

Homework 4

Instructions

Due: 5:00pm on Wednesday, October 6th

1. Add your name between the quotation marks on the author line in the YAML above.
2. Compose your answer to each problem between the bars of red stars.
3. Commit your changes frequently.
4. Be sure to knit your .Rmd to a .pdf file.
5. Push both the .pdf and the .Rmd file to your repo on the class organization before the deadline.

Theory

Problem 1

Based on ISLR Exercise 5.2

We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap from a set of n observations. Let j be an arbitrary integer between 1 and n .

- a. What is the probability that the first bootstrap observation is *not* the j th observation from the original sample? Justify your answer.
- b. What is the probability that the second bootstrap observation is *not* the j th observation from the original sample?
- c. Argue that the probability that the j th observation is *not* in the bootstrap sample is $(1 - 1/n)^n$.
- d. Approximate the probability that the j th observation is *not* in the sample in terms of the constant e .
- e. For each of $n = 5, 10, 100$, what is the probability that the j th observation *is* in the bootstrap sample?
- f. Create a plot that displays, for each integer n from 1 to 100, the probability that the j th observation is in the bootstrap sample. Comment on the results of your plot.
- g. Use the `sample` function to create a bootstrap sample of the numbers 1 through 100. Then use either `dplyr` or base R code to assess whether the bootstrap sample contains the number 1. Repeat a total of 10000 times and compute the proportion of times the number 1 appears in your bootstrap samples. Compare to the results of the previous parts.

-
- a. There are n possibly observations that could be the first bootstrap observation, and all but 1 are not the j th observation, so the probability that the 1st bootstrap observation is not the j th observation is

$$P(\text{1st bootstrap obs. not } j\text{th obs.}) = \frac{n-1}{n} = 1 - \frac{1}{n}$$

- b. By similar reasoning to part (a), the probability that the second bootstrap observation is not the j th observation is also

$$P(\text{2nd bootstrap obs. not } j\text{th obs.}) = \frac{n-1}{n} = 1 - \frac{1}{n}$$

- c. Since observations in the bootstrap sample are chosen independently, and each of the n bootstrap observation has probability $1 - \frac{1}{n}$ of not being the j th observation, then the probability the j th bootstrap observation is not in the bootstrap sample is

$$P(\text{jth obs. not in bootstrap}) = \left(1 - \frac{1}{n}\right)^n$$

- d. Using the compound interest formula for e , we note that if n is large,

$$P(\text{jth obs. not in bootstrap}) = \left(1 - \frac{1}{n}\right)^n \approx e^{-1}$$

- e. Note that the probability that the j th observation *is* in the bootstrap sample is 1 minus the probability that the j th observation is not in the bootstrap:

$$P(\text{jth obs. is in bootstrap}) = 1 - P(\text{jth obs. not in bootstrap}) = 1 - \left(1 - \frac{1}{n}\right)^n \approx 1 - e^{-1}$$

For $n = 5, 100, 10000$, the probabilities that the j th observation is in the bootstrap sample are

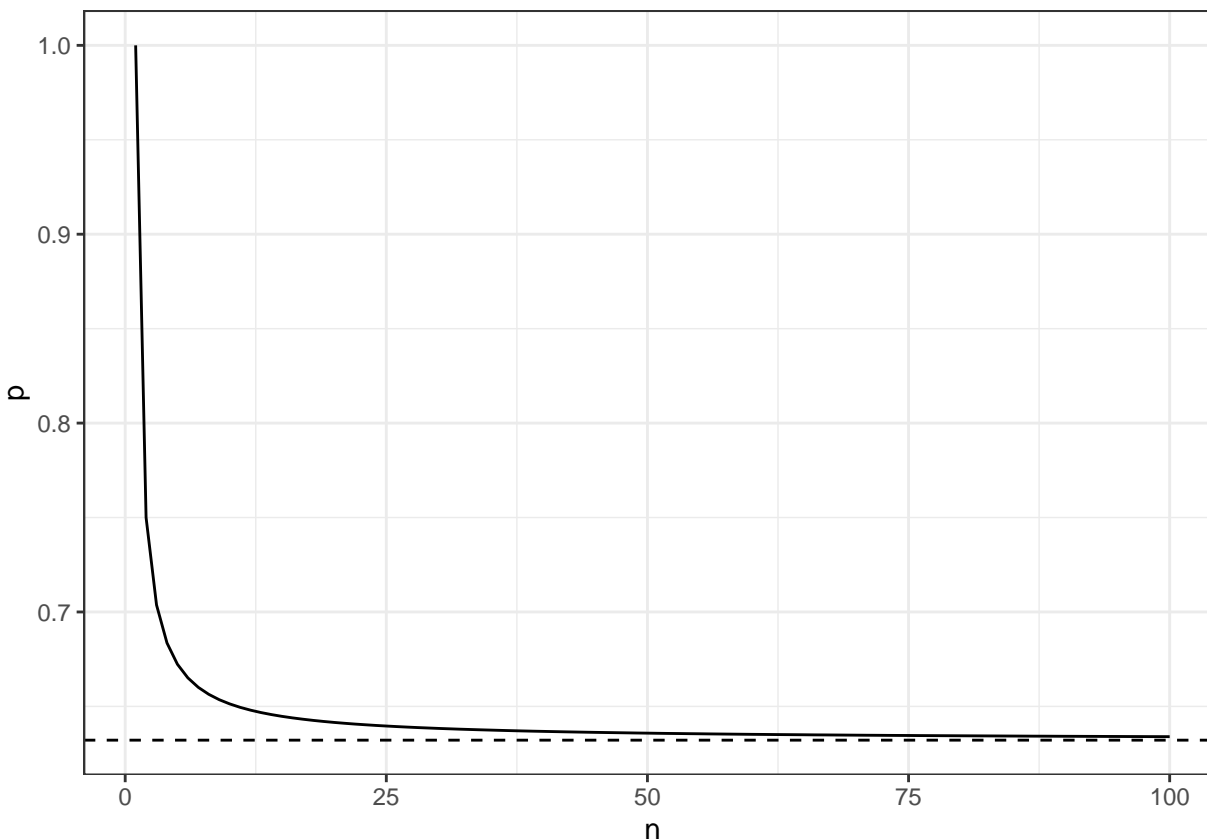
```
n<-c(5,100,10000)
p<-1-(1-1/n)^n
data.frame(n,p)
```

```
##      n      p
## 1     5 0.6723200
## 2    100 0.6339677
## 3 10000 0.6321390
```

