Week 9 - Homework

STAT 420, Summer 2023, D. Unger

Scott Girten
NetID: sgirten2

Directions

Students are encouraged to work together on homework. However, sharing, copying or providing any part of a homework solution or code is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible.

- Be sure to remove this section if you use this `.Rmd` file as a template.
- You may leave the questions in your final document.

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:

```
# Correlation matrix for longley data
round(cor(longley), 2)
```

```
##              GNP.deflator  GNP Unemployed Armed.Forces Population Year Employed
## GNP.deflator         1.00 0.99       0.62         0.46       0.98 0.99     0.97
## GNP                  0.99 1.00       0.60         0.45       0.99 1.00     0.98
## Unemployed           0.62 0.60       1.00        -0.18       0.69 0.67     0.50
## Armed.Forces         0.46 0.45      -0.18         1.00       0.36 0.42     0.46
## Population           0.98 0.99       0.69         0.36       1.00 0.99     0.96
## Year                 0.99 1.00       0.67         0.42       0.99 1.00     0.97
## Employed             0.97 0.98       0.50         0.46       0.96 0.97     1.00
```

The largest correlation between any pair of predictors is 1.00 for predictors Year and GNP.

Table 1: VIF for Model Predictors

Predictor	VIF
GNP.deflator	135.532
GNP	1788.514
Unemployed	33.619
Armed.Forces	3.589
Population	399.151
Year	758.981

(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?

Solution:

```
library(tidyverse)
library(kableExtra)

# Model for longley data
mod = lm(Employed ~ ., data = longley)

# VIF
vif = car::vif(mod)

# VIF table
vif = as_tibble(vif, rownames = NA) %>%
  rownames_to_column('Predictor') %>%
  rename(VIF = value)

# Output table
vif %>%
  kbl(caption = 'VIF for Model Predictors') %>%
  kable_styling()
```

GNP has the largest Variance Inflation Factor (1788.514). Several of the VIFs suggest collinearity among the predictors - especially for GNP, Year, Population and GNP.deflator.

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

Solution:

```
pop_mod = lm(Population ~ GNP.deflator + GNP + Unemployed + Armed.Forces + Year, data = longley)
pop_r2 = scales::percent(summary(pop_mod)$r.squared, 0.01)
```

99.75% of the observed variation in Population is explained by a linear relationship with the other predictors.

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

Solution:

Table 2: VIF for Model Predictors

Predictor	VIF
Unemployed	3.318
Armed.Forces	2.223
Year	3.891

```
pcc = cor(resid(mod), resid(pop_mod))
```

The partial correlation coefficient for `Population` and `Employed` is -2.5399×10^{-17}.

(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?

Solution:

```
# Model from part b
#summary(mod)

# New model fitted with significant predictors from initial model
mod2 = lm(Employed ~ Unemployed + Armed.Forces + Year, data = longley)

# VIF
vif2 = car::vif(mod2)

# VIF table
vif2 = as_tibble(vif2, rownames = NA) %>%
  rownames_to_column('Predictor') %>%
  rename(VIF = value)

max_vif = max(vif2$VIF)

# Output table
vif2 %>%
  kbl(caption = 'VIF for Model Predictors') %>%
  kable_styling()
```

`Year` has the largest Variance Inflation Factor with a value of 3.8909. All of the 3 predictors have relatively low VIF (under 5) and do not suggest multicollinearity in the model.

(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)**

Solution:

```
anova_test = anova(mod2, mod)
f_stat = anova_test[2,5]
pval = anova_test[2,6]
```

