Bootstrap Code.pdf

Lecture 1.pdf

Lecture 2 supplementary.pdf

Lecture 2.pdf

Lecture 3.pdf

Lecture 4.pdf

Lecture 5.pdf

Lecture 6.pdf

Lecture 7.pdf

Lecture 8.pdf

Lecture 9.pdf

Lecture 10.pdf

Lecture 11.pdf

```
##### Example 2 - Bootstrap



library(foreign)

d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")

subd<-subset(d,year==2013&state=="CA")
subd$hourslw<-ifelse(is.na(subd$hourslw)==TRUE,0,subd$hourslw)
subd<-subset(subd,is.na(educ)==FALSE&is.na(nilf)==FALSE&rw>0)
subd<-subd[order(subd$age),]



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



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



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

print(rep)

}


resultsB<-as.data.frame(resultsB)

names(resultsB)<-c("rep",names(coef(fit.full)))


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

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




plot(density(resultsB$educAdvanced), "Coefficient on Advanced Degree \n Original Estimate and Empirical
Distribution",ylim=c(0,16))

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









##### Example 3 - Residual Bootstrap
```

```
library(foreign)

d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")

subd<-subset(d,year==2013&state=="CA")
subd$hourslw<-ifelse(is.na(subd$hourslw)==TRUE,0,subd$hourslw)
subd<-subset(subd,is.na(educ)==FALSE&is.na(nilf)==FALSE&rw>0)
subd<-subd[order(subd$age),]

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

fit.full<-lm(log(rw)~educ,subd)
summary(fit.full)
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.full))+1))

for(rep in 1:B){
rand.resid<-sample(resid.full, nrow(subd),replace=TRUE)
subd$rw_boot<-predict.full+rand.resid
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))

print(rep)

}


resultsR<-as.data.frame(resultsR)


names(resultsR)<-c("rep",names(coef(fit.full)))


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


lines(density(resultsR$educAdvanced),col="blue")




##### Example 4 - Wild Bootstrap



fit.full<-lm(log(rw)~educ,subd)

summary(fit.full)

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.full))+1))
```

```
for(rep in 1:B){
newresid<-ifelse(runif(nrow(subd),0,1)>0.5,resid.full,-resid.full)
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))
print(rep)
}

resultsW<-as.data.frame(resultsW)
names(resultsW)<-c("rep",names(coef(fit.full)))
quantile(resultsW$educAdvanced,prob=c(0.025,0.975),na.rm=TRUE)
lines(density(resultsW$educAdvanced),col="red")
```

Economics 217 - Applied Econometrics II

- Professor: Alan Spearot
  - Email: aspearot@ucsc.edu
  - Office Hours: 1:00-3:00PM Wednesday, 459 Engineering 2
- TA: Jijian Fan
  - Email: jifan@ucsc.edu
  - Section times: TBD
  - Office Hours: TBD
- Exams (non cumulative)
  - Exam 1: Tuesday, January, 31th (in class)
  - Exam 2: Thursday, February, 23th (in class)
  - Exam 3: Monday, March 21th (final exam period)

Economics 217 - Applied Econometrics II

- Topics covered in this course
  - Generalized Linear and Non-linear Models
  - Non-parametric models
  - Resampling techniques
  - Learning models and Data Mining
  - Time series econometrics
- Books and course materials:
  - Asteriou and Hall for time series (same book as 216)
  - Online handouts and vignettes
- Programing
  - R will be the backbone of the course (www.r-project.org)

Economics 217 - Applied Econometrics II

- Basic course expectations
  - There will be some derivations. I will try to keep the course as applied as possible but the course is meant to be more than just programming.
  - Too many people just load data into a computer and look at the results without thinking about where it came from or what the results mean.
  - Code will be presented in class. It is not expected that you sit here with your computer open running code with me (that is for 294A). However, you will need it for the homework and beyond so it is a critical part of the course.
- Assignments
  - Exams will be closed notes.
  - You may work in groups to work on assignments, but you must turn in original work for homework answers. If I see identical work being turned in, we'll have a chat in my office.
  - Explicit cheating will be dealt with harshly.

Linear models to generalized linear models

- In 216, the primary focus was estimation and inference using *linear models*

$$y_i = \mathbf{x}_i^T \beta + u_i$$

  - y_i: dependent variable
  - $\mathbf{x}_i$: $p \times 1$ vector of independent variables
  - β: $p \times 1$ coefficients on the independent variables
  - u_i: error term for individual i.

- An similar representation that will be useful going forward is

$$\mu_i = \mathbf{x}_i^T \beta, \quad where \quad Y_i \sim F(\mu_i)$$

- That is, $\mathbf{x}_i^T \beta$ represents the average of another variable Y_i, where Y_i is random around this average according to the distribution function $f()$

- For the most part, we will assume that observations are independent from one another.

Linear models to generalized linear models (cont)

- There are two ways to estimate the canonical linear model
- Linear regression
  - the command "lm" in R estimates the linear model using linear regression
  - Standard errors assume homoskedasticity, though there are ways to allow for clustering and heteroskedasticity
- Maximum likelihood, also known as normal regression
  - the command "glm" in R estimates the normal regression as a default
  - Why this command is called "glm" is the purpose of the next few lectures.
- What were some of the positives and negatives of linear models, either linear regression or normal regression?

Two ways of estimating the linear model in R

- In R, everything (usually) starts with something called a "data frame"
- For example, if I want to load a CSV file of data and call it "classdata", I write:

classdata<-read.csv(filename,header=TRUE)

  - "filename" is the location of the file on your hard drive, server, website, whatever...
  - "<-" is the assignment operator. The thing on the right is assigned to the thing on the left. This works for a variety of data types in R.
  - "header=TRUE" indicates that the first row of the file identifies column names. This obviously won't be used if there are no headers, in which case you can assign variable names later.

- After loading what is called the "foreign" library, one can load stata .dta files in R

classdata<-read.dta(filename)

- There are a variety of ways to load other data formats - see online documentation.