- f. The requested plot appears as a horizontal line of approximate height $y = 1 - e^{-1}$.

```
n<-seq(1:100)
p<-1-(1-1/n)^n
d<-data.frame(n,p)
```

```
ggplot(d, aes( x = n, y = p))+geom_line()+geom_hline(yintercept = 1-exp(-1), linetype = "dashed")+theme.
```



g. The proportion of times 1 appears in the bootstrap sample is approximately $1 - e^{-1} \approx 0.6321$

```
set.seed(271)
trials<-10000

results<-c()
for (i in 1:trials){
  results[i]<-1 %in% sample(1:100, size = 100, replace = T)
}

mean(results)

## [1] 0.6361
```

Problem 2

For each of the three scenarios listed below, determine which of the following techniques would be most appropriate to implement in order to achieve the desired result. Briefly justify your answer.

Techniques

- (i) Use a single randomly selected validation set to estimate test MSE.
- (ii) Use summary statistics like R^2 and RSE from the linear model summary table to assess model accuracy.
- (iii) Use k -fold cross validation to estimate test MSE.
- (iv) Using bootstrapping to estimate a statistic's bias and variance.
- (v) None of these methods are appropriate.

Scenarios

- a. A statistics professor wants to create a model exploring the relationship between midterm exam scores and final exam scores for a statistics class. The professor has data for 100 students over the past year, and is interested in finding the best of several polynomial models to predict final score as a function of midterm score.
- b. Researchers are interested in building a model to predict house prices in the Woodstock neighborhood as a function of the house's square footage. Currently, they have data for 10 houses and want to assess whether the non-zero correlation they observed between price and size is likely to just have occurred by random chance.
- c. A data set contains observations on 50,000 samples of red and white wines from the north of Portugal. Researchers are interested in building a model to predict wine quality (on scale 1 - 10) based on physicochemical data for each sample. The research want to compare the performance of two complicated models, each of which takes significant time to code and compute.

-
- a. Since the professor wants to assess model performance, he should use one of methods (i), (ii), or (iii). However, method (ii) will favor models that better predict the training data, and so likely result in overfitting. And method (i) may be prone to error, given variability in the validation error due to small sample size (it also means less data is available to fit the model). Method (iii) is most appropriate here, since polynomial models are relatively quick to fit.
 - b. The researchers needs a way to assess variability in estimates, which can often be done using bootstrapping. However, with very few observations, bootstrapping will not provide a reliable estimate, so should not be used. None of the listed methods are appropriate.
 - c. As in part (a), one of methods (i), (ii), or (iii) should be used. However, since the data set is large, method (i) is less prone to variability just due to training / test split. Additionally, method (iii) may be infeasible, given how long it takes to create each model. Method (i) is most appropriate.
-

Applied

Problem 3

Based on ISLR Exercise 5.8

We will now perform cross-validation on a simulated data set.

- a. Use the following code chunk to generate a simulated data set. Then write out the explicit equation for the model used to generate the data.

```
set.seed(1010) #Change this to your favorite number
x <- rnorm(100, 0, 1) ## Generates 100 variables from N(0,1)
e <- rnorm(100,0, 1) ## Generates 100 errors from N(0,1)
y <- x- 2*x^2 + e ##Specifies Y as a function of X plus errors

sim_data<-data.frame(x,y) ## Creates a data frame of X and Y
```

- b. Create a scatterplot of X against Y. Describe the relationship observed. Calculate mean and standard deviation for each of X and Y, as well as the correlation between X and Y.
- c. Set a seed and compute LOOCV errors from fitting each of the following 4 models using least squares:
- d. $Y = \beta_0 + \beta_1 X + \epsilon$
- ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

- iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
- iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$
- v. Repeat part c using a different seed and report your results. How do they compare to your answer from part c? Explain why this occurred.
- e. Set a seed and compute 5-fold CV from fitting each of models in part c.
- f. Set a different seed from the previous part and again compute 5-fold CV from fitting each of the models in part c. How does your answer compare to part e. Explain why this occurred.
- g. Which of the models in c. had the smallest LOOCV? Explain why this makes sense.
- h. Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in c. using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

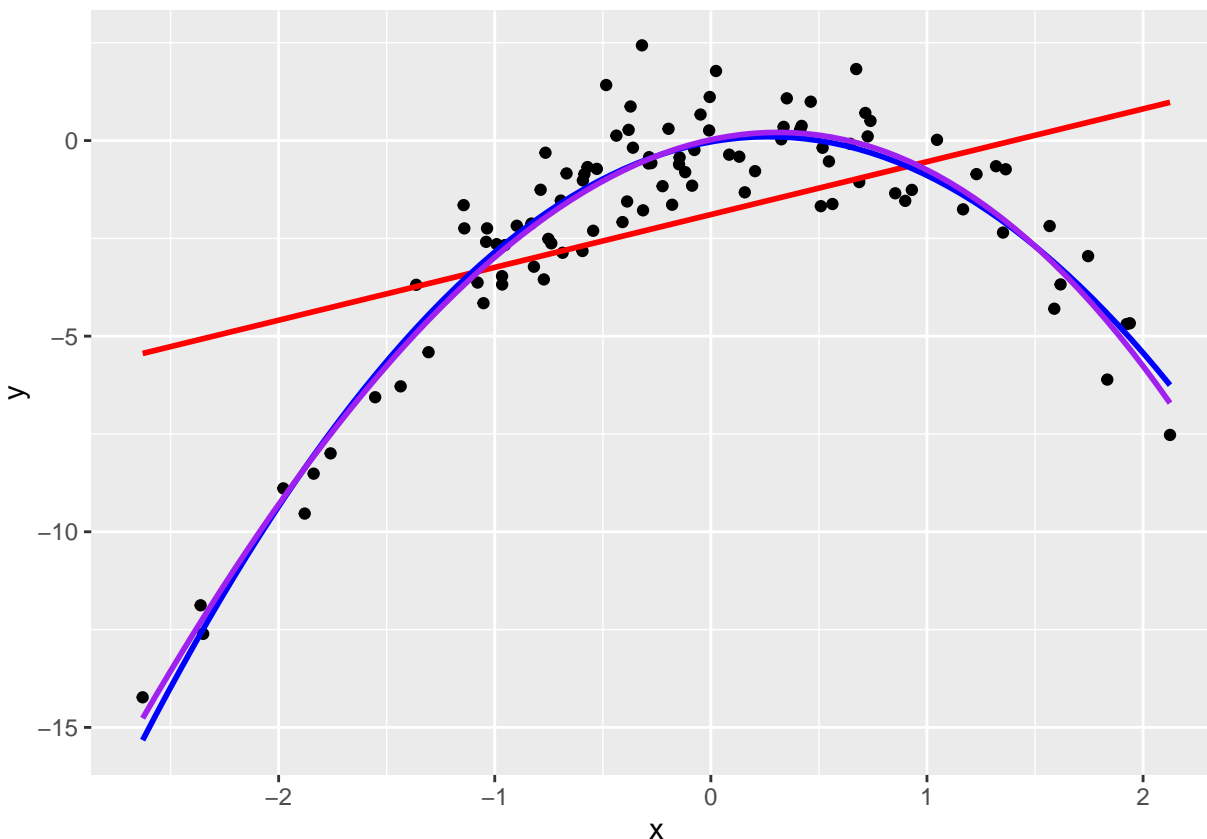
-
- a. The formula specified by the code chunk is $y = x - 2x^2 + \epsilon$

```
set.seed(1010) #Change this to your favorite number
x <- rnorm(100, 0, 1) ## Generates 100 variables from N(0,1)
e <- rnorm(100,0, 1) ## Generates 100 errors from N(0,1)
y <- x- 2*x^2 + e ##Specifies Y as a function of X plus errors

sim_data<-data.frame(x,y) ## Creates a data frame of X and Y
```

- b. Below is a scatterplot of X vs. Y . The line, quadratic, and cubic of best fit are also included, for reference later.

```
sim_data %>% ggplot( aes(x = x , y = y))+
  geom_point()+
  geom_smooth(method = "lm", se = F, color = "red")+
  geom_smooth(method = "lm", formula = y ~ poly(x,2), se = F, color = "blue")+
  geom_smooth(method = "lm", formula = y ~ poly(x,3), se = F, color = "purple")
```



Based on the scatterplot, Y appears to be a quadratic function of X .

```
sim_data %>% summarize(mean_x = mean(x), mean_y = mean(y), sd_x = sd(x), sd_y = sd(y), R = cor(x,y))
```

```
##      mean_x    mean_y    sd_x    sd_y      R
## 1 -0.14124 -2.082307 1.014721 3.034804 0.4515138
```

While the true relationship between X and Y is not linear, the correlation coefficient still indicates a moderately strong positive linear relationship.