$H_0 : Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$

$F - Statistic$: 1.7465

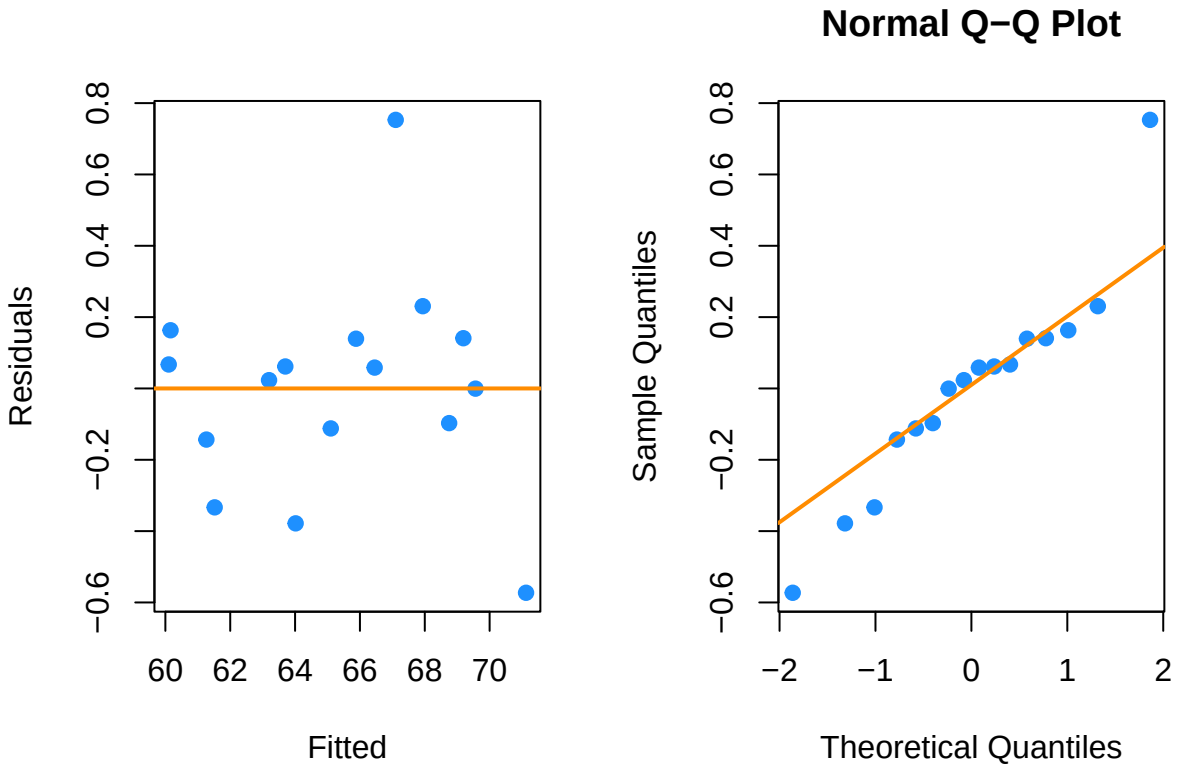
$F - Statistic$ has a χ^2 distribution under the null hypothesis.

$p - value$: 0.227

Decision: Fail to Reject

I would prefer the smaller model **(e)** since the model is smaller and is not statistically different from model **(b)**.

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



The assumption of constant variance and normality appear to be violated. The fitted vs. residuals plot shows larger variance for larger fitted values and the Normal QQ plot has tails on both the left and right side that deviate from normality.

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.

```
#?Credit
```

(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.

```
library(leaps)
library(tidyverse)
#pairs(Credit)



mod_a = lm(Balance ~ ., data = Credit)
vif = car::vif(mod_a)
#summary(mod_a)

all_mod = summary(regsubsets(Balance ~ ., data = Credit))

all_mod$which
```

```
##   (Intercept) Income Limit Rating Cards   Age Education GenderFemale StudentYes
## 1        TRUE  FALSE FALSE   TRUE FALSE FALSE     FALSE        FALSE      FALSE
## 2        TRUE   TRUE FALSE   TRUE FALSE FALSE     FALSE        FALSE      FALSE
## 3        TRUE   TRUE FALSE   TRUE FALSE FALSE     FALSE        FALSE       TRUE
## 4        TRUE   TRUE  TRUE  FALSE  TRUE FALSE     FALSE        FALSE       TRUE
## 5        TRUE   TRUE  TRUE   TRUE  TRUE FALSE     FALSE        FALSE       TRUE
## 6        TRUE   TRUE  TRUE   TRUE  TRUE  TRUE     FALSE        FALSE       TRUE
## 7        TRUE   TRUE  TRUE   TRUE  TRUE  TRUE     FALSE         TRUE       TRUE
## 8        TRUE   TRUE  TRUE   TRUE  TRUE  TRUE     FALSE         TRUE       TRUE
##   MarriedYes EthnicityAsian EthnicityCaucasian
## 1      FALSE          FALSE              FALSE
## 2      FALSE          FALSE              FALSE
## 3      FALSE          FALSE              FALSE
## 4      FALSE          FALSE              FALSE
## 5      FALSE          FALSE              FALSE
## 6      FALSE          FALSE              FALSE
## 7      FALSE          FALSE              FALSE
## 8      FALSE           TRUE              FALSE
```

```
all_mod$adjr2
```

```
## [1] 0.7452 0.8745 0.9495 0.9531 0.9536 0.9540 0.9540 0.9540
```

```
# model 6 has largest adjusted r-squared

# Try AIC backward selection
mod_aic = lm(Balance ~ ., data = Credit)
aic_step = step(mod_aic, direction = 'backward', criteria = 'AIC', trace = FALSE)
#aic_step

mod_3 = lm(Balance  ~  log(Income)*poly(Limit, 2) + Age + Cards + Cards*Age, data = Credit)
mod_a = mod_3
#summary(mod_3)

# check diagnostics
# par(mfrow = c(1, 2))
# plot_fitted_resid(mod_a)
# plot_qq(mod_a)
```

```
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] 145.3
```

```
get_adj_r2(mod_a)
```

```
## [1] 0.9015
```

