Stat 897 Spring 2017 Data Analysis Assignment 2

Penn State

Due January 22, 2017

Acknowledgement: the solution to qeustion 2 is provided by Tan Huan

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)
```

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
```

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$medv)/sqrt(length(Boston$medv))
se.mu.hat
```

```
## [1] 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)
```

The manual is long but it is easy to use; this will take only about 3 lines of code; define a function for your statistic (mean) and then use the `boot` function, like `mean.boot <- boot(...)`.

Solution

```
mean.boot.fn <- function(data, index){
    mean(data[index])
}
mean.boot <- boot(Boston$medv, mean.boot.fn, 1000)
mean.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = mean.boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 -0.01763419   0.4199739
```

End Solution

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?

Solution You can approximate a 95% confidence interval using the formula $[\hat{\mu}$ -1.96SE($\hat{\mu}$),$\hat{\mu}$ + 1.96SE($\hat{\mu}$)]. Instead of using mu.hat we will use the mean generated by the bootstrap estimate. The following values are reported in the description printed when you type `mean.boot`. The "original" is the mean applied to the whole data set, and equals `mean.boot$t0`. The vector `mean.boot$t` is the mean function applied to all 1000 bootstrap samples. The "bias" is the difference `mean(mean.boot$t) - mean.boot$t0`. The std. error is `sd(mean.boot$t)`.

The lower and upper bounds of the confidence interval are as follows.

```
mean(mean.boot$t) - 1.96*sd(mean.boot$t)
```

```
## [1] 21.69202
```

```
mean(mean.boot$t) + 1.96*sd(mean.boot$t)
```

```
## [1] 23.33832
```

```
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 resulting confidence interval is very similar. It is a bit unncessary to use bootstrap here; instead this is meant to give us confidence in using it later for problems that require either more sophisticated techniques or bootstrap's brute-force approach, like the next one. **End Solution**

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

Solution

```
med <- median(Boston$medv)
med
```

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

End Solution

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.

Solution

```
median.boot.fn <- function(data, index){
    median(data[index])
}
median.boot <- boot(Boston$medv, median.boot.fn, 1000)
median.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = median.boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0268   0.3776498
```

That was easy, or easier than Chapter 6 of the 553 lecture notes. The standard error of mean and median are pretty close; not too suprising if you look at the shape of the dataset with `hist(Boston$medv)` or `plot(density(Boston$medv))`. **End Solution**

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).

Solution

```
Q1 <- quantile(Boston$medv, .25)
Q3 <- quantile(Boston$medv, .75)
Q1
```

```
##    25%
## 17.025
```

```
Q3
```

```
## 75%
##  25
```

End Solution

h. Use the bootstrap to estimate the standard errors of these quartiles. Comment on your findings. What insights, if any, do these estimates throw on the distribution of the variable?

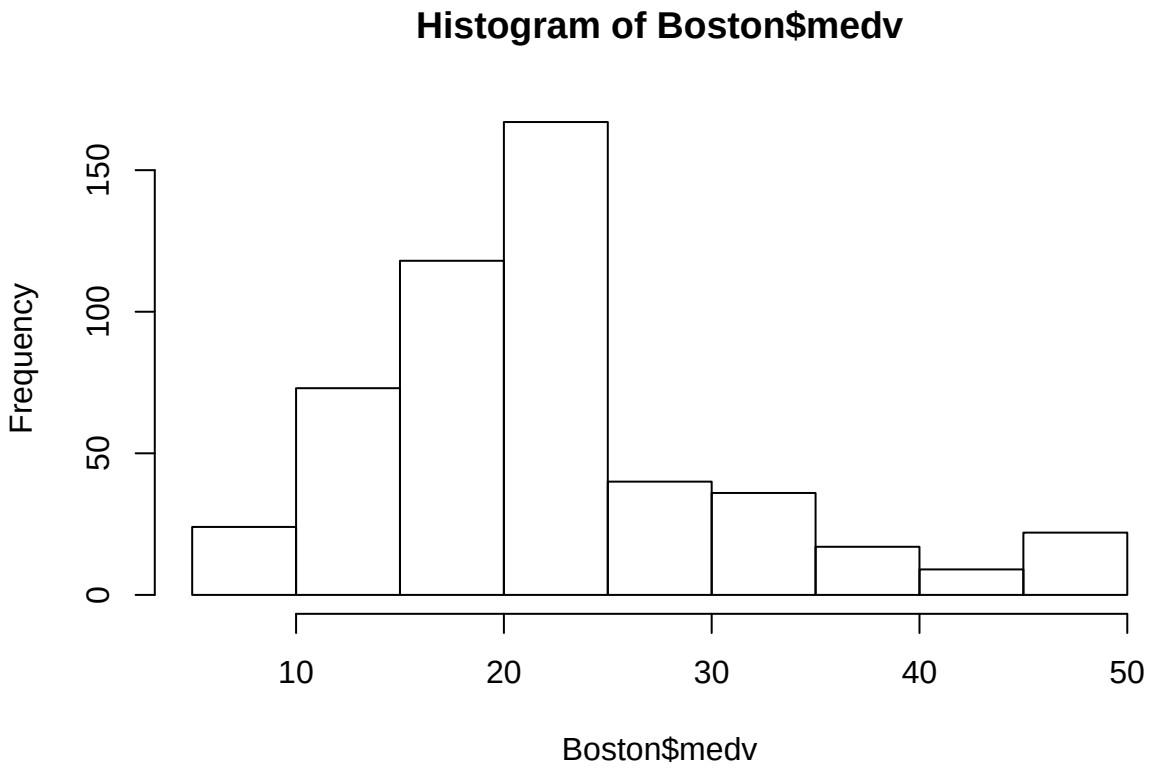
Solution

```
quantile.boot.fn <- function(data, index){
  quantile(data[index], c(.25, .75))
}
quantile.boot <- boot(Boston$medv, quantile.boot.fn, 1000)
quantile.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = quantile.boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   17.025 -0.101175   0.5659324
## t2*   25.000  0.412775   0.8931091
```

