Lab 6: Hands off my stack

Claire Jellison

11/3/2019

go back and fix number 2 in a simple model

```
d <- read.csv("https://bit.ly/36kibHZ")
```

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.

1. Inventing a variable

1) Add a new column, MAPE to the data frame, which is the ratio of Price to Earnings_10MA_back. 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 may want to remove all rows with any missing data.

```
d <- d %>% mutate(MAPE = Price/Earnings_10MA_back)
nrow(d)
```

```
## [1] 1724
```

```
summary(d$MAPE)
```

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

```
df <- na.omit(d)
summary(df$MAPE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.785  11.513  15.464  16.040  18.946  44.196
```

The variable Earnings_10MA_back is a 10 year moving average of earnings so it requires that there be at least 10 years of previous data to generate a value. Thus the the dataset begins in 1871 but the variable can't be computed until 10 years later in 1881. Given that there are 12 observations per year, the data seems to be collected monthly. Therefore there are 10 years without a value and 12 observations per year so a total of 120 missing values.

2) Build a linear model to predict returns using MAPE (and nothing else). What is the coefficient and it's standard error? Is it significant?

```
linear_returns <- lm(Return_10_fwd ~ MAPE, data = df)
summary(linear_returns)
```

```
##
## Call:
## lm(formula = Return_10_fwd ~ MAPE, data = df)
##
## 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
```

The coefficient on MAPE is -0.0045885, meaning that for a unit increase in the ratio of Price to Earnings_10MA_back the average rate of return over the next 10 years decreases by -0.0045885 so the two are inversely correlated, and the standard error is 0.0001727. It is very significant and we can reject the null hypothesis that the 99% confidence level.

3) What is the MSE of this model under five-fold CV? I recommend you go about this by adding a column to your data frame indicating the randomly assigned fold to which every observation belongs. Then use a for-loop to fit and predict across each of the five folds, where you can use the appropriate data by subsetting based on the fold.

```
MSEfcn <- function(model, data){
  n <- nrow(data)
  ys <- data$Return_10_fwd
  y_hats <- predict(model, data)
  residuals <- y_hats - ys
  MSE <- sum(residuals^2)/n
  MSE
}
```

```
k<- 5
MSE_all <- c(0,0,0,0,0)
partition_index5 <- rep(1:k, each=nrow(df)/k) %>%
  sample()

for(i in 1:k){
train<- df[partition_index5!=i,]
test<- df[partition_index5==i,]
model<- lm(Return_10_fwd ~ MAPE, data=train)
MSE_5 <- MSEfcn(model, test)
MSE_all[i] = MSE_5
}
MSE_all
```

```
## [1] 0.002029339 0.001841860 0.001935295 0.001627543 0.001900840
```

```
mean(MSE_all)
```

```
## [1] 0.001866976
```

The MSE of this model based on 5-fold CV is 0.001867706.

Inverting a variable

1)Build a linear model to predict returns using 1/MAPE (and nothing else). What is the coefficient and its standard error? Is it significant?

```
df <- df %>% mutate(invertMAPE = 1/MAPE)
linear_returns2 <- lm(Return_10_fwd ~ invertMAPE, data = df)
summary(linear_returns2)
```

```
##
## Call:
## lm(formula = Return_10_fwd ~ invertMAPE, data = df)
##
## 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 **
## invertMAPE   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
```

Now the coefficient on 1/MAPE is 0.995904 and the standard error is 0.036513. The variable is highly significant.

2)What is the CV MSE of this model? How does it compare to the previous one?

```
k<- 5
MSE_all2 <- c(0,0,0,0,0)
partition_index5 <- rep(1:k, each=nrow(df)/k) %>%
  sample()

for(i in 1:k){
train2<- df[partition_index5!=i,]
test2<- df[partition_index5==i,]
model2<- lm(Return_10_fwd ~ invertMAPE, data=train2)
MSE_5 <- MSEfcn(model2, test2)
MSE_all2[i] = MSE_5
}
MSE_all2
```

```
## [1] 0.001968348 0.001616238 0.002059333 0.001791934 0.001756711
```

```
mean(MSE_all2)
```

```
## [1] 0.001838513
```

The five fold CV MSE is now 0.001840317. The previous CV MSE was 0.001869776. This means that the two CV MSE's are very similar although the newer one is slightly lower this could just be a product of the way that the data was partitioned.