Current Population Survey

- The CPS is a widely used and influential dataset from the US Bureau of Labor Statistics
- The Center for Economics and Policy Research (CEPR) has cleaned this data for use in a way which is consistent over time
  - http://ceprdata.org/cps-uniform-data-extracts/
  - Outgoing Rotational Group: 1979-2013.
  - Data on Wages, Demographics, locations, Industries, Occupations, Labor Force Participation, Education, etc...
  - We'll combine surveys from 1983, 1988,...2008, 2013 to a "pooled cross-section".
- On the course website, this pooled cross-section is listed as "Org.data"
- To load it in R, I write:

```
d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")
```

Summarizing Data in R

- Though much of these techniques are better learned in lab or on your own, I will be going over some basic commands in R to summarize and manipulate data.

- The first lists the contents and variable types within the dataset, and is called "str":

```
str(d)
```

- The second is appropriately named "summary":

```
summary(d)
```

- Summary can be used on individual variables:

```
summary(d$rw)
```

  - The $ picks off a particular variable within the data frame
  - "rw" is the real wage in the org dataset

Two ways of estimating the linear model in R (cont)

- First, let's run the linear regression using the command:

```
wage.lm<-lm(log(rw)~age,d))
```

- Here, log(rw)~age is the regression equation.
  - *Age* is in years
  - *log*(*rw*) is the natural log of the real wage
- To time the procedure, you can instead run the following:

```
ptm <- proc.time()
wage.lm<-lm(log(rw)~age,d)
proc.time() - ptm
```

  - *proc.time*() reports the time when this function is called
  - *proc.time*() − *ptm* subtracts the previous time from the current time.

Two ways of estimating the linear model in R (cont)

- Next, let's run the normal regression using the **glm** command (using the timing command from earlier):

```
ptm <- proc.time()
wage.lm<-glm(log(rw)~age,d,family=gaussian))
proc.time() - ptm
```

- Again, log(rw)~age is the regression equation.
- family=gaussian indicates that we are using a gaussian (normal) distribution to model the error term
- GLM is estimated by maximum likelihood (as we'll see).
- What are the differences in estimates, t-statistics, standard errors, and regression performance?

Linear models to generalized linear models

- In many cases, as we discussed earlier, the canonical linear model is too simple.
- Thankfully, there is a parsimonious and powerful extension of the linear model called the *generalized linear model*

$$g(\mu_i) = \mathbf{x}_i^T \beta$$

- $g()$ is called the **link function**. Naturally, it links the mean of the observed variable to the linear function of covariates.
  - This is a crucial step in enriching our models. $g()$ bounds the link between observable data and outcomes.
  - What technique from 216 does this remind you of?
- Where this all becomes "generalized" is when we introduce a wide class of distributions that have nice properties within this setup.

The exponential family of distributions

- Generalized linear models are based (in part) on the *exponential family of distributions*
- The canonical exponential family is any distribution that can be arranged as.

$$f(y;\theta) = \exp(yb(\theta) + c(\theta) + d(y))$$

  - y is the random variable
  - θ is some parameter of interest.

- A nice property is that the exponential family is log-additive. That is, take logs of $f(y;\theta)$

$$\log(f(y;\theta)) = yb(\theta) + c(\theta) + d(y)$$

- This makes manipulating the Log-likelihood function particularly useful.

The exponential family: Gaussian

- Many distributions are of the exponential family.
- Consider the normal distribution

$$f(y;\mu) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{1}{2\sigma^2}(y-\mu)^2\right]$$

- Simplify by expanding $(y-\mu)^2$

$$f(y;\mu) = \exp\left[-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}\log\left(2\pi\sigma^2\right)\right]$$

- σ^2 is considered a nuisance parameter, and the focus is on μ.

$$b(\mu) = \frac{\mu}{\sigma^2} \quad , \quad c(\mu) = -\frac{\mu^2}{2\sigma^2} - \frac{1}{2}\log\left(2\pi\sigma^2\right) \quad , \quad d(y) = -\frac{y^2}{2\sigma^2}$$

- These functions will be used later when estimating equations based around these distributions.

The exponential family: Gaussian in R

- In R, there are two ways to view distributions of random variables.
- The first is to use the function and plot the function using a sequence of numbers.
- For the Gaussian distribution, this is done with the following commands:

1. Create a vector of length 1000 evenly spaced between -10 and 10.

```
x<-seq(-10,10,length.out=1000)
```

2. Insert this vector into the standard normal distribution

```
pdf<-dnorm(x,mean = 0, sd = 1, log = FALSE)
```

3. Plot the PDF as a function of x.

```
plot(pdf x, ylab="pdf",xlab="x",main="PDF of Standard normal")
```

The exponential family: Gaussian in R

- For my tastes, the best way to view distributions is to generate a large vector of random variables, and then view it with a histogram or a kernel density.
- For the Gaussian distribution, first generate a random vector of length 1000 using the command 'rnorm'

```
x<-rnorm(1000,mean = 0, sd = 1)
```

- Then, plot the density using the plot command

```
plot(density(x),xlab='x',ylab='pdf of x')
```

- We'll learn about the technical details of kernel densities during the *non-parametric* section of the course, but basically we are estimating a smooth relationship between one variable (pdf) and the other (x).
- why do you think this estimate looks bad? What can be done about it?

The exponential family: Binomial

- As you learned last quarter with the probit and logit, not all variables are continuous and in some cases they are dichotomous.

- The binomial distribution represents such variables, where p is the probability some event occurs:

$$f(y;p) = \binom{n}{y} p^y (1-p)^{n-y}$$

- y is the number of times the event has occurred after n trials.

- As a remind, n "choose" y is $\binom{n}{y} = \frac{n!}{y!(n-y)!}$.

- p is the parameter of interest. Rearranging, we get:

$$f(y;p) = \exp\left[y\log(p) + \log(1-p)(n-y) + \log\binom{n}{y}\right]$$

- Collecting y's

$$f(y;p) = \exp\left[y\log\left(\frac{p}{1-p}\right) + n\log(1-p) + \log\binom{n}{y}\right]$$

- Here, $b(p) = \log\left(\frac{p}{1-p}\right)$, $c(p) = n\log(1-p)$, and $d(y) = \log\binom{n}{y}$

The exponential family: Binomial in R

- To visualize this distribution, like before, generate a random vector of length 1000 but instead using the command 'rbinom'

```
x<-rbinom(size=1000,n=100,p=0.5)
```

- Three components of this function
  - size=1000 is the length of the random vector
  - n=100 represents a distribution based on n trials
  - p=0.5 is the probability of the event occuring
- Then, plot the density using the plot command

```
plot(density(x),xlab='x',ylab='pdf of x')
```

- What does this distribution look like?