- c. We use the `loo_cv` function from `rsample` to create folds (note that `vfold_cv` from the same package could also be used, with `v = n`).

```
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.0.5
```

```
set.seed(1010)
loocv_split <- loo_cv(sim_data)
```

We then create an R function to fit each polynomial model and calculate MSE. For efficiency, we'll create one function that depends on degree, and specialize for each.

```
poly_mse <- function(split, deg){
  training_data <- analysis(split)
  model <- lm(y ~ poly(x, degree = deg, raw = T), data = training_data)
  test <- assessment(split)
  preds <- predict(model, test)
  mse <- mean((test$y - preds)^2)
  mse
}
```

```

}

get_lin_mse <- function(split){poly_mse(split, 1)}
get_quad_mse <- function(split){poly_mse(split, 2)}
get_cub_mse <- function(split){poly_mse(split, 3)}
get_quart_mse <- function(split){poly_mse(split, 4)}

```

Now, we use `map_dbl` from `purrr` to apply this function to each split.

```

loocv_split$lin_mse <- map_dbl(loocv_split$splits, get_lin_mse)
loocv_split$quad_mse <- map_dbl(loocv_split$splits, get_quad_mse)
loocv_split$cub_mse <- map_dbl(loocv_split$splits, get_cub_mse)
loocv_split$quart_mse <- map_dbl(loocv_split$splits, get_quart_mse)

```

And finally, we compute the average MSE for each model type:

```

loocv_split %>% summarize(cv_lin = mean(lin_mse),
                        cv_quad = mean(quad_mse),
                        cv_cub = mean(cub_mse),
                        cv_quart = mean(quart_mse))

```

```

## # A tibble: 1 x 4
##   cv_lin cv_quad cv_cub cv_quart
##   <dbl>  <dbl>  <dbl>   <dbl>
## 1    7.90    1.02    1.01    1.04

```

- d. Setting a different seed and repeating the calculation gives exactly the same result. This is because randomization only determines the order which observations are left out from the training set. But the total CV error is the average across all iterations, which doesn't depend on order observations are selected.

```

set.seed(1)
loocv_split <- loo_cv(sim_data)

```

```

loocv_split$lin_mse <- map_dbl(loocv_split$splits, get_lin_mse)
loocv_split$quad_mse <- map_dbl(loocv_split$splits, get_quad_mse)
loocv_split$cub_mse <- map_dbl(loocv_split$splits, get_cub_mse)
loocv_split$quart_mse <- map_dbl(loocv_split$splits, get_quart_mse)

```

```

loocv_split %>% summarize(cv_lin = mean(lin_mse),
                        cv_quad = mean(quad_mse),
                        cv_cub = mean(cub_mse),
                        cv_quart = mean(quart_mse))

```

```

## # A tibble: 1 x 4
##   cv_lin cv_quad cv_cub cv_quart
##   <dbl>  <dbl>  <dbl>   <dbl>
## 1    7.90    1.02    1.01    1.04

```

- e. We can reuse much of the code from the previous parts. All we need to change is the split data frame:

```

set.seed(111)
poly_cv <- vfold_cv(sim_data, v = 5)

poly_cv$lin_mse <- map_dbl(poly_cv$splits, get_lin_mse)
poly_cv$quad_mse <- map_dbl(poly_cv$splits, get_quad_mse)
poly_cv$cub_mse <- map_dbl(poly_cv$splits, get_cub_mse)
poly_cv$quart_mse <- map_dbl(poly_cv$splits, get_quart_mse)

```

```
poly_cv %>% summarize(cv_lin = mean(lin_mse),
                      cv_quad = mean(quad_mse),
                      cv_cub = mean(cub_mse),
                      cv_quart = mean(quart_mse))
```

```
## # A tibble: 1 x 4
##   cv_lin cv_quad cv_cub cv_quart
##   <dbl>  <dbl> <dbl>   <dbl>
## 1   7.58    1.02  1.00    1.02
```

f. Now, we do observe different cross-validation error rates, since randomization determines which observations are in each of the 5 folds, and we do not average across all possible folds.

```
set.seed(101010)
poly_cv <- vfold_cv(sim_data, v = 5)

poly_cv$lin_mse <- map_dbl(poly_cv$splits, get_lin_mse)
poly_cv$quad_mse <- map_dbl(poly_cv$splits, get_quad_mse)
poly_cv$cub_mse <- map_dbl(poly_cv$splits, get_cub_mse)
poly_cv$quart_mse <- map_dbl(poly_cv$splits, get_quart_mse)

poly_cv %>% summarize(cv_lin = mean(lin_mse),
                      cv_quad = mean(quad_mse),
                      cv_cub = mean(cub_mse),
                      cv_quart = mean(quart_mse))
```

```
## # A tibble: 1 x 4
##   cv_lin cv_quad cv_cub cv_quart
##   <dbl>  <dbl> <dbl>   <dbl>
## 1   8.16    1.01  1.00    1.01
```

g. The quadratic model appears to have the least LOOCV and 5-fold CV error, which is not surprising, given that the underlying relationship between X and Y is actually quadratic.

However, it is worth noting that for different seed values for the simulated data (for example `set.seed(101010)`), the cubic model may actually have the best CV fit. One explanation for this is that the simulated data also contains random noise, as well as randomized X values. Due to chance, these two sources of randomness can produce a sample which looks more like a cubic relationship (especially if the formula for that cubic is very, very closed to a quadratic). In this case, it would not be surprising to find the cubic model actually has the best fit.

h. We see that the linear and quadratic terms are significant at the 0.001 level in all models they are present. The cubic and quartic terms are not significant in any models they are present in.