We can see Q3 has a larger standard error than Q1, indicating this distribution is slightly right skewed. Check the histogram

```
hist(Boston$medv)
```



End Solution

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. Come up with a simple example (in words) of the bias-variance tradeoff: some real or simulated data, and a class of models in which flexibility can be changed.

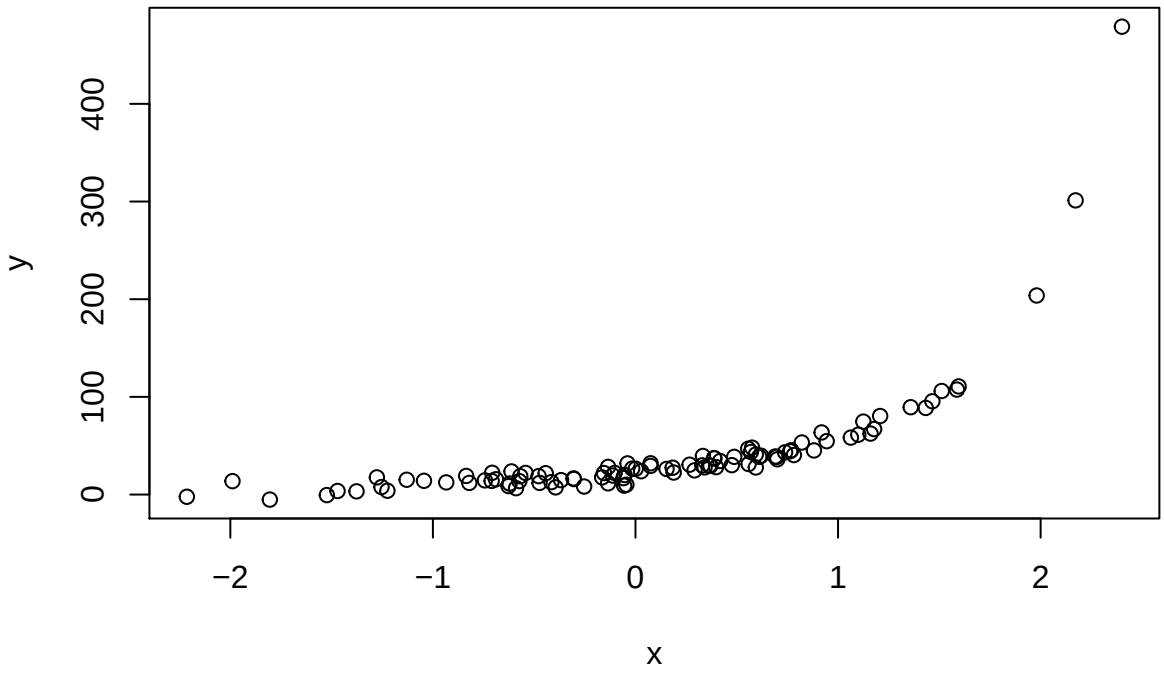
SOlution: The example is based on 100 pairs (x,y) of simulated data. The true relationship between x and y is y=2.5+20*exp(x)+0.3*x^5*exp(x). The simulated data will be fitted by simple linear regression and polynomial regressions to show bias-variance tradeoff. The flexibility of the model increases as the order of the polynomial regression increases.

The simulated data are generated as follows:

```
set.seed(1)
x=rnorm(100)
set.seed(2)
y=2.5+20*exp(x)+0.3*x^5*exp(x)+rnorm(100,0,5)
```

The following scatter plot shows the simulated dataset graphically.

```
plot(x,y)
```



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

Solution: In this case, overfitting means the model fits the sample data points (training dataset) too closely to have a very small training MSE while the test MSE is large. Underfitting means the fitted model has a very large bias from the true relationship stated in Part a of this section and does not capture the pattern of the training dataset at all. Since the lnear and polynomial regressions (up to the 10th order) will be used, the flexibility is measured by the degree of freedom of the model under use. For the most restrictive model, the simple linear model, the degree of freedom is 2. For the most flexible model, the 10th order polnomial model, the degree of freedom is 11.

c. Now create a numerical example of bias-variance tradeoff. Use some real or simulated 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). Explain what data you use (or generate) and the class of models, showing all R code. You might find it easiest to use a hold-out test data set.