The exponential family: Poisson

- The Poisson distribution is another extremely important distribution, commonly used to represent count variables.
  - It can also be used with continuous data to bound variables below at zero.
- Assume for now that y is a discrete random variable taking on values 0 and higher. The poisson distribution is written as:

$$f(y;\theta) = \frac{\theta^y \exp[-\theta]}{y!}$$

- Quick simplification yields:

$$f(y;\theta) = \exp[y\log(\theta) - \theta - \log(y!)]$$

- To estimate this in R, use

```
x<-rpois(n,theta)
```

  where n is the length of the random vector.

- Then, plot as before (note R calls theta "lambda", but I'm following the book with notation).

Logit and Probit in GLM

- You learned (last quarter) how to estimate the logit and probit models. Both can be structured using the GLM framework
- Both Probit and Logit are of the binomial family.
  - With the binomial distribution, p is the probability that an event occurs, and p is also the mean of the dichotomous variable, μ.
- The "link" function tells use how we link μ_i to the vector of explanatory variables
- For the logit model, use the logit function for the link:

$$g(\mu) = \log\left(\frac{\mu_i}{1-\mu_i}\right) = \mathbf{x}_i^T\beta$$

- Exponentiating both sides:

$$\left(\frac{\mu_i}{1-\mu_i}\right) = \exp\left(\mathbf{x}_i^T\beta\right)$$

- Rearranging for μ

$$\mu_i = \frac{\exp\left(\mathbf{x}_i^T\beta\right)}{1+\exp\left(\mathbf{x}_i^T\beta\right)}$$

  which is the familiar logit formula.

R Examples - Logit and Probit

- For probit, use the inverse of the standard normal for the link:

$$\begin{aligned} & g(\mu_i) &=& \Phi^{-1}(\mu_i) \\ \Rightarrow \quad & \mu_i &=& \Phi\left(\mathbf{x}_i^T \beta\right) \end{aligned}$$

- To estimate a logit model in R, use

```
glm(y x,classdata,family=binomial(link="logit"))
```

- This looks like any other regression with the exception of the family/link component
  - family=binomial() indicates that we have a dichotomous variable, one or zero
  - the "link" function tells use how we link the mean to the vector of explanatory variables
  - If we are using profit, replace "logit" with "probit" in code

Estimating Logit and Probit

- Let's now estimate a logit model using the Org dataset.
- In the Org dataset, *nilf* is an indicator variable taking on the value of 1 if the respondent is not in the labor force
  - This is crucial of estimates of unemployment, since to be unemployed you must be in the labor force
  - Also, *nilf* will also include "discouraged workers", in the sense that fluctuations in this variable will represent cyclical changes in willingness to search for work.
- To estimate the Logit model, use:

```
glm_logit<-glm(nilf~age,d,family=binomial(link="logit"))
```

- For Probit:

```
glm_probit<-glm(nilf~age,d,family=binomial(link="probit"))
```

- R provides the estimates for β's. Use "summary" to report standard errors and such.
- To compute marginal effects, additional commands should be used.

Aside estimating Logit and Probit with factor variables

- So far, our regressions with the Org dataset have been pretty simple
  - No dummy variables other other factors to absorb variation in the dependent variable or test other hypotheses
  - Not using panel data to examine variation within individuals
- In R, *factor variables* can be used to easily estimate fixed effects
  - We're going to ignore one major problem for a while (what is called "incidental parameters") when estimating fixed factors using maximum likelihood, and instead focus on syntax
- In Org, *educ* is a factor variable identifying maximum education level as less than high school, high school degree, some college, college degree, and advanced
- We can add education to our model using the following

```
glm_logit_ed<-glm(nilf~age+educ,d,family=binomial(link="logit"))
glm_probit_ed<-glm(nilf~age+educ,d,family=binomial(link="probit"))
```

GLM Extensions: Poisson Regression

- Similar to the logit-probit models, we often wish to constrain the response to explanatory variables to feasible values
- Many variables have a distribution that is non-negative $y \in [0, \infty)$
  - wages cannot be non-negative (unless you're a musician)
  - Time or durations are non-negative
  - Any count data is non-negative
- The Poisson regression is suitable to estimate these types of models
- Poisson has mostly been used to estimate count and simple duration models, but recently has been applied to any model with a non-negative dependent variable.
  - Dollars and cents are count variables, right?

GLM Extensions: Poisson Regression

- Recall that the Poisson distribution is written by:

$$f(y;\theta) = \exp[y\log(\theta) - \theta - (\log y!)]$$

- We will show later that the mean of the Poisson distribution is:

$$\mathrm{E}(y) = \theta$$

- So, the distribution can be rewritten as:

$$f(y;\mu) = \exp[y\log(\mu) - \mu - (\log y!)]$$

- We now need a link function. The log-linear model is the typical choice:

$$g(\mu) = \log(\mu) = x\beta$$

- Thus, the mean of the Poisson distribution is bounded by the exponential function:

$$\mathrm{E}[y|x] = \mu = \exp(x\beta)$$

GLM Extensions: Poisson Regression

- To estimate the Poisson regression, the typical syntax is

```
glm(y~x,classdata,family=poission(link="log"))
```

- Within the Org dataset, we will predict hours worked (*hourslw*) with age and the level of education for all individuals in the labor force (but not necessarily working).

- To make estimation go a bit more quickly, we first restrict the sample to include:
  - Only residents from California
  - Only respondents from the 2013 survey
  - Only individuals in the labor force

- This is done with:

```
subd<-subset(d,nilf==0&year==2013&state=="CA")
```

- "subd" is the smaller dataframe we will work with.