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

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

```
get_num_params(mod_a)
```

```
## [1] 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.

Solution:

```
# One of my trials from part (a) that did not meet all of the requirements
mod_b = lm(Balance  ~  Income + poly(Rating, 2) + Student + Student*Income, 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_b)
```

```
## [1] 94.32
```

```
get_adj_r2(mod_b)
```

```
## [1] 0.9595
```

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

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

```
get_num_params(mod_b)
```

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

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.

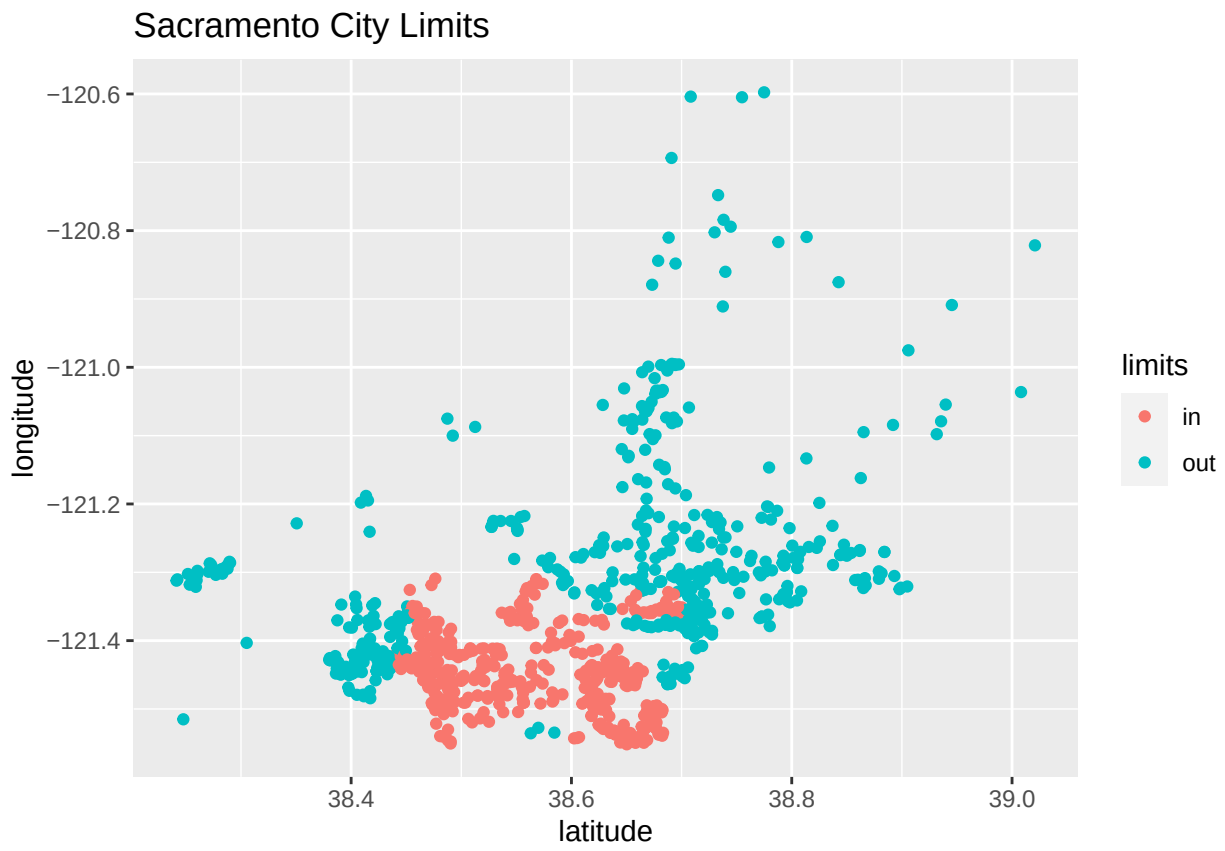
```
library(caret)
library(ggplot2)
data(Sacramento)