A simple model

A simple-minded model says that the expected returns over the next ten years should be exactly equal to 1/MAPE.

1) Find the training MSE for this model.

```
ntrain <- nrow(df)
ysimple <- df$Return_10_fwd
yhatsimple <- 1/df$MAPE
residualsimple <- yhatsimple - ysimple
MSEsimple <- sum(residualsimple^2)/ntrain
MSEsimple
```

```
## [1] 0.001896346
```

```
summary(yhatsimple)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02263 0.05278 0.06467 0.07270 0.08686 0.20898
```

2) Explain why the training MSE is equivalent to the estimate of the test MSE that we would get through five-fold CV.

The training MSE will be the same as the estimate of the test MSE from five-fold CV because we are getting estimates on an observation by observation basis.

Is simple sufficient?

The model that we fit in no. 2 is very similar to the simple-minded model. Let's 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 no. 2 to each one to see if our observed slope in no. 2 is probable under the simple-minded model. We could also bootstrap the data set many times, fitting model 2 each time, then see where the simple-minded model lays in that distribution. Since we already have practiced with simulation, let's do the bootstrap method.

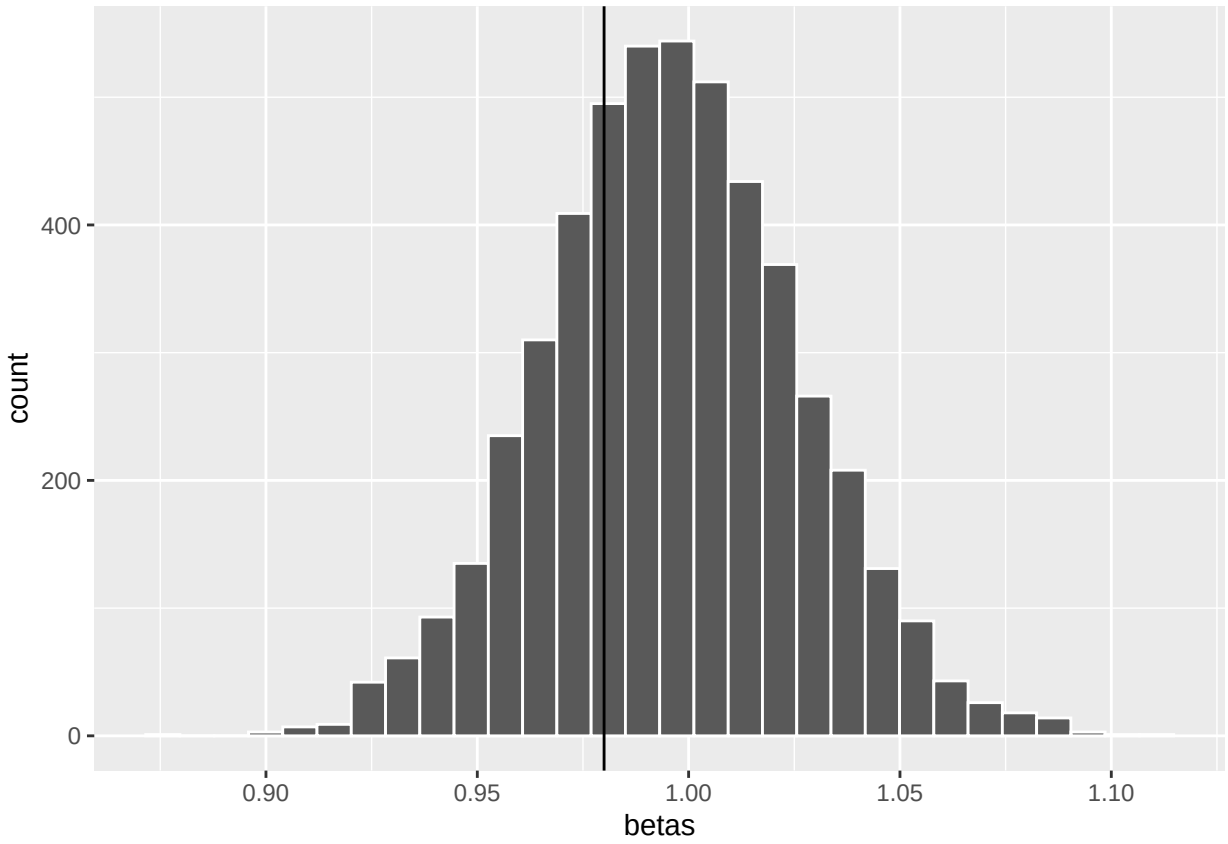
1) Form the bootstrap distribution for the slope of 1/MAPE (the code from class may be helpful). Plot this distribution with the parameter of interest (the slope corresponding to the simple-minded model) indicated by a vertical line.

```
betas <- rep(NA, 5000)

for(i in 1:5000) {
  boot_ind <- sample(1:nrow(df),
                     size = nrow(df),
                     replace = TRUE)
  df_boot <- df[boot_ind, ]
  betas[i] <- coef(lm(Return_10_fwd ~ invertMAPE,
                      data = df_boot))[2]
}
```

```
dfnew <- data.frame(betas)
ggplot(dfnew, aes(betas)) +
  geom_histogram(col = "white") + geom_vline(xintercept = .98)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2) 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 question 2? Please explain any difference you've found.