GLM Extensions: Poisson Regression

- Next, note that some individuals have "NA" listed as hours worked. We want to change these to zero using the following command

```
sub$hourslw<-ifelse(is.na(subd$hourslw),0,subd$hourslw)
```

- Since hours worked is bounded below at zero, Poisson is a reasonable approach to estimate the determinants of hours worked.

- Estimate the Poisson model, and compare to the normal regression

```
normalreg<-glm(hourslw~age+educ,subd,family=gaussian)
summary(normalreg)
poissonreg<-glm(hourslw~age+educ,subd,family=poisson(link="log"))
summary(poissonreg)
```

- What are the differences in residuals and regression diagnostics?

Predictions in GLM Models

- Predictions are central to applied applications
  - Predict clicking behavior on ads
  - Prediction intervals for stock prices
- A vast majority of R commands use "predict()" to generate a vector of predictions
- Example using Logit

```
glm_logit<-glm(nilf~age+educ,d,family=binomial(link="logit"))
glm_predict_1<-predict(glm_logit)
summary(glm_predict_1)
length(glm_predict_1)
nrow(d)
```

- What do you notice about the predictions?

Predictions in GLM Models

- There are two issues
  - The vector of predictions is, by default, the same length as the vector of feasible output
  - The predictions are on the scale of the link function, not the response
- Two solutions (respectively):
  - Define "newdata" as the original dataset, in this case "d".
  - Use option type="response".
- Example using Logit

```
glm_predict_2<-predict(glm_logit,newdata=d,type="response")
summary(glm_predict_2)
length(glm_predict_2)
nrow(d)
d$nilf_predict<-as.numeric(glm_predict_2)
```

- You can also extract standard errors of the predictions

```
glm_predict_3<-predict(glm_logit,newdata=d,type="response", se=TRUE)
```

- Command is similar for "lm" but without option for type.

Lecture 2 - Technical Aspects of GLM estimation

- Topics Covered
  - First and Second Moment for the canonical exponential Family
  - Maximum Likelihood
  - Newton-Raphson
  - Fisher Information
  - Inference in GLMs

The exponential family: First Moment

- GLMs with the canonical exponential family can be estimated using the same technique and the same function with R (with slight adjustments to the syntax)
- Part of the reason is that they also have a similar form of the mean and variance of their distributions.
- To see this, start with one of the basic properties of all distribution functions:

$$\int f(y;\theta)\,dy = 1$$

- Differentiating with respect to θ

$$\int \frac{df(y;\theta)}{d\theta} dy = 0$$

- Any changes to the distribution through θ must cancel each other out over the support of y.

The exponential family: First Moment (cont)

- Recall that

$$f(y;\theta) = \exp\left(yb(\theta) + c(\theta) + d(y)\right)$$

- Differentiating with respect to θ

$$\begin{aligned}\frac{df(y;\theta)}{d\theta} &= \left(yb'(\theta) + c'(\theta)\right)\exp\left(yb(\theta) + c(\theta) + d(y)\right) \\ &= \left(yb'(\theta) + c'(\theta)\right)f(y;\theta)\end{aligned}$$

- Plugging into $\int \frac{df(y;\theta)}{d\theta}dy = 0$, we have:

$$\int \left(yb'(\theta) + c'(\theta)\right)f(y;\theta)\,dy = 0$$

- Breaking the integral into two parts:

$$b'(\theta)\int yf(y;\theta)\,dy + c'(\theta)\int f(y;\theta)\,dy = 0$$

- How do I simplify these components?

The exponential family: First Moment (cont)

- One definition and one property that are useful:

$$\mathrm{E}(y) = \int y f(y;\theta)\,dy \quad , \quad \int f(y;\theta)\,dy = 1$$

- Thus,

$$b'(\theta)\underbrace{\int y f(y;\theta)\,dy}_{=\mathrm{E}(y)} + c'(\theta)\underbrace{\int f(y;\theta)\,dy}_{=1} = 0$$

$$b'(\theta)\mathrm{E}(y) + c'(\theta) = 0$$

$$\Rightarrow \quad \mathrm{E}(y) = -\frac{c'(\theta)}{b'(\theta)}$$

- Both $b(\theta)$ and $c(\theta)$ affect the mean of the y.
  - $c(\theta)$ is often called the "scale" function/parameter
  - $b(\theta)$ is often called the "shape" function, since it interacts with y.
- These can be most clearly seen when taking the log of the PDF:

$$\log(f(y;\theta)) = yb(\theta) + c(\theta) + d(y)$$

The exponential family: Second Moment

- To solve for variance, differentiate $\int \frac{df(y;\theta)}{d\theta} dy = 0$ with respect to θ

$$\int \frac{d^2 f(y;\theta)}{d\theta^2} dy = 0$$

- Recalling that:

$$\frac{df(y;\theta)}{d\theta} = \left(yb'(\theta) + c'(\theta)\right) f(y;\theta)$$

- We take a second derivative to get:

$$\begin{aligned}
\frac{d^2 f(y;\theta)}{d\theta^2} &= \left(yb''(\theta) + c''(\theta)\right) f(y;\theta) + \left(yb'(\theta) + c'(\theta)\right)^2 f(y;\theta) \\
&= \left(yb''(\theta) + c''(\theta)\right) f(y;\theta) + b'(\theta)^2 \left(y + \frac{c'(\theta)}{b'(\theta)}\right)^2 f(y;\theta) \\
&= \left(yb''(\theta) + c''(\theta)\right) f(y;\theta) + b'(\theta)^2 (y - \mathrm{E}(y))^2 f(y;\theta)
\end{aligned}$$

- To complete the derivation, substitute into $\int \frac{d^2 f(y;\theta)}{d\theta^2} dy = 0$

The exponential family: Second Moment (cont)

- Precisely,

$$\int \big(yb''(\theta)+c''(\theta)\big)f(y;\theta)+b'(\theta)^2(y-\mathrm{E}(y))^2 f(y;\theta)\,dy=0$$

- Using the same operations as before, first distribute the integral:

$$b''(\theta)\int yf(y;\theta)\,dy+c''(\theta)\int f(y;\theta)\,dy+b'(\theta)^2\int (y-\mathrm{E}(y))^2 f(y;\theta)\,dy=0$$