sac_data = Sacramento
sac_data$limits = factor(ifelse(sac_data$city == "SACRAMENTO", "in", "out"))
sac_data = subset(sac_data, select = -c(city, zip))
```

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.

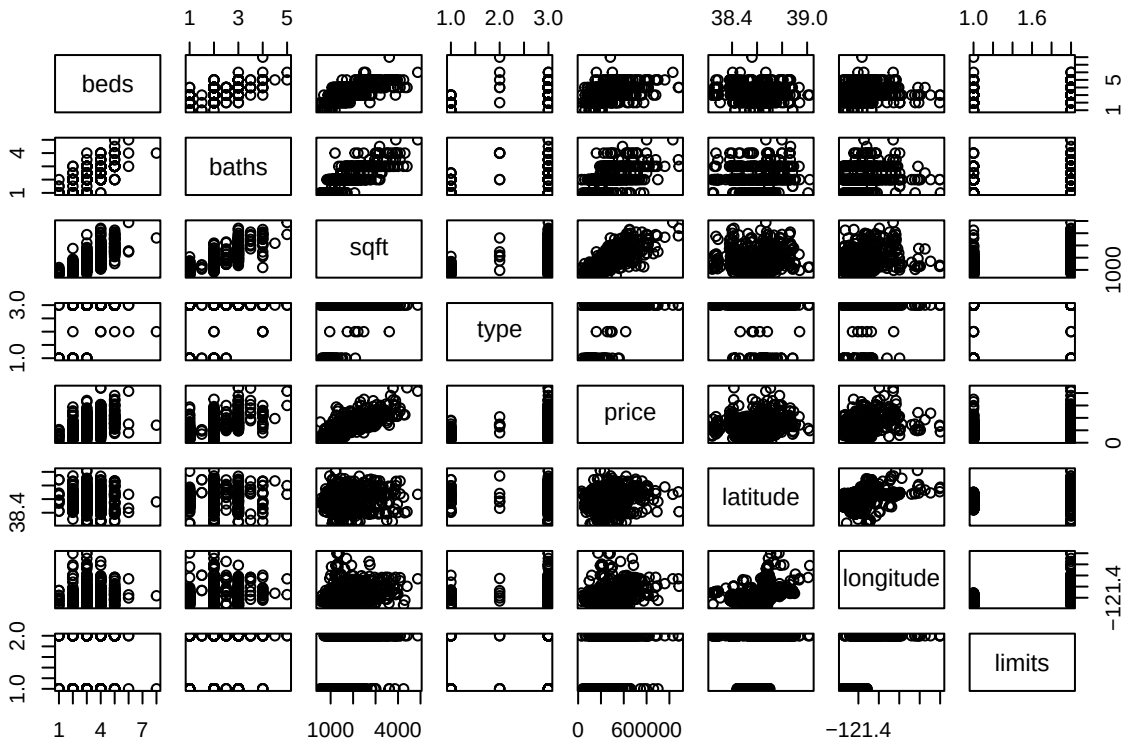
```
set.seed(420)
sac_trn_idx  = sample(nrow(sac_data), size = trunc(0.80 * nrow(sac_data)))
sac_trn_data = sac_data[sac_trn_idx, ]
sac_tst_data = sac_data[-sac_trn_idx, ]
```

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.

Solution:

```
# Look at the data
pairs(sac_trn_data)
```




```
# Backward AIC for getting a starting place for variable selection
#tp_aic = lm(price ~ (.)^2, data = sac_trn_data)
#step(tp_aic, direction = 'backward', criteria = 'AIC')

train_price = lm(price ~ beds +  sqft + type + latitude + longitude + limits + beds:sqft + beds:longitu
                 data = sac_trn_data)