However, as in the previous part, it's possible for either the cubic or the quartic term to be significant (depending on the seed value).

```
model <- list()
for (i in 1:4){
  model[[i]] <- lm(y ~ poly(x, degree = i, raw = T), data = sim_data)
}

summary(model[[1]])
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree = i, raw = T), data = sim_data)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.7885 -0.6176  0.7698  1.6961  4.7555
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -1.8916     0.2748  -6.883 5.59e-10 ***
## poly(x, degree = i, raw = T)  1.3504     0.2696   5.009 2.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.722 on 98 degrees of freedom
## Multiple R-squared:  0.2039, Adjusted R-squared:  0.1957
## F-statistic: 25.09 on 1 and 98 DF,  p-value: 2.417e-06
summary(model[[2]])

##
## Call:
## lm(formula = y ~ poly(x, degree = i, raw = T), data = sim_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.68833 -0.82615 -0.02206  0.72824  2.96541
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.03226     0.12464  -0.259   0.796
## poly(x, degree = i, raw = T)1  0.98123     0.09966   9.846 2.88e-16 ***
## poly(x, degree = i, raw = T)2 -1.83916     0.07294 -25.216 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9953 on 97 degrees of freedom
## Multiple R-squared:  0.8946, Adjusted R-squared:  0.8924
## F-statistic: 411.7 on 2 and 97 DF,  p-value: < 2.2e-16
summary(model[[3]])

##
## Call:
## lm(formula = y ~ poly(x, degree = i, raw = T), data = sim_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.80671 -0.81652 -0.05082  0.65679  2.98143
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.02209     0.13046   0.169   0.866
## poly(x, degree = i, raw = T)1  1.19399     0.18605   6.417 5.24e-09 ***
## poly(x, degree = i, raw = T)2 -1.88871     0.08135 -23.217 < 2e-16 ***
## poly(x, degree = i, raw = T)3 -0.07739     0.05724  -1.352   0.180
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.991 on 96 degrees of freedom
## Multiple R-squared:  0.8966, Adjusted R-squared:  0.8934
## F-statistic: 277.4 on 3 and 96 DF,  p-value: < 2.2e-16

summary(model[[4]])

##
## Call:
## lm(formula = y ~ poly(x, degree = i, raw = T), data = sim_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.8048 -0.8141 -0.0493  0.6686  2.9718
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.031080   0.149315   0.208    0.836
## poly(x, degree = i, raw = T)1  1.183524   0.204678   5.782 9.37e-08 ***
## poly(x, degree = i, raw = T)2 -1.914650   0.221718  -8.636 1.36e-13 ***
## poly(x, degree = i, raw = T)3 -0.071928   0.072044  -0.998    0.321
## poly(x, degree = i, raw = T)4  0.006173   0.049034   0.126    0.900
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9962 on 95 degrees of freedom
## Multiple R-squared:  0.8966, Adjusted R-squared:  0.8923
## F-statistic:  206 on 4 and 95 DF,  p-value: < 2.2e-16
```

Problem 4

In this problem and the next, we look at the relationship between US stock prices, the earnings of the corporations, and the returns on investment in stocks, with returns counting both changes in stock price and dividends paid to stock holders. A corporation's **earnings** in a given year is its income minus its expenses. The return on an investment over a year is the fractional change in its value, $(v_{t+1} - v_t)/v_t$, and the average rate of return over k years is

$$[(v_{t+k} - v_t)/v_t]^{1/k}.$$

Read this data from the csv in the accompanying data folder:

```
stocks <- read.csv("data/stocks.csv")
```

The dataset contains the following variables:

- **Date**, with fractions of a year indicating months
- **Price** of an index of US stocks (inflation-adjusted)
- **Earnings** per share (also inflation-adjusted);
- **Earnings_10MA_back**, a ten-year moving average of earnings, looking backwards from the current date;
- **Return_cumul**, cumulative return of investing in the stock index, from the beginning;
- **Return_10_fwd**, the average rate of return over the next 10 years from the current date.

“Returns” will refer to **Return_10_fwd** throughout.

Inventing a variable

- Add a new column, `MAPE` to the data frame, which is the ratio of `Price` to `Earnings_10MA_back`. `MAPE` stands for the *monetary-adjusted price-earnings* ratio, and represents the number of years it would take to recoup the cost of a share, assuming the earnings stayed constant at their current level.

Bring up the summary statistics for the new column using the `summary()` command. Why are there exactly 120 NAs? For ease of computing for the rest of the lab, you should remove all rows with any missing data.

- Build a simple linear model to predict `Returns` as a function of `MAPE`. What is the slope of the model and its standard error? Is it statistically significant?
- What is the MSE of this model under 5-fold CV?

Inverting a variable

- Build a simple linear model to predict `Returns` as a function of `1/MAPE`. What is the slope of the model and its standard error? Is it statistically significant?
- What is the CV MSE of this model? How does it compare to the previous one?