- Then impose the definition of expectations and variance:

$$b''(\theta)\mathrm{E}(y)+c''(\theta)+b'(\theta)^2\mathrm{Var}(Y)=0$$

- Finally, solving for variance:

$$\mathrm{Var}(Y)=-\frac{b''(\theta)\mathrm{E}(y)+c''(\theta)}{b'(\theta)^2}$$

The exponential family: Summary

- Thus, for the canonical exponential family of distributions,

$$f(y;\theta) = \exp\left(yb(\theta) + c(\theta) + d(y)\right),$$

the mean and variance of the variables are precisely characterized by the functions $b(\theta)$ and $c(\theta)$

$$\begin{aligned} \mathrm{E}(y) &= -\frac{c'(\theta)}{b'(\theta)} \\ \mathrm{Var}(Y) &= -\frac{b''(\theta)\mathrm{E}(y) + c''(\theta)}{b'(\theta)^2} \end{aligned}$$

- Thus, the parameters we estimate are linked to the mean and variance through these equations.

Maximum Likelihood Estimation

- All of these properties are helpful for estimating relationships that are assumed to follow the canonical exponential family.

- As you might recall from 216, the likelihood function is written as:

$$L = \prod_{i=1}^{N} f(y_i; \theta)$$

- The Log-likelihood function, $l = \log(L)$, is

$$l = \sum_{i=1}^{N} \log\left(f(y_i; \theta)\right)$$

- Within the exponential family,

$$l = \sum_{i=1}^{N} \left[a(y_i)b(\theta) + c(\theta) + d(y_i)\right]$$

- Remember that θ links to some underlying mean parameter of the model, μ, which is the mean of y, which itself links to the covariates by the link function

- When choosing optimal θ, only $b(\theta)$ and $c(\theta)$ and outcomes y_i matter.

Maximum Likelihood Estimation

- The derivative of the log-likelihood function with respect to some parameter θ is called the "score", U.

$$\begin{aligned} U \equiv \frac{dl}{d\theta} &= \sum_{i=1}^{N} \frac{d}{d\theta} \log f(y_i;\theta) \\ &= \sum_{i=1}^{N} \frac{\frac{d}{d\theta} f(y_i;\theta)}{f(y_i;\theta)} \end{aligned}$$

- The expected value of U is zero. To see this, note that

$$\begin{aligned} \mathrm{E}[U] &= \sum_{i=1}^{N} \mathrm{E}\left[\frac{\frac{d}{d\theta} f(y_i;\theta)}{f(y_i;\theta)} \right] \\ &= \sum_{i=1}^{N} \int \frac{\frac{d}{d\theta} f(y;\theta)}{f(y;\theta)} f(y;\theta)\, dy \\ &= \sum_{i=1}^{N} \int \frac{d}{d\theta} f(y;\theta)\, dy \\ &= \sum_{i=1}^{N} \frac{d}{d\theta} \underbrace{\int f(y;\theta)\, dy}_{=1} = 0 \end{aligned}$$

Maximum Likelihood for Exponential Family

- To make this simple to start, let us assume that:

$$g(\mu) = \beta$$

- Under this assumption, we are essentially choosing one value of θ that is the same for every person, since the mean of y is assumed to be invariant to other covariates

- After estimating θ, then we can link to μ using the assumed distribution, and then β using the link function..

- Taking the derivative of l with respect to θ

$$U = \frac{dl}{d\theta} = \sum_{i=1}^{N} \frac{dl_i}{d\theta} = 0$$

- For univariate functions, this can be done by hand in some cases

- Though in practice, this is done using standard computational techniques, such as Newton-Raphson.

Univariate Numerical Optimization by Newton-Raphson

- The idea behind Newton-Raphson is pretty simple. Suppose you have a function $U(\theta)$, and you want to find the roots of the function.

$$U(\theta) = 0$$

- For Newton-Raphson, we iterate over different values for θ, trying to find a solution. θ^m is defined as the "mth" iteration (not to the power of m).

- Suppose that were are at a value θ^{m-1}, and would like to approximate the function $U(\theta)$ at θ^m. By a first-order Taylor series approximation:

$$U(\theta^m) = U(\theta^{m-1}) + \frac{dU(\theta)}{d\theta}\left(\theta^m - \theta^{m-1}\right)$$

- Substituting $U(\theta^m) = 0$, and solving for θ^m, we have

$$\begin{aligned} 0 &= U(\theta^{m-1}) + \frac{dU(\theta)}{d\theta}\left(\theta^m - \theta^{m-1}\right) \\ 0 &= \frac{U(\theta^{m-1})}{\frac{dU(\theta)}{d\theta}} + \left(\theta^m - \theta^{m-1}\right) \\ \Rightarrow \quad \theta^m &= \theta^{m-1} - \frac{U(\theta^{m-1})}{\frac{dU(\theta^{m-1})}{d\theta}} \end{aligned}$$

- The Newton-Raphson algorithm is based on this equation

Univariate Numerical Optimization by Newton-Raphson

- Newton-Raphson algorithm
  1. Begin with an initial guess, θ^0
  2. Solve for
  $$\theta^1 = \theta^0 - \frac{U(\theta^0)}{\frac{dU(\theta^0)}{d\theta}}$$
  3. If $\left|\theta^1 - \theta^0\right| < \epsilon$, then stop.
  4. If $\left|\theta^1 - \theta^0\right| > \epsilon$, then use θ^1 as initial guess and repeat from step 1.
- This always works when nicely behavior functions (continuous, differentiable) have a unique, global maximum.
- Other techniques are used when you cannot guarantee a unique global maximum. They all seem to have funny names (simulated annealing, particle swarm, etc..)
- Broyden's method is a variant of Newton-Raphson that approximates $\frac{dU(\theta^0)}{d\theta}$ using past changes in the function. Useful, but very slow. If you can take derivatives, you can speed up the process.

Newton-Raphson Example

- Here is a simple version of Newton-Raphson. We wish to find the value at which the following function is zero:

$$f(x) = (x-1)^2$$

- Obviously, we know the answer is $x = 1$. But, let's work through this iteratively.