```
nn <- nrow(df)
meanb <- mean(betas)
sdb <- sd(betas)
confint95lower = meanb - 2*sdb/sqrt(nn)
confint95upper = meanb + 2*sdb/sqrt(nn)
conf95 <- cbind(confint95lower,confint95upper)
conf95
```

```
##      confint95lower confint95upper
## [1,]      0.9944506      0.9975758
```

```
confint(linear_returns2)
```

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

We can see that the confidence interval is pretty narrow for the bootstap. On the other hand, using the confint command on the model from number two yields a much wider confidence interval.

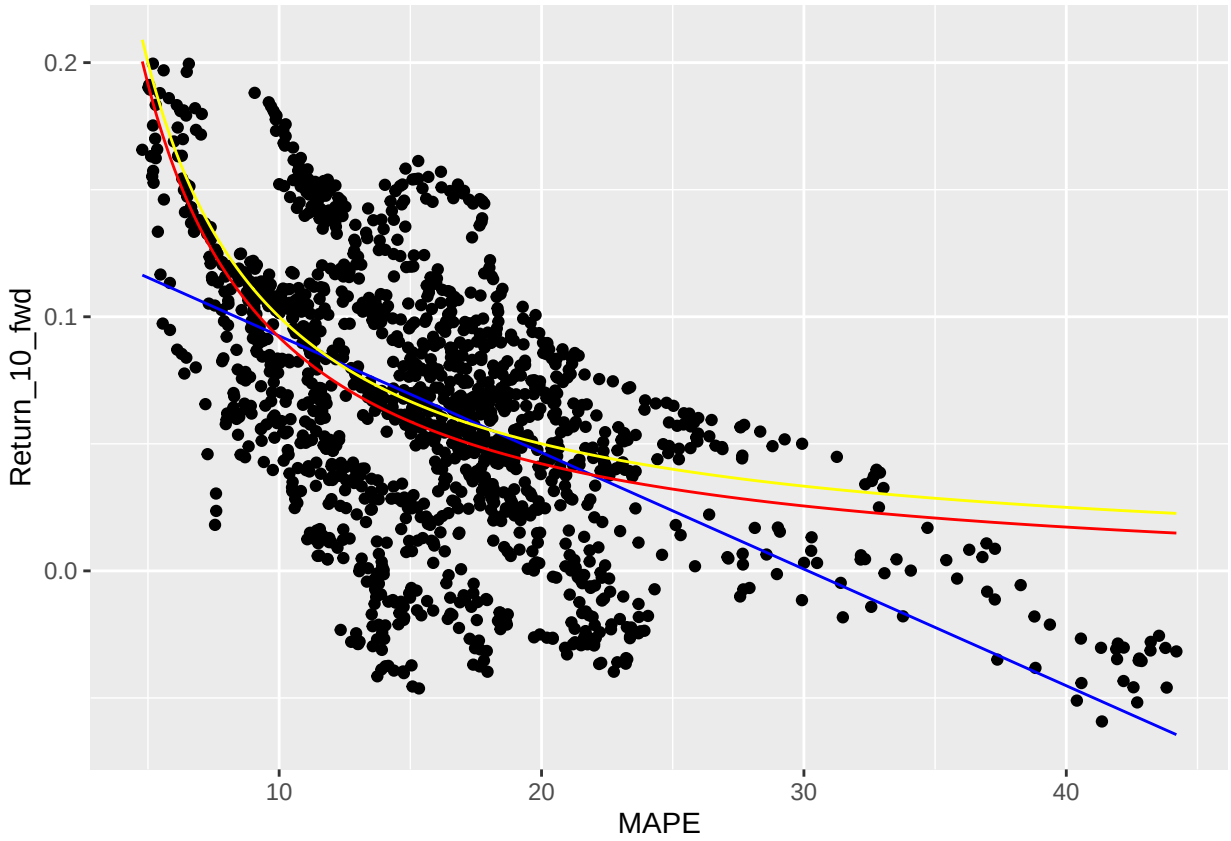
One big happy plot

For this problem, you need to only include one plot and one paragraph of writing. Also, in this problem, take "line" to mean "straight or curved line" as appropriate, and be sure to plot actual lines and not disconnected points.

1) Make a scatterplot of the returns against MAPE.
2) Add two lines showing the predictions from the models you fit in problems 1 and 2.
3) Add a line showing the predictions from the simple-minded model from problem 3.

