Stat 897 Fall 2017 Data Analysis Assignment 2

Penn State

Due September 3, 2017

Knitting from the command line and alternative editors

Assuming you have everything installed, you can render this R markdown file from your shell command line with

```
Rscript -e "rmarkdown::render('DAA02.Rmd')"
```

or by hitting the knit button in RStudio. This might be more comfortable if you prefer to edit your R code in emacs (e.g. with ess or polymode) or vim.

1. Boston housing and using bootstrap estimators

Consider the Boston housing data set, from the MASS library.

```
library(MASS)
library(stats)
library(boot)
library (ISLR)
```

Let's specify a random seed for these exercises so that we get the same results each time they are run.

```
set.seed(97)
```

If the procedure uses any randomization, and you use a different seed (or don't specify one), your results and the solution may differ.

a. Based on this data set, provide an estimate for the population mean of medv. Call this estimate $\hat{\mu}$, or `mu.hat` in R.

```
mu.hat = mean(Boston$medv)
mu.hat
```

```
## [1] 22.53281
```

mu.hat = 22.53281

b. Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result. Hint: you can review this in the Stat 800 notes (link).

```
se_mu_hat = sd(Boston$medv) / sqrt(506)
se_mu_hat
```

```
## [1] 0.4088611
```

SE(mu.hat) = 0.4088611

c. Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

The first step is to load the bootstrap library.

```
library(boot)
```

You'll want to define a function for your statistic (mean) and then use the `boot` function, like `mean.boot <- boot(...)`. See this resource (link) for a helpful example using this function.

```
mean.fn=function (data,index){
    return (mean(data[index]))
}
set.seed(3)
mean.boot <- boot(Boston$medv, mean.fn, R=1000)
mean.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = mean.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 0.001749605   0.4058516
```

Roughly speaking, the standard error tells us the average amount that this estimate mu_hat differs from the actual value of mu. Tthis deviation shrinks with n - the more observations we have, the smaller the standard error of mu_hat.

In the above we find that mu_hat=22.53281 and SE(mu_hat) = 0.4058516 This is vert close to our computed value in part b of 0.4088611

d. Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of medv. Compare it to the results obtained using `t.test(Boston$medv)`. Assuming you defined `mean.boot <- boot(...)` appropriately above, what are `mean.boot$t0`, `mean.boot$t`, and the "bias" reported?

Confidence interval from bootstrap: Based on the results above the 95% confidence interval is: 21.7214794 and 23.3441406

```
t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

The results from t.test give: 95 percent confidence interval: 21.72953 23.33608

Both the results are very close to each other.

```
mean.boot$t0
```

```
## [1] 22.53281
```

```
mean(mean.boot$t)
```

```
## [1] 22.53456
```

```
bias = mean(mean.boot$t) - mean.boot$t0
bias
```

```
## [1] 0.001749605
```

mean.boot$t0 = 22.53281 *mean.boot*t - shows 1000 values for all the invocations of mean.fn during the bootstrap mean(mean.boot$t) = 22.53456 bias = 0.001749605

e. Based on this data set, provide an estimate, for the median value of medv in the population.

```
median.hat = median(Boston$medv)
median.hat
```

```
## [1] 21.2
```

median.hat = 21.2

f. We now would like to estimate the standard error of the median. Unfortunately, it is not as simple to produce a formula for computing the standard error of the median (although there are some things we can do with order statistics; check out Stat 553 if you are interested). But we don't need to that. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
median.fn=function (data,index){
    return (median(data[index]))
}
set.seed(3)
median.boot <- boot(Boston$medv, median.fn, R=1000)
median.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = median.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0291   0.3790209
```

We see that the original median is given as 21.2 (as above) and the se(median_hat) = 0.3790209

g. Based on this data set, provide an estimate for Q1 and Q3 of medv in Boston suburbs (and for the three or four of you in the class that live in Boston, have a laugh at how much things have changed since 1978; these values are in thousands of dollars).

```
summary(Boston$medv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   17.02   21.20   22.53   25.00   50.00
```

```
quantile(Boston$medv)
```

```
##     0%    25%    50%    75%   100%
##  5.000 17.025 21.200 25.000 50.000
```

