Scribing For February 10: Walk through of the Home Work

Scribing by Erin Larson and William Herbst

Part 1: Boot strapping

A) With bootstrapping, you can simulate real-world variation. What we did at first was to fit a linear model to all the data, and then we asked for the summary to find the standard error. However, if we bootstrap and take a large number of samples (with replacement) from our sample, we can get something that looks a little more like real world uncertainty. If you take the standard error of your bootstrapping output, you see that it is larger than the standard error generated by the fit model of the whole sample. This explains why our original Beta was different from the published Yahoo beta. To test this, you can input:

```
names(marketmodel)
lm1=lm(WMT~SP500, data=marketmodel)
summary(lm1)

boot1= do(1000)*lm(WMT~SP500, data=marketmodel)
hist(boot1$SP500)
summary(boot1)
```

B) There are two different ways to do this, shown below:

```
#1B

#This method produces baseline/offset form so the first unit of
output (Intercept) represents ARCHITECTURE. The difference between
Arch and Liberal Arts is then the 7th column, which is why "7" is
referenced in the histogram
boot=do(1000)*lm(SAT.Q ~ School, data=resample(ut2000))
lm(SAT.Q ~ School, data= resample(ut2000))
boot2=do(1000)*coef(lm(SAT.Q ~ School, data= resample(ut2000)))

#quantify uncertainty around thedifference btwn ARCH and LibArts
confint(boot)

#plot distribution of differences
hist(boot[,7])

#Another Method:
#You could also solve this by making school-specific subsets
names(ut2000)
subArch=subset(ut2000, School='ARCHITECTURE')
subLA=subset(ut2000, School='Lberal Arts')

#Find Mean SAT for each school
mean(resample(subArch)$SAT.Q)
mean(resample(subLA)$SAT.Q)
```

```
#Bootstrap to find the means from random samples
boot3=do(1000)*(mean(resample(subArch)$SAT.Q)-
mean(resample(subLA)$SAT.Q))
confint(boot3)
```

You may notice some difference between the output of these two methods. This comes as a result of both montecarlo error (from the bootstrapping) and because the first method deals with the entire data frame, while the second takes a sample strictly from the schools required. The first method's samples may include a varying number of cases for each school (they don't draw the same number of cases from each school every time the simulation runs), which is especially important for Architecture, which has only 32 observations in the first place.

Part 2: Prediction Interval Problem

A) Prediction of the amount of variation in y predicted by X implies R squared (otherwise known as the Coefficient of Determination. Fit a linear model to the data, and then ask for the summary. If R^2 > 90%, then you can use the cheap test.

```
lm1=lm(expensive~cheap, data=shocks)
summary(lm1)
```

B) Now test the data against the stricter set of parameters

```
#Is the slope close to 1?
confint(lm1, level=.95)

#Now Test the Specific values
newdata=data.frame(cheap= c(510,550,590))

#use the predict function, you will get three columns of data
predict(lm1,newdata,interval='prediction',level=0.95)

#Subtract data to see if it falls within the 33 point rule
#One test will fail:
618.3066-581.5288

#Why use the predict function? It takes into account the uncertainty
of the std. err and B0 + B1x. However, we have to assume normality
of the data to use it.
hist(resid(lm1))

2*sd(resid(lm1)) #this yields a naïve result…. Ignores the
variability in the B0 + B1x
```

Part 3: Challenge: Cross Validation

The idea behind cross validation is that you take a portion of your sample, fit a model to it, and then use that model as a predictor on the other portion of your sample (these are the training and test sets). If the model that was fit to the training set is a reasonable predictor of the test set, then you can assume that your model is a good predictor of the population.

```
#Check column names and number of observations
names(utilities)
N=nrow(utilities)

#About 50% of the data used to train, 50% to test
Ntrain=59
Ntest= N- Ntrain

#Sample the overall data set to create training set
trainset=sample(1:N,Ntrain)
util.train=utilities[trainset,]
util.test=utilities[-trainset,]

#Fit linear models… only showing linear and quadratic but you get the
point
lm1=lm(gasbill~temp,data=util.test)
lm2=lm(gasbill~temp + I(temp^2),data=util.train)

#Predict the testing data
pred1=predict(lm1,newdata=util.test)
pred2=predict(lm2,newdata=util.test)

#Compute each model's mean-squared error (keeps the negative residuals
from being canceled out)
mse1= mean((pred1=util.test$gasbill)^2)
mse2=mean((pred2-util.test$gasbill)^2)

#Now we want to take multiple samples and average the result
 nSims=100

#Create matrix (suitcase to store your loop results in)
Mysim= matrix(0, nrow=nSims, ncol=4)

#begin FOR loop and store results in matrix "suitcase"
For (1=n, 1:nSims) {
   #Pick new train set
   trainset=sample(1:N,Ntrain)
   util.train=utilities[trainset,]
   util.test=utilities[-trainset,]

   #Fit model to sample
   lm1=lm(gasbill~temp,data=util.test)
   lm2=lm(gasbill~temp + I(temp^2),data=util.train)

   #Predict on test set
```

```
    pred1=predict(lm1,newdata=util.test)
    pred2=predict(lm2,newdata=util.test)

    #Compute MSE
    mse1= mean((pred1=util.test$gasbill)^2)
    mse2=mean((pred2-util.test$gasbill)^2)

    # Store result as row in the matrix
    Mysim= c(mse1, mse2 )
}

#Display average across all samples
Colmeans(Mysim)
```

Further Notes from the End of Class

Assumptions we make to do the LM function (to fit regression)

1) Linearity
2) Residuals are normally distributed
3) Independence→ no one residual provides information about another residual
4) Constant Variance→ homoskedasticity

If any of these assumptions are violated, residuals and predictions may be wrong

Non-independent residuals often occur in time-related data, as there may be seasonal patterns

B_1 → true, unknown slope

$\widehat{B_1}$→ estimate of the slope

$\widehat{\sigma_1}$ → standard error of the estimated slope

t_1→ (difference between the estimate and truth/estimated error), or the signal to noise ratio, it is usually calculated in respect to a reference value, which is usually 0 (for hypothesis testing purposes)

tstat: ($\widehat{B_1}$- B_1)/ $\widehat{\sigma_1}$= truth/error