- For newton-raphson, we need an initial guess. Let's say $x^0 = 0$

- Next, we need the derivative of the function.

$$\frac{df(x)}{dx} = 2x - 2$$

- Now, we iterate!

$$\begin{aligned} x^1 &= x^0 - \frac{f(x^0)}{\frac{df(x^0)}{dx}} \\ &= 0 - \frac{f(0)}{\frac{df(0)}{dx}} \\ x^1 &= 0 - \frac{1}{-2} = \frac{1}{2} \end{aligned}$$

Newton-Raphson Example

- Again!!

$$\begin{aligned} x^2 &= x^1 - \frac{f(x^1)}{\frac{df(x^1)}{dx}} \\ &= \frac{1}{2} - \frac{f(\frac{1}{2})}{\frac{df(\frac{1}{2})}{dx}} \\ x^2 &= \frac{1}{2} - \frac{\frac{1}{4}}{-1} = \frac{1}{4} \end{aligned}$$

- Check the value of $f(x)$

$$f(\frac{1}{4}) = (\frac{1}{4} - 1)^2 = \frac{9}{16} \neq 0$$

- Difference in x's: $|\frac{1}{4} - 0| = \frac{1}{4}$

Newton-Raphson Example

- Again!!

$$\begin{aligned} x^3 &= x^2 - \frac{f(x^2)}{\frac{df(x^2)}{dx}} \\ &= \frac{1}{4} - \frac{f(\frac{1}{4})}{\frac{df(\frac{1}{4})}{dx}} \\ &= \frac{1}{4} - \frac{\frac{9}{16}}{-\frac{6}{4}} \\ x^2 &= \frac{1}{4} + \frac{3}{3} = 5/8 \end{aligned}$$

- Check the value of $f(x)$

$$f(\frac{5}{8}) = (\frac{5}{8} - 1)^2 = \frac{9}{64}$$

- We are closer to 0 for the outcome.

- Difference in x's: $|\frac{1}{4} - \frac{5}{8}| = \frac{3}{8}$

Newton-Raphson Example

- Again!!

$$\begin{aligned} x^4 &= x^3 - \frac{f(x^3)}{\frac{df(x^3)}{dx}} \\ &= \frac{5}{8} - \frac{f(frac58)}{\frac{df(frac58)}{dx}} \\ &= \frac{5}{8} - \frac{\frac{9}{64}}{-\frac{6}{8}} \\ x^4 &= \frac{5}{8} + \frac{9}{48} = \frac{39}{48} \end{aligned}$$

- Check the value of $f(x)$

$$f(\frac{39}{48}) = (\frac{39}{48} - 1)^2 = \left(\frac{9}{48}\right)^2$$

- We are closer to 0 for the outcome.

- Difference in x's: $|\frac{39}{48} - \frac{5}{8}| = \frac{3}{16}$

- We'll stop here, but you keep going until the difference in x's is small enough.

Multivariate Newton Raphson

- Newton Raphson can be extended to a setting with multiple variables over which we maximize a function.

- Suppose that there are p variables, indexed $\beta_j, j = 1...p$, over which we are maximizing a function f

- For this case,

$$\frac{df}{d\beta_j} \equiv U_j(\beta) = 0$$

must equal zero for all j, where β represents the $px1$ vector of β_j's

- A multi-variate first-order taylor-series expansion is written as:

$$\mathbf{U}^m = \mathbf{U}^{m-1} + \mathbf{J}^{m-1}\left(\beta^m - \beta^{m-1}\right)$$

where:

  - $\mathbf{J}^{m-1}$ is the Jacobian matrix of U at iteration $m-1$

  - $\mathbf{U}^m$ is the $px1$ vector of scoring values at iteration m.

Multivariate Newton Raphson (cont.)

- As a reminder, the Jacobian is a *pxp* matrix with $\frac{dU_j}{d\beta_k}$ is the j^{th} row and k^{th} column.
- The element in the j^{th} row and k^{th} column of $\mathbf{J}$ is written as J_{jk}
- Trying to hit $\mathbf{U}^m = \mathbf{0}$ (all scores equal to zero) using the first-order approximation, we get:

$$\mathbf{0} = \mathbf{U}^{m-1} + \mathbf{J}^{m-1}\left(\beta^m - \beta^{m-1}\right)$$

- Rearranging:

$$\beta^m \quad = \quad \beta^{m-1} - \left(\mathbf{J}^{m-1}\right)^{-1}\mathbf{U}^{m-1}$$

- Again, we iterate until a solution.

Multivariate Maximum Likelihood for Exponential Family

- We now extend our earlier model to allow for a vector of covariates (which may include constants)

$$g(\mu_i) = \mathbf{x}_i^T \beta$$

- Recall that μ_i links to to the mean of the distribution by θ_i
- Taking the derivative of l with respect to some parameter β_j

$$U_j = \frac{dl}{d\beta_j} = \sum_{i=1}^{N} \frac{dl_i}{d\theta_i} \frac{d\theta_i}{d\mu_i} \frac{d\mu_i}{d\beta_j}$$

- $\frac{dl_i}{d\theta_i}$ is once again written as:

$$\begin{aligned} \frac{dl_i}{d\theta_i} &= \frac{d}{d\theta_i}\left(y_i b(\theta_i) + c(\theta_i) + d(y_i)\right) \\ &= y_i b'(\theta_i) + c'(\theta_i) \\ &= b'(\theta_i)\left(y_i + \frac{c'(\theta_i)}{b'(\theta_i)}\right) = b'(\theta_i)\left(y_i - \mu_i\right) \end{aligned}$$

- The last step is since $\mu_i = \mathrm{E}(Y_i) = -\frac{c'(\theta)}{b'(\theta)}$

Multivariate Maximum Likelihood for Exponential Family

- $\frac{d\theta_i}{d\mu_i}$ is the inverse of $\frac{d\mu_i}{d\theta_i}$:

$$\begin{aligned}\frac{d\mu_i}{d\theta_i} &= -\frac{c''(\theta_i)b'(\theta_i) - c'(\theta_i)b''(\theta_i)}{b'(\theta_i)^2} \\ &= -b'(\theta_i)\frac{c''(\theta_i) - c'(\theta_i)\frac{b''(\theta_i)}{b'(\theta_i)}}{b'(\theta_i)^2} = b'(\theta_i)\mathrm{Var}(Y_i)\end{aligned}$$