A simple model

A simple-minded model says that the expected returns over the next ten years should be exactly equal to `1/MAPE` (i.e earnings / price)

- Find the *training* MSE for this model.
- Explain why the training MSE is equivalent to the estimate of the test MSE that we would get through five-fold CV.

-
- We note that the variable `Earnings_10MA_back` also has exactly NA's for the first 120 months This is not surprising, since it is calculated as a 10 year moving average looking back, and there is not 10 years of data prior to those first 120 months (i.e the first 10 years).

```
stocks <- read.csv("data/stocks.csv")

stocks<-stocks %>% mutate(MAPE = Price/Earnings_10MA_back)

summary(stocks$MAPE)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
##   4.785  11.708  15.947  16.554  19.959  44.196    120

stocks<-stocks %>% drop_na( )
```

- The slope of the linear model is -0.0046 with standard error 0.00017 . This estimate is statistically significant.

```
MAPE_mod<-lm(Return_10_fwd ~ MAPE, data = stocks)
summary(MAPE_mod)

##
## Call:
## lm(formula = Return_10_fwd ~ MAPE, data = stocks)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.116777 -0.029650  0.004347  0.028478  0.093157
```

```
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1383475  0.0029889   46.29  <2e-16 ***
## MAPE         -0.0045885  0.0001727  -26.57  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04321 on 1482 degrees of freedom
## Multiple R-squared:  0.3226, Adjusted R-squared:  0.3221
## F-statistic: 705.8 on 1 and 1482 DF,  p-value: < 2.2e-16
```

c. As in the previous problem, we use `vfold_cv` in `rsample`. The error under 5-fold CV is 0.00187 (or close to this, depending on seed used)

```
set.seed(101010)
stocks_cv <- vfold_cv(stocks, v = 5)

get_stocks_mse <- function(split){
  training_data <- analysis(split)
  model <- lm(Return_10_fwd ~ MAPE, data = training_data)
  test <- assessment(split)
  preds <- predict(model, test)
  mse <- mean((test$Return_10_fwd - preds)^2)
  mse
}

stocks_cv$mse <- map_dbl(stocks_cv$splits, get_stocks_mse)

mean(stocks_cv$mse)

## [1] 0.001870864
```

d. For ease of use later in the problem, we add a new variable `r_MAPE` equal to $1/\text{MAPE}$.

```
stocks <- stocks %>% mutate(r_MAPE = 1/MAPE)
```

The slope of the linear model is 0.996 with standard error 0.037. This estimate is statistically significant.

```
rMAPE_mod <- lm(Return_10_fwd ~ r_MAPE, data = stocks)
summary(rMAPE_mod)
```

```
##
## Call:
## lm(formula = Return_10_fwd ~ r_MAPE, data = stocks)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.106298 -0.030839  0.002955  0.028179  0.103866
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.007659   0.002878  -2.661  0.00788 **
## r_MAPE       0.995904   0.036513  27.275  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04284 on 1482 degrees of freedom
```

```
## Multiple R-squared:  0.3342, Adjusted R-squared:  0.3338
## F-statistic: 743.9 on 1 and 1482 DF,  p-value: < 2.2e-16
```

e. The CV of the rMAPE model is 0.00183, which is very close to the CV of the MAPE model.

```
get_stocks_mse_rmape <- function(split){
  training_data <- analysis(split)
  model <- lm(Return_10_fwd ~ r_MAPE, data = training_data)
  test <- assessment(split)
  preds <- predict(model, test)
  mse <- mean((test$Return_10_fwd - preds)^2)
  mse
}
```

```
set.seed(101010)
stocks_cv <- vfold_cv(stocks, v = 5)

stocks_cv$mse_rmape <- map_dbl(stocks_cv$splits, get_stocks_mse_rmape)

mean(stocks_cv$mse_rmape)
```

```
## [1] 0.001837416
```

f. The training MSE for the simple model is 0.0019

```
stocks %>% mutate(residual_sq = (Return_10_fwd - r_MAPE)^2 ) %>%
  summarize(MSE = mean(residual_sq))
```

```
##           MSE
## 1 0.001896346
```

g. Ordinarily, when we perform 5-fold CV, we fit a total of 5 models (one for each fold that is omitted from the training set), compute the test MSE of these models on the respective fold, and average the results.

However, in this problem, we do not fit a new model for each fold, but rather, we use the simple model each time. When we then average the 5 test MSEs, we are actually just averaging the squared residuals from the same simple model across the entire data set, which is exactly the training MSE.

Problem 5

Is simple sufficient?

The model that we fit in part 4d is very similar to the simple-minded model. Lets compare the similarity in these models. We could go about this in two ways. We could *simulate* from the simple-minded model many times and fit a model of the same form as 4d. to each one to see if our observed slope in 4d. is probable under the simple-minded model. We could also *bootstrap* the data set many times, fitting this model each time, then see where the simple-minded model lays in that distribution. Since we already have practiced with simulation, lets do the bootstrap method.

- Form the bootstrap distribution for the slope of $1/\text{MAPE}$. Plot this distribution with the parameter of interest (the slope corresponding to the simple-minded model) indicated by a vertical line.
- What is the approximate 95% bootstrap confidence interval for the slope? How does this interval compare to the one returned by running `confint()` on your model object from 4d. of Problem 4? Explain any differences you observe.