From above we can see Q1 = 17.02 and Q3 = 25

h. Use the bootstrap to estimate the standard errors of these quartiles. Comment on your findings. What do these estimates tell you about the distribution of the variable?

```
q1.fn=function (data,index){
  q = quantile(data[index]);
  return (q[2])
}
set.seed(1)
q1.boot <- boot(Boston$medv, q1.fn, R=1000)
q1.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = q1.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*   17.025 -0.08675   0.5580207
```

```
q3.fn=function (data,index){
  q = quantile(data[index]);
  return (q[4])
}

q3.boot <- boot(Boston$medv, q3.fn, R=1000)
q3.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = q3.fn, R = 1000)
##
##
```

```
## Bootstrap Statistics :
##     original  bias    std. error
## t1*       25  0.4081   0.9083437
```

We find that SE(Q1) = 0.5580207 and SE(Q3) = 0.9083437

Finding: We see that the confidence interval is narrower for Q1 - this may point to higher variance at Q3 levels as compared to Q1 - this may point to skewness and with a right tail.

2. Finding your own example of the bias-variance tradeoff

The bias-variance tradeoff discussed in Section 2 of the book is one of the most important concepts to internalize from this class.

a. Describe a class of models in which flexibility can be changed, leading to a bias-variance tradeoff.

We will use the regressiong with varying flexibility by changing the polynomial degree. We should see that as the complexity is increased (increase in poly degree) we see the testing error comes down initially (because the decrease in bias is > increase in variance)

However as we keep increasing the complexity the decrease in bias is shadowed by a bigger increase in variance and we see that the testing error either flattens or even rises.

b. Describe what overfitting and underfitting mean in this case, and what property of the model you use to measure "flexibility."

As explained above the complexity and/or flexibility is increased by the polynomial degree of the model. We should see that initially as we increase the flexibility we have an overall decrease in the testing error. This is because initially when the model is not flexible at all it has a large bias (but low variance). We can say that the model is underfitting the data.

As we gradually increase flexibility we see that the bias becomes less (but the variance increases). The decrease in bias > increase in variance and we see improvement (lowering) in the testing error.

As we further keep increasing the flexibility we ae now entering the realm of overfitting. At this point the model is learning the patterns in the training data - its bias is not decreasing more than the increase in variance since the model is now very sensitive to the training data it is using.

c. Now create a numerical example of bias-variance tradeoff. Simulate some data and fit at least 10 models of varying flexibility (for example, you could use splines like the book, or regression with more an more interaction and quadratic terms, or a GLM with an increasing number of interaction terms and therefore parameters; anything you are comfortable with). Plot the data you generate and explain the models you fit, showing all R code. For each fitted model, produce both training and test error.

```
set.seed(3)
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
```

```
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                         name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
dim(Auto)
```

```
## [1] 392   9
```

