Economics 217 - Sampling and Resampling methods

- Topics covered in this lecture
  - Monte Carlo Simulation
  - Bootstrap Resampling
  - Bootstrap percentile intervals
- We will be doing lots of coding in these lectures, so make sure you follow lecture with trying examples at home.

Introduction to Monte Carlo

- Most of the models we have discussed utilize large sample properties and the central limit theorem
  - With large samples, we know something about the distribution of the estimates.
  - Small sample properties are much more difficult to derive, and in practice the sampling distribution is often unknown
- Sampling and resampling techniques help us test our models under small samples and evaluate the bias that might come from results using small samples and real data.
- We will first study **Monte Carlo Analysis**.
- Simple put, we generate fake data and then test the statistics of interest using the fake data
- Model should perform well in two dimensions
  - Estimate should be centered around the actual value
  - Should not over or under reject true parameter.

A simple Monte Carlo to test small sample OLS

- Suppose we start with the following very simple model:

$$y_i = x_i + u_i$$

- Intercept is zero, slope coefficient $\beta = 1$
- Let's test this model using the following procedure:
  1. Pick a sample size of N
  2. Generate N values of x_i between zero and 1
  3. Randomly generate N values of u_i between zero and 1 using some distribution
  4. Using values of x_i and u_i, generate y_i
  5. Estimate model and collect estimates for β
- Repeat procedure B times.
- What do you think will happen if we have a small sample?

R example: 10 replications

```
B<-10
N<-50
results<-data.frame(matrix(NA, nrow=B, ncol=3))
names(results)<-c("rep","B0","B1")

for(rep in 1:B){
     x<-rnorm(N,mean=0,sd=1)
     u<-rnorm(N,mean=0,sd=1)
     y<-x+u
     fit.B<-lm(y~x)
     results$rep[rep]<-rep
     results$B0[rep]<-as.numeric(coef(fit.B)[1])
     results$B1[rep]<-as.numeric(coef(fit.B)[2])
}

par(mfrow=c(2,2))
plot(density(results$B0),main="Sampling Distribution of B0, B=10")
abline(v=0)
plot(density(results$B1),main="Sampling Distribution of B1, B=10")
abline(v=1)
```

Monte Carlo to Bootstrap

- To summarize, Monte Carlo analysis is great for assessing the performance of a estimator
  - Is it biased?
  - Is inference reasonable?
- However, there is a big down-side
  - We're testing a known function with fake data. Need a technique to evaluate bias within a real-world context
  - If we are running a monte carlo on a linear regression, we're essentially evaluating the central limit theorem (which we know works in the limit)
- There is no silver bullet, but the *Bootstrap* is as close as we get.
  - **Resamples** from observational data to create an **empirical distribution** of a test statistic
  - Develops confidence intervals and other techniques for inference from this empirical distribution

Bootstrap Logic

- The "Bootstrap" is a technique that was original proposed by Bradley Efron in the 70's
  - The cliche is apt here - we pick the data up by its "bootstraps".
- The essential bootstrap insight is the following:
  - Typical inference is justified via the central limit theorem
  - The central limit theorem is crucial since we usually only have one sample to work with.
  - However, as the sample is from the population, a **resample** of the sample is also a sample from the population.
  - Hence, we can generate sampling variation by resampling from the sample of the population.
- Critical Questions for a bootstrap analysis
  - How do we resample?
  - What statistic do we use (T-stat, CI, etc..)?

Three Methods of Resampling

- Data resampling:
  - Sampling observations of (y_i, x_i) from the original data, *with replacement*.
- Residual Resampling
  - Collect residuals from the original model, $\widehat{u}$
  - Sample these residuals with replacement to construct a new vector of residuals of equal size, $\widetilde{u}$
  - Define new dependent variable as

$$\widetilde{y} = \widehat{y} + \widetilde{u}$$

- Wild Bootstrap
  - Collect residuals from the original model, $\widehat{u}$
  - Randomly multiply them by -1 or 1 with equal probability, getting , $\widetilde{u}$
  - Define new dependent variable as

$$\widetilde{y} = \widehat{y} + \widetilde{u}$$

After resampling

- After choosing a method of resampling, we run the resampling procedure B times, getting B observations of a statistic of interest
  - Usually parameter estimates
  - Could be t-statistics
- Assuming that we are collecting parameters, it is most straightforward to calculate bootstrap confidence intervals.
- Precisely, after collecting B estimates using the bootstrap samples, the "Percentile Confidence Interval" is simply the following:

$$C = (\ \ q(\alpha/2)\ \ ,\ \ q(1-\alpha/2)\ \ )$$

  where α is the desired level of significance (say, 5%), and $q(x)$ is the xth percentile of the bootstrap estimates
- For example, if $\alpha = 10$, $q(\alpha/2)$ is the 5th percentile of the bootstrap estimates, and $q(1-\alpha/2)$ is the 95th percentile.