One big happy plot

- c. Make a scatterplot of the returns against MAPE. Add two curves showing the predictions from the models you fit in 4b. and 4d. Add a line showing the predictions from the simple-minded model.

The big picture

- d. **Cross-validation for model selection:** using CV MSE, which model would you select to make predictions of returns? Looking at the plot in part c., does this seem like a good model? What are its strengths and weaknesses for prediction?
- e. **Bootstrapping for uncertainty estimation:** based on your bootstrapping procedure for the slope of the linear model using 1/MAPE as a predictor, is the simple-minded model a plausible model given our data?

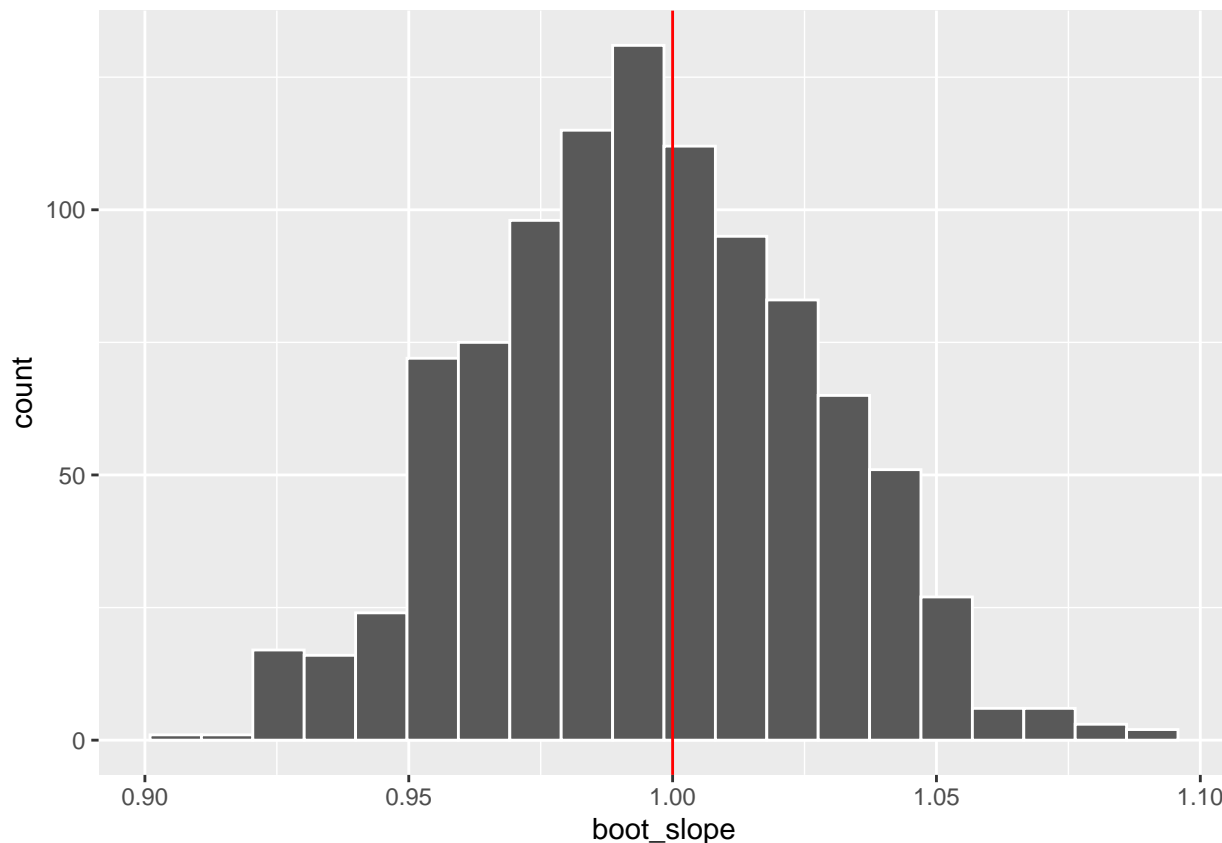
-
- a. Using bootstraps in rsample:

```
set.seed(10102)
stocks_boot <- bootstraps(stocks, times = 1000)

get_slope <- function(split){
  training_data <- analysis(split)
  model <- lm(Return_10_fwd ~ r_MAPE, data = training_data)
  slope <- coef(model)[2]
  slope
}

stocks_boot$boot_slope <- map_dbl(stocks_boot$splits, get_slope)

ggplot(stocks_boot, aes( x = boot_slope))+
  geom_histogram(bins = 20, color = "white")+
  geom_vline(xintercept = 1, color = "red")
```



- b. We compute the 95% confidence interval using quantiles and obtain: (0.935, 1.056). This is similar (but narrower) than the 95% interval obtained from `confint` of (0.924, 1.067). The `confint` function builds intervals assuming that data approximately follows a *t*-distribution. But the bootstrap method builds confidence intervals based on the shape of the bootstrap distribution, which appears to be narrow here than the corresponding *t*-distribution.

```
stocks_boot %>% summarize(q.025 = quantile(boot_slope, 0.025), q.975 = quantile(boot_slope, 0.975))
```

```
## # A tibble: 1 x 2
##   q.025 q.975
##   <dbl> <dbl>
## 1 0.935  1.05
```

```
confint(rMAPE_mod)
```

```
##           2.5 %      97.5 %
## (Intercept) -0.01330433 -0.002013051
## r_MAPE      0.92428102  1.067526198
```