```
mod1 <- lm(Return_10_fwd ~ MAPE, data = df)
mod2 <- lm(Return_10_fwd ~ invertMAPE, data = df)

ggplot(df, aes(x = MAPE, y = Return_10_fwd)) + geom_point() + geom_line(data = fortify(mod1), aes(x = MA
```



We see here that the first model stands out for being quite different from the other two models. In particular it appears that for low values of MAPE, the simple model and the invertMAPE model look like they preform better while for higher values of MAPE the first model appears to be a better fit.

The big picture

1) Cross-validation for model selection: using CV MSE, which model would you select to make predictions of returns? Looking at the plot in question 5, does this seem like a good model? What are its strengths and weaknesses for prediction?

Using the CV MSE the second model produced the lowest MSE, but all three were fairly close to one another. Above, this model is shown by the red line. It appears to do a good job at the lower range of MAPE but yield overly high estimates at the upper range of the MAPE values. However, the majority of the data is concentrated towards the lower values, so this was probably not a huge disadvantage for the model.

2) 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?

The simple model appears to do quite well and mirrors the second model although producing slightly higher

predictions.

Chapter 5 exercises

4) Suppose that we use some statistical learning method to make a prediction for the response Y for a particular value of the predictor X. Carefully describe how we might estimate the standard deviation of our prediction.

You can preform the bootstrap. For some data with n observations, you can get a new datasets Z_i with n observations that were sampled with replacement from the original data. With the datasets Z_i where $i\epsilon\{0, B\}$ for some large number B you can get B estimates of α_i. You can find the standard errors of the α using the typical standard error equation. This can serve as an estimate for the standard deviation of our prediction.

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

(a) Generate a simulated data set as follows:

```
set.seed(1)
x1=rnorm(100)
y1=x1-2*x1^2+rnorm(100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

The number of observations n is 100. p would be two since there are two coefficients in the equation aside from the intercept.

$$y = x - 2x^2 + \epsilon$$

(b) Create a scatterplot of X against Y . Comment on what you find.