R Bootstrapping

- The function "boot" takes care of much of bootstrapping, but we will do it ourselves to build on the understanding of the procedure.

- The main item we need is a function to resample from a vector or data frame, with replacement.

```
sample(x, size, replace = FALSE, prob = NULL)
```

  - If "x" is a number, is draws samples from 1:x
  - If "x" is a vector, is draws samples from that vector
  - "size" is the number of observations in the desired sample
  - "replace" indicates if you want sampling done with replacement.

- To construct a random sample from a data frame, defining our data frame as "df", write the following

```
df[sample(nrow(df), n, replace=TRUE),]
```

- We use this syntax to construct bootstrap samples.

R Bootstrapping

- One can place the random sampling within a function to make the entire procedure modular

```
randomSample = function(df,n) {
    return(df[sample(nrow(df),n),])
}
```

- Finally, we run the bootstrap procedure via the following

```
form<-as.formula(log(rw)~educ)
fit.full<-lm(form,subd)

B<-1000
N<-1000
resultsB<-matrix(NA,nrow=B,ncol = (length(coef(fit.B))+1))

for(rep in 1:B){
      fit.B<-lm(form,randomSample(subd,N))
      coef.B<-as.numeric(coef(fit.B))
      resultsB[rep,1]<-rep
      resultsB[rep,2:ncol(resultsB)]<-t(as.matrix(coef.B))
}

resultsB<-as.data.frame(resultsB)
names(resultsB)<-c("rep",names(coef(fit.full)))
```

R Bootstrapping

- To construct 95% confidence intervals, run:

```
quantile(resultsB$(Intercept),prob=c(0.025,0.975),na.rm=TRUE)

quantile(resultsB$educCollege,prob=c(0.025,0.975),na.rm=TRUE)

quantile(resultsB$educAdvanced,prob=c(0.025,0.975),na.rm=TRUE)
```

- Also, compare the original estimate of *educAdvanced* against the entire distribution of bootstrap estimates

```
plot(density(resultsB$educAdvanced)main="Coefficient on Advanced
Degree")

abline(v=coef(fit.full)[5])
```

Residual Resampling

- Another approach to bootstrapping is "residual bootstrapping".
- Estimate a base model, and then use the model and "new" residuals to generate new outcome variables.
- New residuals are resampled, with replacement, from the old residuals

```
resid.full<-as.numeric(fit.full$residuals)
predict.full<-as.numeric(fit.full$fitted.values)

B<-1000
resultsR<-matrix(NA,nrow=B,ncol = (length(coef(fit.B))+1))

for(rep in 1:B){
   rand.resid<-sample(residuals.full, nrow(subd),replace=TRUE)
   subd$rw_boot<-predict.full+newresid
   fit.B<-lm(rw_boot~educ,subd)
   coef.B<-as.numeric(coef(fit.B))
   resultsR[rep,1]<-rep
   resultsR[rep,2:ncol(resultsR)]<-t(as.matrix(coef.B))
}
```

Wild Bootstrap

- The Wild bootstrap generates new residuals by randomly multiplying old residuals by -1 or 1.

- New residuals are resampled, with replacement, from the old residuals

```
resid.full<-as.numeric(fit.full$residuals)
predict.full<-as.numeric(fit.full$fitted.values)

B<-1000
resultsW<-matrix(NA,nrow=B,ncol = (length(coef(fit.B))+1))

for(rep in 1:B){
    newresid<-ifelse(runif(nrow(subd),0,1)>0.5,rand.resid,-rand.resid)
    subd$rw_boot<-predict.full+newresid
    fit.B<-lm(rw_boot~educ,subd)
    coef.B<-as.numeric(coef(fit.B))
    resultsW[rep,1]<-rep
    resultsW[rep,2:ncol(resultsW)]<-t(as.matrix(coef.B))
}
```