#summary(train_price)
get_loocv_rmse(train_price)
```

```
## [1] 75701
```

(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 $y = x$.

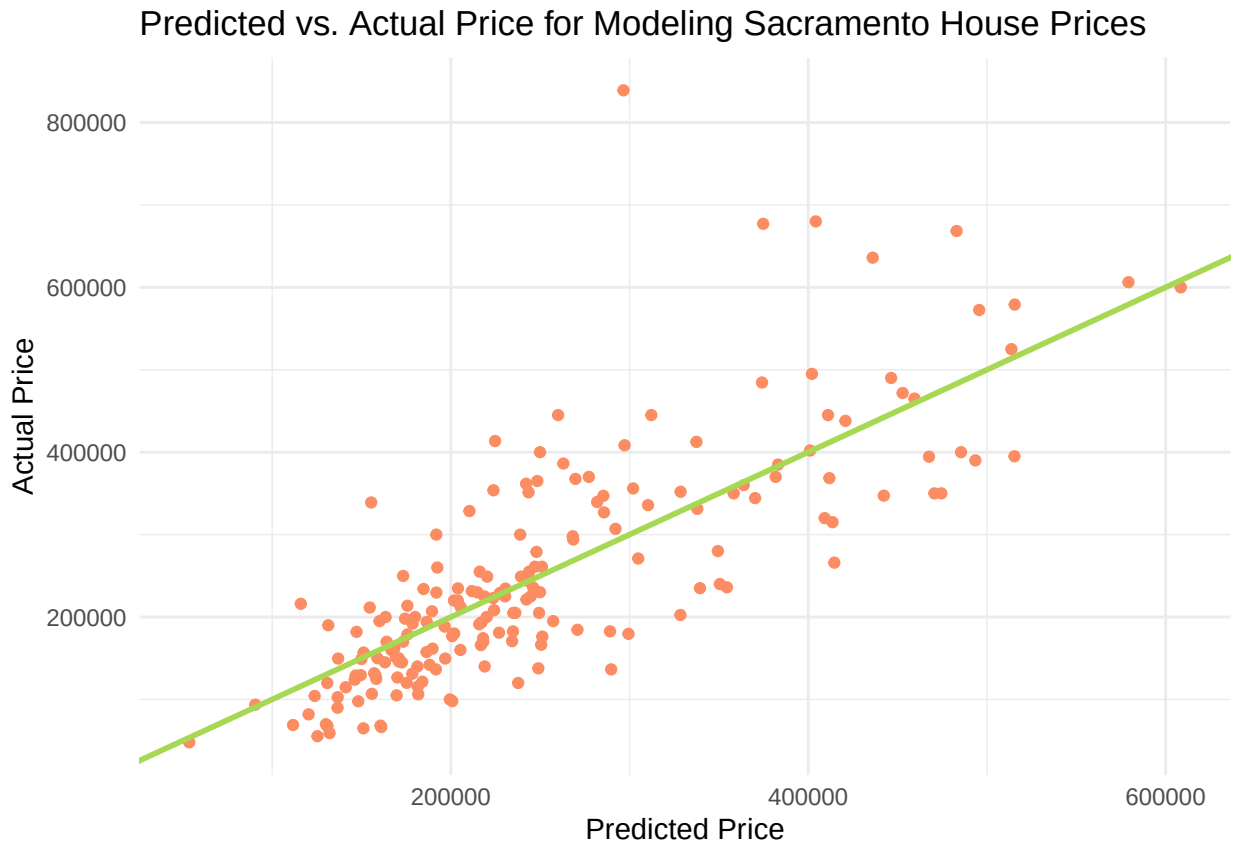
Based on all of this information, argue whether or not this model is useful.

Solution:

```
library(tidyverse)

# Make predictions from testing data
price_predict = predict(train_price, newdata = sac_tst_data)

# Bind predictions to test data frame and calculate percent error
df_plot = sac_tst_data %>%
  bind_cols('predicted_price' = price_predict) %>%
  mutate(abs_error = abs(predicted_price - price)) %>%
  mutate(pct_error = abs_error / predicted_price)

# Average percent error
avg_pct_error = scales::percent(mean(df_plot$pct_error), 0.01)
```

The average percent error of my model is 24.19%.

