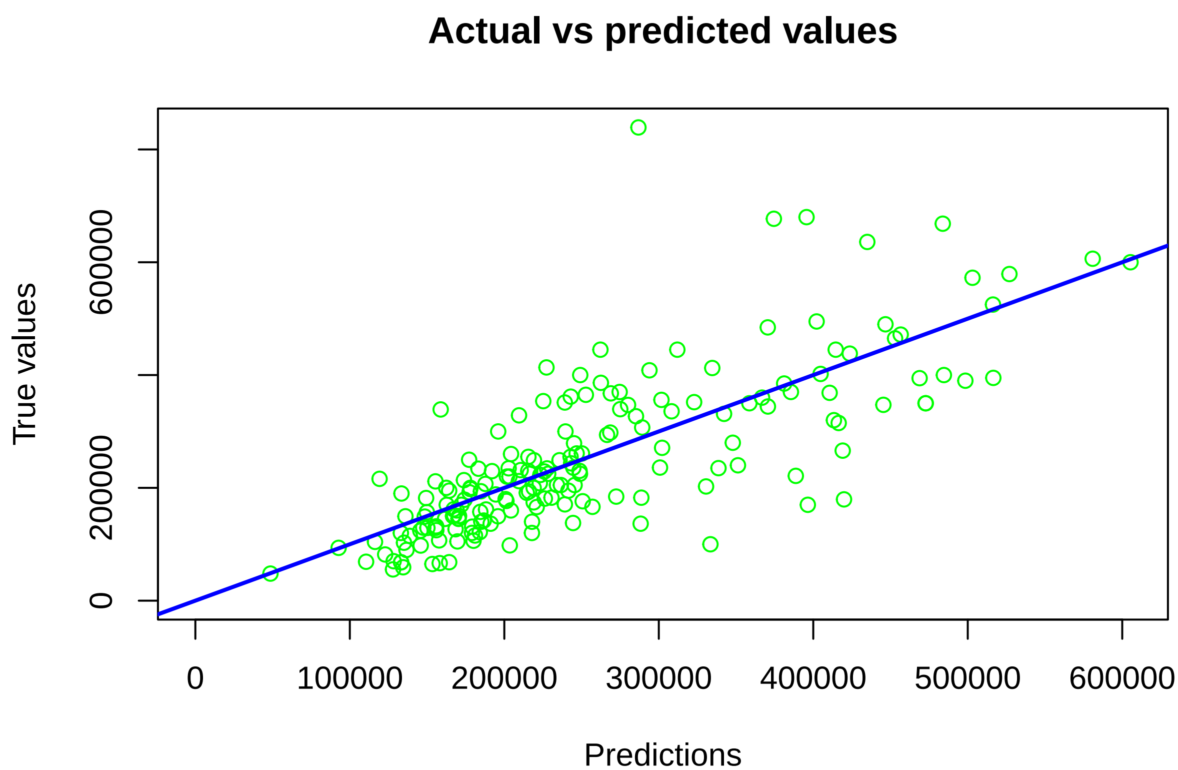
- Plot the predicted versus the actual values and add the line $y = x$.

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

```r
predictions = predict(mod_a, sac_tst_data)
sum(
  abs(predictions - sac_tst_data$price) / predict(mod_a, sac_tst_data)
  ) * 100 / nrow(predictions)
```

```
## numeric(0)
```

```r
plot(
  predictions,
  sac_tst_data$price,
  col = "green",
  xlim = c(0, max(predictions)),
  ylim = c(0, max(sac_tst_data$price)),
  ylab = "True values",
  xlab = "Predictions",
  main = "Actual vs predicted values")
abline(0, 1, lty=1, col="blue", lwd=2)
```



Visually, the model clearly has a fit, but the size of the error is often so large the model can be hardly called reliable.

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:

```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.

```r
set.seed(24111999)

fn_bic = rep(0, 300)
fn_aic = rep(0, 300)
fp_bic = rep(0, 300)
fp_aic = rep(0, 300)

for (i in 1:300) {
  sim_data = 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))
  m = lm(y ~ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_10, sim_data)
  summary(m)
  back_aic = step(m, direction = "backward", trace = 0)
  summary(back_aic)
  fn_aic[i] = sum(!(signif %in% names(coef(back_aic))))
  fp_aic[i] = sum(names(coef(back_aic)) %in% not_sig)
  back_bic = step(m, direction = "backward", trace = 0, k = log(n))
  fn_bic[i] = sum(!(signif %in% names(coef(back_bic))))
  fp_bic[i] = sum(names(coef(back_bic)) %in% not_sig)
}
```

```r
res = cbind(
  c(mean(fp_aic),
    mean(fn_aic)),
  c(mean(fp_bic),
    mean(fn_bic)))
colnames(res) = c('AIC', 'BIC')
rownames(res) = c('Average false positive params', 'Average false negative params')
knitr::kable(res)
```

	AIC	BIC
Average false positive params	0.8733	0.2433
Average false negative params	0.0000	0.0000

(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.

```r
set.seed(94)
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)
)
```

```r
set.seed(24111999)

fn_bic = rep(0, 300)
fn_aic = rep(0, 300)
fp_bic = rep(0, 300)
fp_aic = rep(0, 300)

for (i in 1:300) {
  sim_data = 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))
  m = lm(y ~ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_10, sim_data)
  summary(m)
  back_aic = step(m, direction = "backward", trace = 0)
  summary(back_aic)
  fn_aic[i] = sum(!(signif %in% names(coef(back_aic))))
  fp_aic[i] = sum(names(coef(back_aic)) %in% not_sig)
  back_bic = step(m, direction = "backward", trace = 0, k = log(n))
  fn_bic[i] = sum(!(signif %in% names(coef(back_bic))))
  fp_bic[i] = sum(names(coef(back_bic)) %in% not_sig)
```

```
}
```

```
res = cbind(
  c(mean(fp_aic),
    mean(fn_aic)),
  c(mean(fp_bic),
    mean(fn_bic)))
colnames(res) = c('AIC', 'BIC')
rownames(res) = c('Average false positive params', 'Average false negative params')
knitr::kable(res)
```

	AIC	BIC
Average false positive params	1.503	1.003
Average false negative params	0.810	0.860