- Thus,

$$\frac{d\theta_i}{d\mu_i} = \frac{1}{b'(\theta)\mathrm{Var}(Y_i)}$$

- Finally, since $g(\mu_i) = \mathbf{x}_i^T\beta$, we have:

$$\begin{aligned}\frac{dg(\mu_i)}{d\mu_i}\frac{d\mu_i}{d\beta_j} &= x_{ij} \\ \Rightarrow \quad \frac{d\mu_i}{d\beta_j} &= \frac{x_{ij}}{\frac{dg(\mu_i)}{d\mu_i}}\end{aligned}$$

- Overall, we have that the derivative of the likelihood function (the "score") is:

$$U_j = \sum_{i=1}^{N}\frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)}\frac{x_{ij}}{\frac{dg(\mu)}{d\mu}} = 0$$

- To find the maximum likelihood estimates, U_j must be zero for all j.

Examples of Scoring Functions: Gaussian

- Gaussian regression with the identity link:
  - Identity link: $g(\mu_i) = \mu_i = x_i^T \beta$
  - Gaussian Distribution: $\text{Var}(Y_i) = \sigma$
- Thus, the score can be written as:

$$\begin{aligned} U_j &= \sum_{i=1}^{N} \frac{(y_i - \mu_i)}{\text{Var}(Y_i)} \frac{x_{ij}}{\frac{dg(\mu)}{d\mu}} = 0 \\ &= \sum_{i=1}^{N} \frac{\left(y_i - x_i^T \beta\right)}{\sigma} \frac{x_{ij}}{1} = 0 \\ &= \sum_{i=1}^{N} \left(y_i - x_i^T \beta\right) x_{ij} = 0 \end{aligned}$$

- What does this remind you of?

Examples of Scoring Functions: Poisson

- Recall the Poisson distribution:

$$f(y;\theta) = \frac{\theta^y \exp[-\theta]}{y!}$$

- Poisson has a very cool property:
  - $\mathrm{E}(Y_i) = \mathrm{Var}(Y_i) = \theta_i$
- Assuming the identity link: $g(\mu_i) = \mu_i = x_i^T \beta = \theta_i$
- Thus, the score can be written as:

$$\begin{aligned} U_j &= \sum_{i=1}^{N} \frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{x_{ij}}{\frac{dg(\mu_i)}{d\mu_i}} = 0 \\ &= \sum_{i=1}^{N} \frac{\left(y_i - x_i^T \beta\right) x_{ij}}{x_i^T \beta} = 0 \end{aligned}$$

- We will use this a bit later when continuing the Poisson example

Multivariate Maximum Likelihood for Exponential Family

- The last piece for multivariate estimation of GLM models is the *information matrix*, **J**, which is made up of the elements J_{jk}
  - **J** is also called the "Fisher Information Matrix", named after Ronald Fisher.
  - Accuracy or (information given by X) around the maximum likelihood solution is defined by the curvature of the likelihood function at these points. This is why we call it information.
- The element J_{jk} is simply the covariance between score functions

$$J_{jk} = \mathrm{E}\left[U_j U_k\right]$$

- Importantly, for GLM models, J_{jk} is also the Jacobian matrix of the scoring functions (or, the Hessian matrix for the log-likelihood function)
- Thus, the information matrix is used in optimization, as well in variance-covariance estimation.

Information Matrix

- Using the formula for U_j, $\mathrm{E}\left[U_j U_k\right]$ can be written as:

$$\mathrm{E}\left[U_j U_k\right] = \mathrm{E}\left(\sum_{i=1}^{N} \frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{x_{ij}}{\frac{dg(\mu_i)}{d\mu_i}} \sum_{l=1}^{N} \frac{(y_l - \mu_l)}{\mathrm{Var}(Y_l)} \frac{x_{ik}}{\frac{dg(\mu_l)}{d\mu_l}} \right)$$

- Expanding the summation into the square and cross-products

$$\mathrm{E}\left[U_j U_k\right] = \mathrm{E}\left(\sum_{i=1}^{N} \frac{(y_i - \mu_i)^2}{\mathrm{Var}(Y_i)^2} \frac{x_{ij} x_{ik}}{\left(\frac{dg(\mu_i)}{d\mu_i}\right)^2} \right) + \mathrm{E}\left(\sum_{i=1}^{N} \sum_{l \neq i}^{N} \frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{x_{ij}}{\frac{dg(\mu_i)}{d\mu_i}} \frac{(y_l - \mu_l)}{\mathrm{Var}(Y_l)} \frac{x_{lk}}{\frac{dg(\mu_l)}{d\mu_l}} \right)$$

- Since the expectation is only applied to random data (y's)

$$\begin{aligned} \mathrm{E}\left[U_j U_k\right] \quad = \quad & \left(\sum_{i=1}^{N} \frac{\mathrm{E}(y_i - \mu_i)^2}{\mathrm{Var}(Y_i)^2} \frac{x_{ij} x_{ik}}{\left(\frac{dg(\mu_i)}{d\mu_i}\right)^2} \right) \\ & + \left(\sum_{i=1}^{N} \sum_{l \neq i}^{N} \frac{1}{\mathrm{Var}(Y_i)} \frac{x_{ij}}{\frac{dg(\mu_i)}{d\mu_i}} \frac{1}{\mathrm{Var}(Y_l)} \frac{x_{lk}}{\frac{dg(\mu_l)}{d\mu_l}} \mathrm{E}\left[(y_i - \mu_i)(y_l - \mu_l)\right] \right) \end{aligned}$$

- If observations are independent $\mathrm{E}\left[(y_i - \mu_i)(y_l - \mu_l)\right] = 0$ for all $i \neq l$. Finally,

$$J_{jk} = \mathrm{E}\left[U_j U_k\right] = \sum_{i=1}^{N} \frac{1}{\mathrm{Var}(Y_i)} \frac{x_{ij} x_{ik}}{\left(\frac{dg(\mu_i)}{d\mu_i}\right)^2}$$