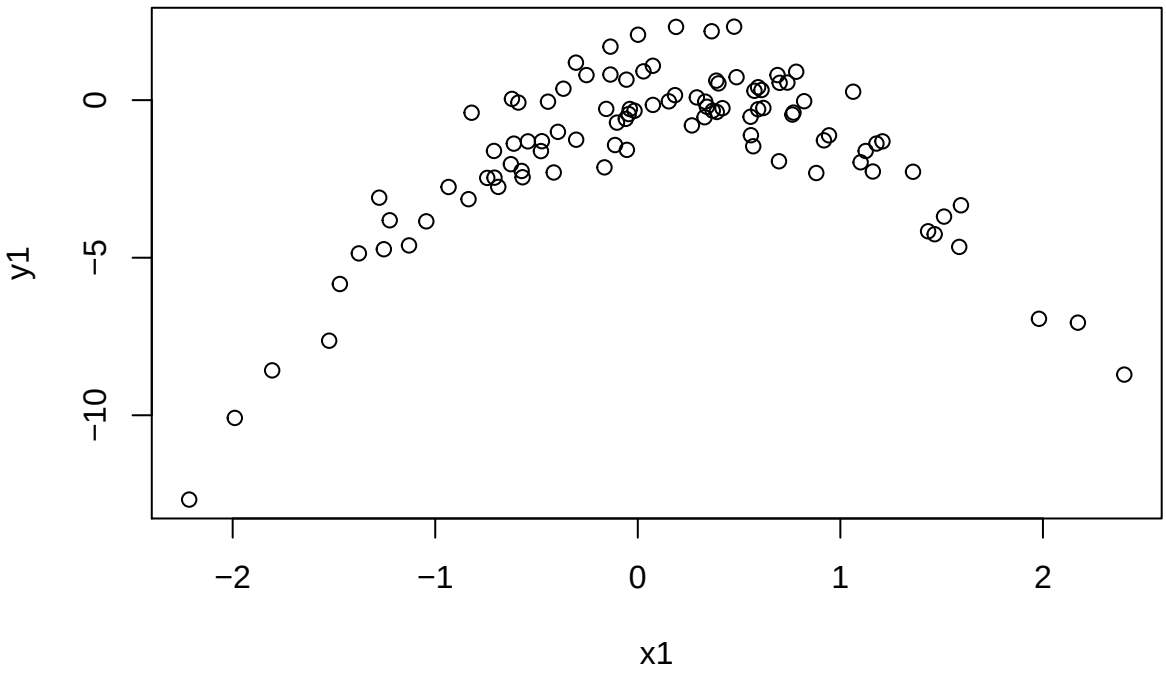
```
summary(x1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.2147 -0.4942  0.1139  0.1089  0.6915  2.4016
```

```
summary(y1)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -12.67519  -2.34584  -0.90394  -1.55002   0.05108   2.33127
```

```
newleave <- cbind(x1,y1)
plot(newleave)
```



The plot looks like basically like a normal distribution curve centered slightly to the left of zero.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

i. $Y = \beta_0 + \beta_1 X + \epsilon$
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. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

```
MSEfcn8 <- function(model, data){
  n <- nrow(data)
  ys <- data$y1
  y_hats <- predict(model, data)
  residuals <- y_hats - ys
  MSE <- sum(residuals^2)
  MSE
}
```

```
set.seed(9)

vecc <- c(1:100)
geneight <- data.frame(x1,y1)

MSEi <- c(rep(0,100))
MSEii <- c(rep(0,100))
MSEiii <- c(rep(0,100))
MSEiv <- c(rep(0,100))


for (i in c(1:100)){
leftout <- geneight[i,]
rest <- geneight[-c(i:i),]
```

```
modeli = lm(y1 ~ x1, data = rest)
modelii = lm(y1 ~ x1 + I(x1**2), data = rest)
modeliii = lm(y1 ~ x1 + I(x1**2) + I(x1**3), data = rest)
modeliv = lm(y1 ~  x1 + I(x1**2) + I(x1**3) + I(x1**4), data = rest)

MSEi[i] =  MSEfcn8(modeli,leftout)
MSEii[i] = MSEfcn8(modelii,leftout)
MSEiii[i] = MSEfcn8(modeliii,leftout)
MSEiv[i] = MSEfcn8(modeliv,leftout)
}

mean(MSEi)
```

```
## [1] 7.288162
```

```
mean(MSEii)
```

```
## [1] 0.9374236
```

```
mean(MSEiii)
```

```
## [1] 0.9566218
```

```
mean(MSEiv)
```

```
## [1] 0.9539049
```

```
#geneight[1,]
#geneight[-c(1:1),]
```

For the first model, 7.288162, for the second, 0.9374236, for the third 0.9566218, and for the fourth 0.9539049. The second model seems preferred.