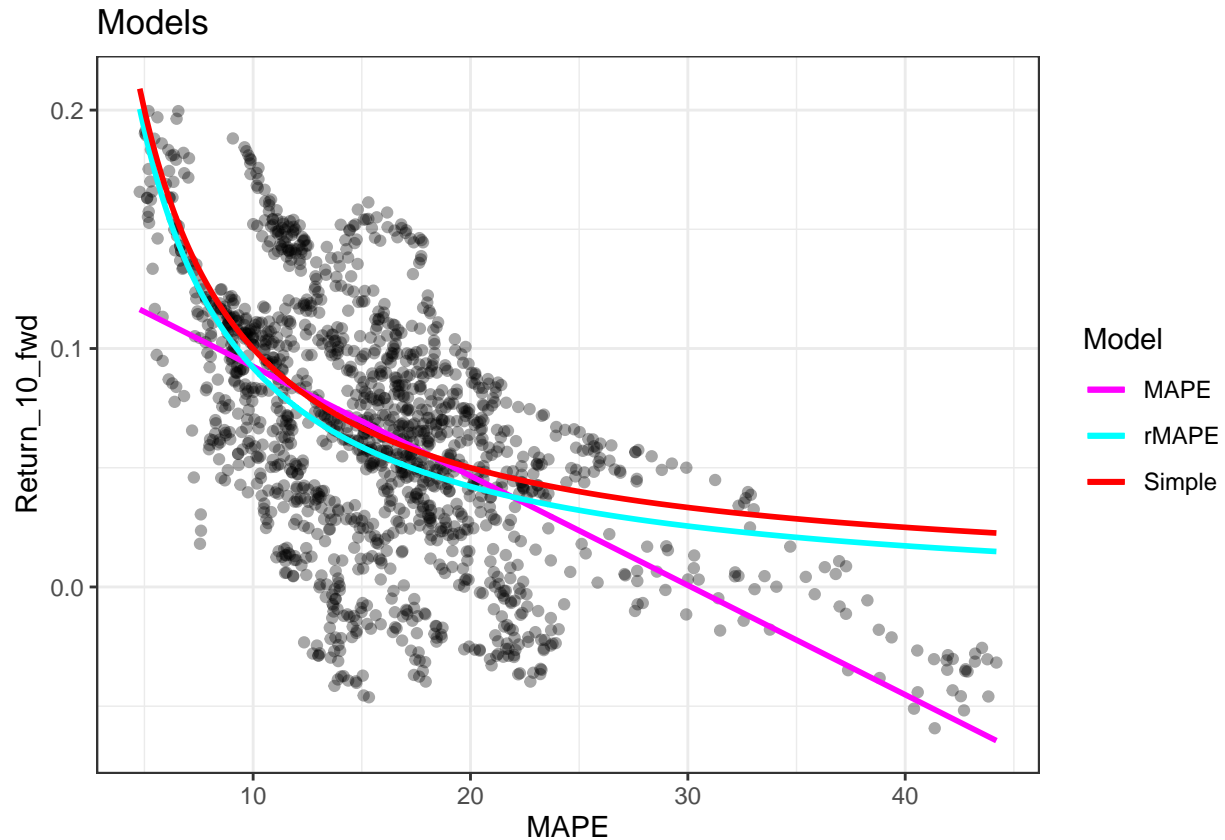
c.

We'll add predictions from both the MAPE and rMAPE models to the `stocks` dataframe.

```
stocks <- stocks %>% mutate(preds_MAPE = predict(MAPE_mod),
                             preds_rMAPE = predict(rMAPE_mod),
                             )
```

```
colors <- c("MAPE" = "magenta", "rMAPE" = "cyan", "Simple" = "red")
ggplot(stocks, aes(x = MAPE)) +
```

```
geom_point(aes(y = Return_10_fwd), alpha = .35)+
geom_line(aes(y = preds_MAPE, color = "MAPE"), size = 1)+
geom_line(aes(y = preds_rMAPE, color = "rMAPE"), size = 1)+
geom_line(aes(y = r_MAPE, color = "Simple"), size = 1)+
theme_bw()+
labs(color = "Model", title = "Models")+
scale_color_manual(values = colors)
```



- d. The model predicting **Returns** as a function of $1/\text{MAPE}$ has the lowest CV MSE of 0.00184 (compared to 0.00187 for MAPE and .0019 for the simple model). This seems like a reasonable model based on the data, capturing the the non-linear trend in **Returns**. However, it does tend to significantly overpredict for values of MAPE larger than about 25.
- e. The bootstrap confidence interval for the slope of $1/\text{MAPE}$ contains the value 1, indicating that the simple model is plausible given the data. Given the philosophy of parsimony, the simple model may be preferable to the $1/\text{MAPE}$ linear regression, since its coefficients are more easily interpreted, and the test MSEs are comparable.