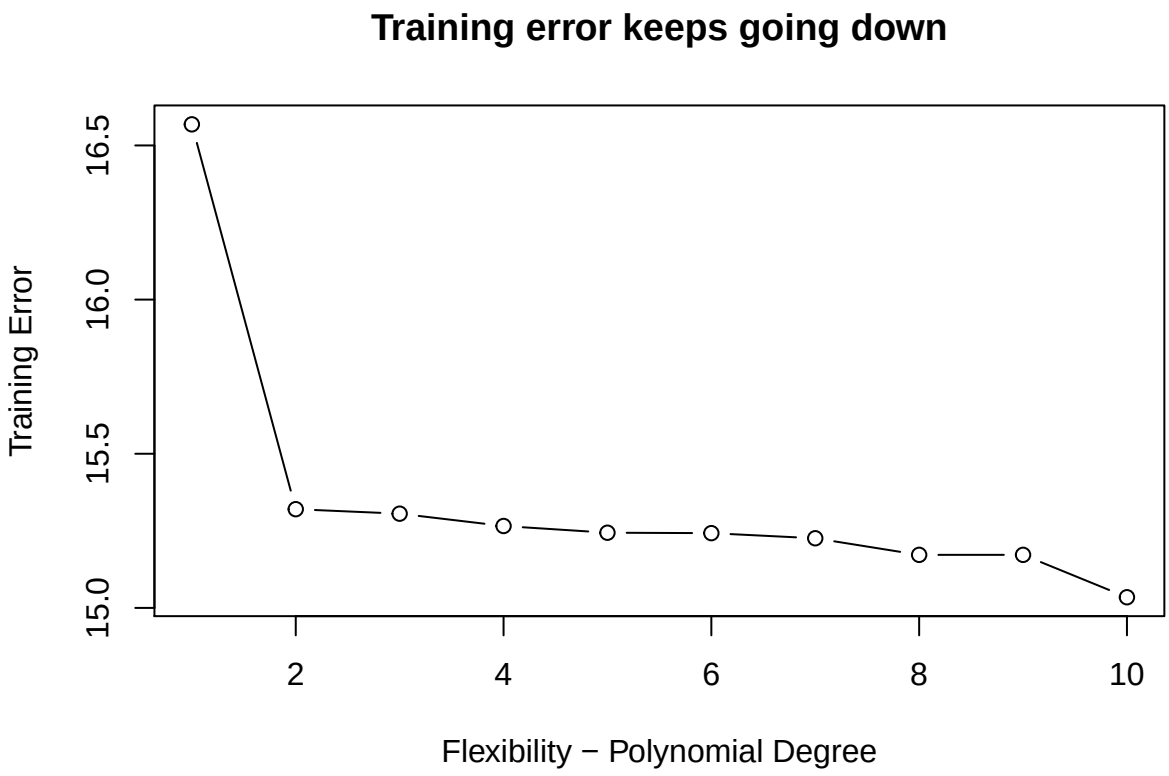
```
train = sample(392, 196)

attach(Auto)
train.error=rep (0,10)
test.error=rep (0,10)

for (i in 1:10){
  lm.fit = lm(mpg ~ poly(displacement + horsepower + weight + acceleration, i), data=Auto, subset=train)
  train.error[i] = mean((mpg - predict (lm.fit, Auto))[train ]^2)
  test.error[i] = mean((mpg - predict (lm.fit, Auto))[-train ]^2)
}
train.error
```

```
##  [1] 16.56837 15.32018 15.30565 15.26564 15.24392 15.24246 15.22586
##  [8] 15.17209 15.17207 15.03460
```

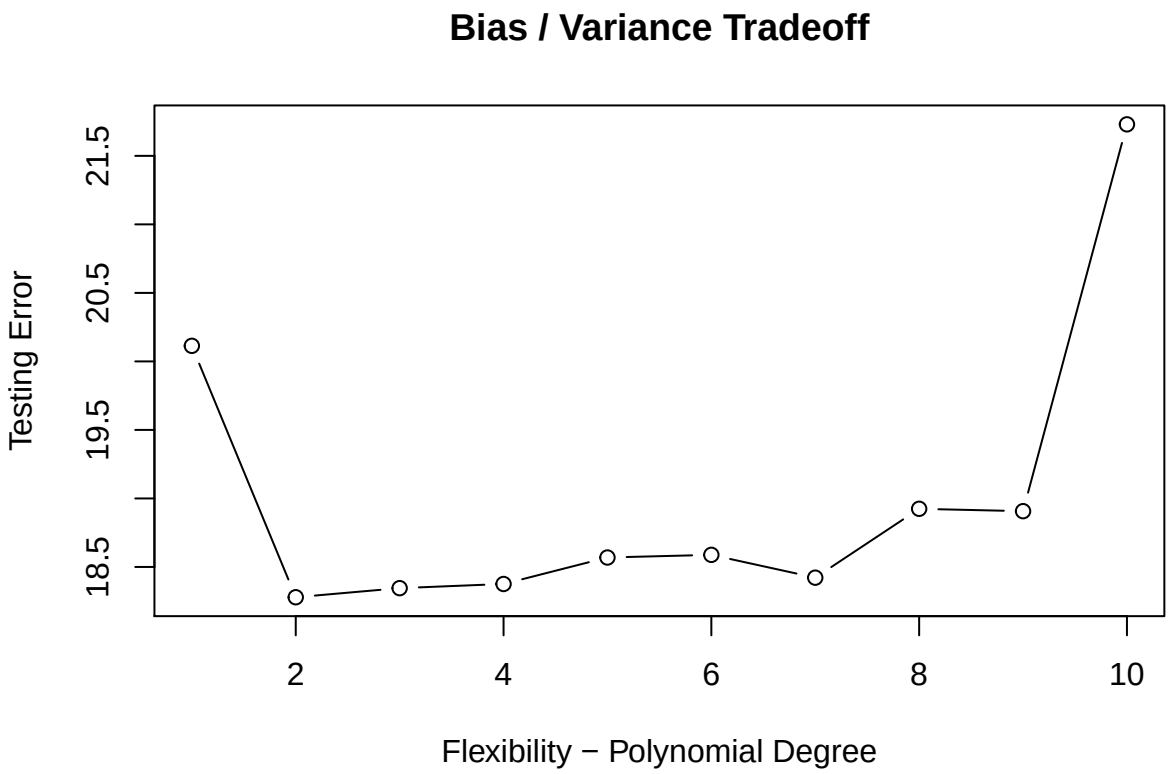
```
test.error
```

```
##  [1] 20.11361 18.27897 18.34534 18.37571 18.56896 18.58830 18.42178
##  [8] 18.92435 18.90701 21.72998
```

```
plot(x = 1:10, y = train.error, type='b',xlab = "Flexibility - Polynomial Degree", ylab = "Training Erro
```