```
# Predicted vs actual price plot
p = ggplot(df_plot, aes(x = predicted_price, y = price)) +
  geom_point(color = '#fc8d62') +
  geom_abline(slope = 1, intercept = 0, color = '#a6d854', size = 1) +
  theme_minimal() +
  labs(x = 'Predicted Price',
       y = 'Actual Price',
       title = 'Predicted vs. Actual Price for Modeling Sacramento House Prices')

p
```



I believe this model is useful for predicting Sacramento house prices. The distribution of the error terms is symmetric about the line $y = x$ and displays an approximately constant variance for changes in predicted price. There does not appear to be a skewness above or below the line $y = x$ of the error terms but there does appear to be a few outliers in the error terms. The average percent error of 24.19% does seem reasonable in terms of the range of the error of the predicted price.

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.

solution:

```
set.seed(19790221)
library(kableExtra)

# Loop parameters
iter = 300


# Instantiate vectors for holding results
fn_aic = rep(0,300)
fn_bic = rep(0,300)
fp_aic = rep(0,300)
```

Table 3: Comparison of AIC/BIC Error in Selecting Variables

Error Type	AIC	BIC
False Negative	0.00%	0.00%
False Positive	19.07%	19.07%

```
fp_bic = rep(0,300)

for(i in 1:iter){

  # Simulate data
  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)
  )

  # fit model
  mod = lm(y ~ ., data = sim_data_1)


  # AIC and BIC selection
  aic = step(mod, direction = 'backward', criteria = 'AIC', trace = FALSE)
  bic = step(mod, direction = 'backward', criteria = 'BIC', trace = FALSE)

  # false negatives
  fn_aic[i] = sum(!(signif %in% names(coef(aic)))) / 5
  fn_bic[i] = sum(!(signif %in% names(coef(bic)))) / 5

  # false positives
  fp_aic[i] = sum(names(coef(aic)) %in% not_sig) / 5
  fp_bic[i] = sum(names(coef(bic)) %in% not_sig) / 5

}

# Table for output
results = tibble('Error Type' = c('False Negative', 'False Positive'),
                 'AIC' = c(scales::percent(mean(fn_aic), 0.01), scales::percent(mean(fp_aic), 0.01)),
                 'BIC' = c(scales::percent(mean(fn_bic), 0.01), scales::percent(mean(fp_bic), 0.01)))

# Display Table output
results %>%
  kbl(caption = 'Comparison of AIC/BIC Error in Selecting Variables') %>%
  kable_styling()
```

The rates of false positives and false negatives for AIC selection and BIC selection are the same. It is interesting that both methods produce 0.00% false negatives for this data and the rate of false positives is the same for both methods.

(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(19790221)
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)
)
```

Solution:

```
# Loop parameters
iter = 300


# Instantiate vectors for holding results
fn_aic2 = rep(0,300)
fn_bic2 = rep(0,300)
fp_aic2 = rep(0,300)
fp_bic2 = rep(0,300)

for(i in 1:iter){

  # Simulate data
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)
)

  # fit model
  mod = lm(y ~ ., data = sim_data_2)

  # AIC and BIC selection
  aic = step(mod, direction = 'backward', criteria = 'AIC', trace = FALSE)
  bic = step(mod, direction = 'backward', criteria = 'BIC', trace = FALSE)
```

Table 4: Comparison of AIC/BIC Error in Selecting Variables

Error Type	AIC	BIC
False Negative	16.07%	16.07%
False Positive	29.00%	29.00%

```
  # false negatives
  fn_aic2[i] = sum(!(signif %in% names(coef(aic)))) / 5
  fn_bic2[i] = sum(!(signif %in% names(coef(bic)))) / 5

  # false positives
  fp_aic2[i] = sum(names(coef(aic)) %in% not_sig) / 5
  fp_bic2[i] = sum(names(coef(bic)) %in% not_sig) / 5

}

# Table for output
results2 = tibble('Error Type' = c('False Negative', 'False Positive'),
                  'AIC' = c(scales::percent(mean(fn_aic2), 0.01), scales::percent(mean(fp_aic2), 0.01)),
                  'BIC' = c(scales::percent(mean(fn_bic2), 0.01), scales::percent(mean(fp_bic2), 0.01)))

# Display Table output
results2 %>%
  kbl(caption = 'Comparison of AIC/BIC Error in Selecting Variables') %>%
  kable_styling()
```

Again, the rates of false negative and false positive are the same for both methods (AIC/BIC). In part (b), the modeling produced false negatives where in part (a) the modeling did not produce false negatives. Additionally the rate of false positives was higher in part (b) than in part (a). A reason for the difference in performance could be due to the fact that variables `x_8, x_9 and x_10` are collinear to `x_1 and x_2` since their values are based on the values of `x_1 and x_2`. This collinearity could cause `x_8, x_9 or x_10` to be selected by the algorithms instead of `x_1 and x_2` and could explain the increase for both false positives and false negatives. Part (a) did not have this issue since all of the variables in part (a) were independent of each other and did not have collinearity among the predictors.