Solution: The data were generated previsouly in Part a of this section which are 100 pairs (x,y) of simulated data points with true relationship as y=2.5+20*exp(x)+0.3*xˆ5*exp(x). The data were generated in Part a of this section. The class of models to perform this analysis is linear and polynomial regressions. They are the simple linear model to the 10th order polynomial model. The flexibility increases as the order of the polynomial regression increases. The previsouly generated data is saved to the dataframe "sim" as follows.

```
sim=data.frame(y=y,x=x)
```

The code to generate the training MSEs and test MSEs for the 10 models is as follows. tr.error represents training MSE and cv.error represents test MSEs.

```
tr.error.10=rep(0,10)
cv.error.10=rep(0,10)
set.seed(100)
```

```
for(i in 1:10){
  glm.fit=glm(y~poly(x,i),data=sim)
  tr.error.10[i]=mean((y-predict(glm.fit,sim))^2)
  cv.error.10[i]=cv.glm(sim,glm.fit,K=10)$delta[1]
}
```

The training MSEs are as follows:

```
tr.error.10
```

```
##  [1] 1910.30788  800.41089  253.50563   90.96223   45.92390   31.38758
##  [7]   30.04195   29.92689   28.86677   25.95078
```

The test MSEs are as follows:

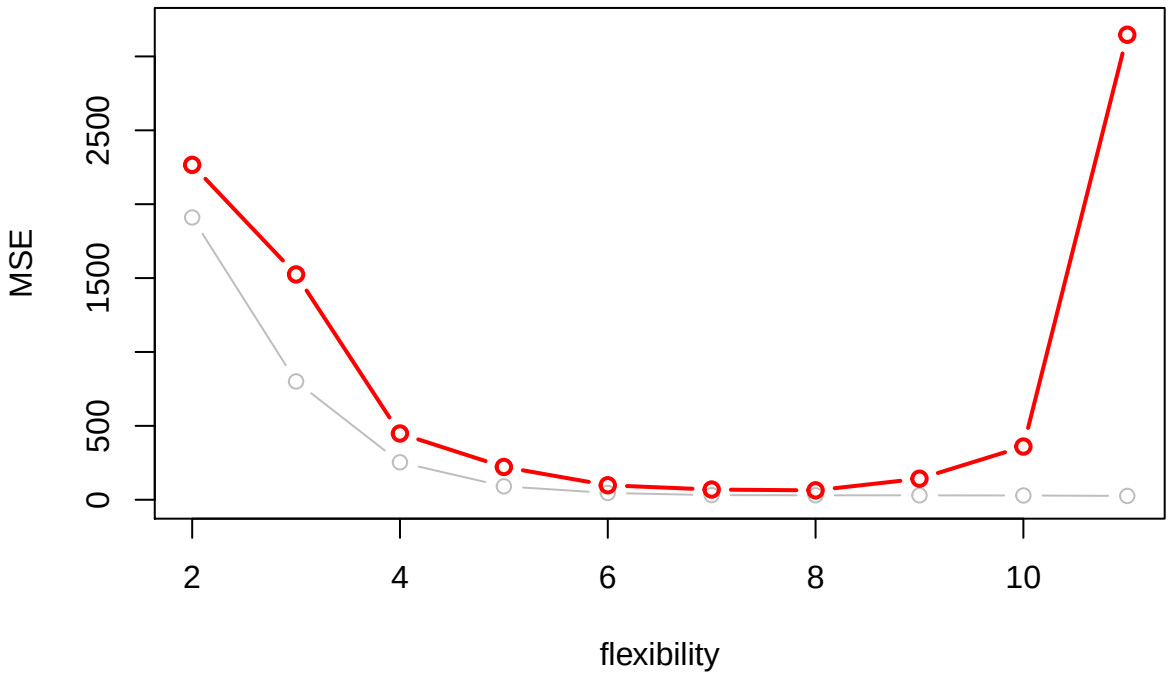
```
cv.error.10
```

```
##  [1] 2266.26855 1524.66617  448.36869  221.97284   98.09290   69.05351
##  [7]   63.60289  142.58484  359.96459 3146.57688
```

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).

Solution: The plot is created as follows:

```
flex=2:11
plot(flex,tr.error.10,type="b",col="gray",xlab="flexibility",ylab="MSE",ylim=c(0,3200))
lines(flex,cv.error.10,type="b",col="red",lwd=2)
```



The red line represents the trend of test MSEs while the gray line represents the trend of training MSEs. As shown in the above plot, the test MSE trend exhibits a "u" shape as the flexibility of the fitting model increases and the training MSE trend appears to be monotonically decreasing as the model flexibility rises.

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