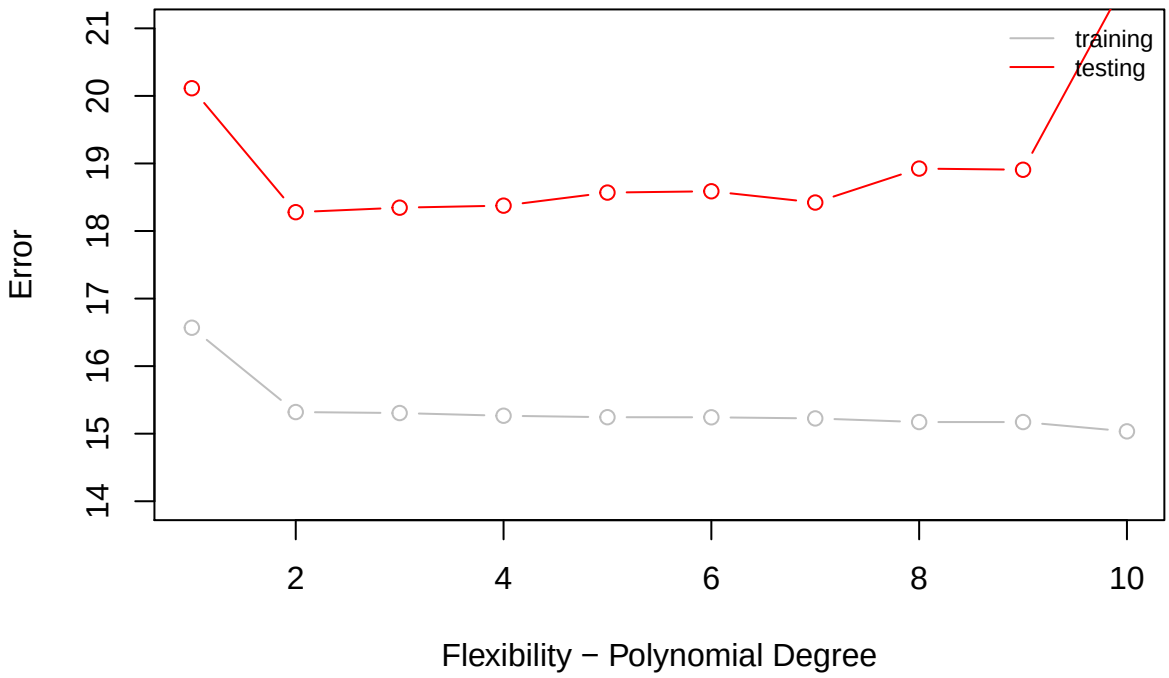



```
plot(x = 1:10, y = test.error, type='b',xlab = "Flexibility - Polynomial Degree", ylab = "Testing Error
```



d. Produce a plot like Figure 2.9(Right) on page 31 summarizing the bias-variance tradeoff for your setting. The plot should clearly show two lines: monotonically decreasing training error (the gray line in the book) and a U-shaped test error curve (the red line in the book).

```
plot(x = 1:10, y = train.error, type='b',xlab = "Flexibility - Polynomial Degree", ylab = "Error", col=
lines(x = 1:10, y = test.error, type='b', col='red')
legend('topright', c('training', 'testing') ,
   lty=1, col=c('grey', 'red'), bty='n', cex=.75)
```



Finally submit BOTH your .rmd file and the resulting .pdf file with Canvas as Data Analysis Assignment 2.