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
set.seed(11)
vecc <- c(1:100)
geneight <- data.frame(x1,y1)

MSEi <- c(rep(0,100))
MSEii <- c(rep(0,100))
MSEiii <- c(rep(0,100))
MSEiv <- c(rep(0,100))


for (i in c(1:100)){
leftout <- geneight[i,]
rest <- geneight[-c(i:i),]


modeli = lm(y1 ~ x1, data = rest)
modelii = lm(y1 ~ x1 + I(x1**2), data = rest)
modeliii = lm(y1 ~ x1 + I(x1**2) + I(x1**3), data = rest)
modeliv = lm(y1 ~  x1 + I(x1**2) + I(x1**3) + I(x1**4), data = rest)

MSEi[i] =  MSEfcn8(modeli,leftout)
MSEii[i] = MSEfcn8(modelii,leftout)
```

```
MSEiii[i] = MSEfcn8(modeliii,leftout)
MSEiv[i] = MSEfcn8(modeliv,leftout)
}

mean(MSEi)
```

```
## [1] 7.288162
```

```
mean(MSEii)
```

```
## [1] 0.9374236
```

```
mean(MSEiii)
```

```
## [1] 0.9566218
```

```
mean(MSEiv)
```

```
## [1] 0.9539049
```

```
#geneight[1,]
#geneight[-c(1:1),]
```

Yes the results are identical because with the leave one out method, every observation is left out exactly once, so there isn't room for variation and it doesn't depend on the randomness of partitioning like with k-fold.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The second model at the lowest LOOCV error. I expected the first one to be bad since that model is linear and the data generating process used a quadratic term. The form of the second model is closest to the true functional form so I expected that it would preform best. Furthmore, models 3 and 4 would just be overfitting some of the noise so I thought their LOOCV would be slightly lower than for the second model.

(f) 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?

```
x2 = x1**2
x3 = x1**3
x4 = x1**4
modeli = lm(y1 ~ x1, data = geneight)
modelii = lm(y1 ~ x1 + x2, data = geneight)
modeliii = lm(y1 ~ x1 + x2 + x3, data = geneight)
modeliv = lm(y1 ~  x1 + x2 + x3 + x4, data = geneight)
summary(modeli)
```

```
##
## Call:
## lm(formula = y1 ~ x1, data = geneight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5161 -0.6800  0.6812  1.5491  3.8183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.6254     0.2619  -6.205 1.31e-08 ***
## x1            0.6925     0.2909   2.380   0.0192 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.6 on 98 degrees of freedom
## Multiple R-squared:  0.05465,    Adjusted R-squared:  0.045
## F-statistic: 5.665 on 1 and 98 DF,  p-value: 0.01924
```

```
summary(modelii)
```

```
##
## Call:
## lm(formula = y1 ~ x1 + x2, data = geneight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9650 -0.6254 -0.1288  0.5803  2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.05672    0.11766   0.482    0.631
## x1           1.01716    0.10798   9.420  2.4e-15 ***
## x2          -2.11892    0.08477 -24.997  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.958 on 97 degrees of freedom
## Multiple R-squared:  0.873,  Adjusted R-squared:  0.8704
## F-statistic: 333.3 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
summary(modeliii)
```

```
##
## Call:
## lm(formula = y1 ~ x1 + x2 + x3, data = geneight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9765 -0.6302 -0.1227  0.5545  2.2843
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06151    0.11950   0.515    0.608
## x1           0.97528    0.18728   5.208 1.09e-06 ***
## x2          -2.12379    0.08700 -24.411  < 2e-16 ***
## x3           0.01764    0.06429   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9626 on 96 degrees of freedom
## Multiple R-squared:  0.8731, Adjusted R-squared:  0.8691
## F-statistic: 220.1 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
summary(modeliv)
```

```
##
## Call:
```

```
## lm(formula = y1 ~ x1 + x2 + x3 + x4, data = geneight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0550 -0.6212 -0.1567  0.5952  2.2267
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.156703   0.139462   1.124    0.264
## x1           1.030826   0.191337   5.387 5.17e-07 ***
## x2          -2.409898   0.234855 -10.261  < 2e-16 ***
## x3          -0.009133   0.067229  -0.136    0.892
## x4           0.069785   0.053240   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9591 on 95 degrees of freedom
## Multiple R-squared:  0.8753, Adjusted R-squared:  0.8701
## F-statistic: 166.7 on 4 and 95 DF,  p-value: < 2.2e-16
```

Yes, we see that only the first two coefficients (aside from the intercept) are significant indicating the the second model is the best and doesn't include unecesary flexibility with extra parameters.