Examples of Information Matrix

- We wish to simplify the following elements of the matrix **J**

$$J_{jk} = \mathrm{E}\left[U_j U_k\right] = \sum_{i=1}^{N} \frac{1}{\mathrm{Var}\left(Y_i\right)} \frac{x_{ij} x_{ik}}{\left(\frac{dg(\mu_i)}{d\mu_i}\right)^2}$$

- For **Gaussian**, assuming an identity link, we get:

$$J_{jk} = \mathrm{E}\left[U_j U_k\right] = \frac{1}{\sigma} \sum_{i=1}^{N} x_{ij} x_{ik}$$

- For **Poisson**, assuming an identity link, $\mathrm{Var}\left(Y_i\right) = x_i^T \beta$, we get:

$$J_{jk} = \mathrm{E}\left[U_j U_k\right] = \sum_{i=1}^{N} \frac{x_{ij} x_{ik}}{x_i^T \beta}$$

- Let's now write out the entire procedure for Poisson and $\mu_i = \beta_1 x_{i1} + \beta_2 x_{i2}$, where $x_{i1} = 1$ for all i (ie. a constant)
  - That is, $\mu_i = \beta_1 + \beta_2 x_{i2}$

Examples of Information Matrix

- Since $x_{i1} = 1$ for all i, J_{11} is written as:

$$J_{11} = \mathrm{E}[U_1 U_1] = \sum_{i=1}^{N} \frac{1}{\beta_1 + \beta_2 x_{i2}}$$

- J_{12} is written as:

$$J_{12} = \mathrm{E}[U_1 U_2] = \sum_{i=1}^{N} \frac{x_{i2}}{\beta_1 + \beta_2 x_{i2}}$$

- J_{21} is written as:

$$J_{21} = \mathrm{E}[U_2 U_1] = \sum_{i=1}^{N} \frac{x_{i2}}{\beta_1 + \beta_2 x_{i2}}$$

- J_{22} is written as:

$$J_{22} = \mathrm{E}[U_2 U_2] = \sum_{i=1}^{N} \frac{x_{i2}^2}{\beta_1 + \beta_2 x_{i2}}$$

- On your own, you should write this for the Gaussian distribution under the same link $\mu_i = \beta_1 + \beta_2 x_{i2}$.

Examples of Information Matrix

- Thus, we can write the matrix $\mathbf{J}$

$$\mathbf{J} = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} \frac{1}{\beta_1+\beta_2 x_{i2}} & \sum_{i=1}^{N} \frac{x_{i2}}{\beta_1+\beta_2 x_{i2}} \\ \sum_{i=1}^{N} \frac{x_{i2}}{\beta_1+\beta_2 x_{i2}} & \sum_{i=1}^{N} \frac{x_{i2}^2}{\beta_1+\beta_2 x_{i2}} \end{pmatrix}$$

- Recalling that the score is written as:

$$U_j = \sum_{i=1}^{N} \frac{\left(y_i - x_i^T \beta\right) x_{ij}}{x_i^T \beta} = 0$$

- A matrix $\mathbf{U}$ of scoring functions can be written as:

$$\mathbf{U} = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} \frac{y_i-\beta_1-\beta_2 x_{i2}}{\beta_1+\beta_2 x_{i2}} \\ \sum_{i=1}^{N} \frac{(y_i-\beta_1-\beta_2 x_{i2})x_{i2}}{\beta_1+\beta_2 x_{i2}} \end{pmatrix}$$

- So, by Newton Raphson, we find our solution by iterating the following:

$$\begin{pmatrix} \beta_1^{new} \\ \beta_2^{new} \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} - \mathbf{J}^{-1}\mathbf{U}$$

- R uses "Iteratively Re-weighted Least Squares ", which is identical to this (though approached differently)

Inference in GLM Models

- For inference regarding one parameter, use t-test as you would with OLS
  - Central limit theorem works for GLMs
  - The variance-covariance matrix of β's is $\mathbf{J}^{-1}$
- For joint-tests:
  - Use F-test and F-distribution for normal regression
  - Use "Likelihood Ratio" test and Chi-square distribution for all others
- Likelihood Ratios are a simple comparison of the "maximal model", i.e. the best we could do given the data, and the actual model:

$$D = 2\left(l(\beta_{max}; y) - l(\widehat{\beta}; y)\right)$$

- D is also called "deviance", and a summary of which is provided in regression results.
- $l(\beta_{max}; y)$ is constructed by basically using y_i for μ_i in the likelihood function, and then calculating likelihood.

Likelihood Ratio Test

- The likelihood ratio tests does exactly as the name suggests - compares the likelihood of two different models.
- Suppose that $\widehat{\beta}$ are the estimates from the full unrestricted model, and $\widehat{\beta}_A$ is an alternate set of parameter estimates that impose restrictions on the model.
- To test these restrictions, first calculate:

$$\Delta D = 2\left(l(\widehat{\beta}_A; y) - l(\widehat{\beta}; y)\right)$$

- Then compare this value to $\chi^2(r, p)$, which is the value from a chi-squared distribution, where:
  - r is the number of restrictions.
  - p is the preferred probability of false rejection (note that programs, including R, may require the confidence level as opposed to probability of false rejection).

LR Test in R

- There are a few ways to execute the LR test in R.
- Can calculate the likelihood ratio directly.
- Using our previous Poisson example for hours worked, let's test for the joint effect of all education dummy categories.

```
poissonreg<-glm(hourslw~age+educ,subd,family=poisson(link="log"))
summary(poissonreg)
poissonreg2<-glm(hourslw~age,subd,family=poisson(link="log"))
summary(poissonreg2)
LR<-(poissonreg2$deviance-poissonreg$deviance)
```

- Then, we compare the LR to the Chi-square distribution

```
chi_crit<-qchisq(.95, df=4)
ifelse(LR>chi_crit,"Reject the restrictions", "Fail to reject the restrictions")
```

- Or, you can construct the P-value for false rejection

```
pchisq(LR, 4, lower.tail = FALSE)
```

LR Test in R

- There are a few ways to execute the LR test in R.
- The best is using the "lrtest" command from the "lmtest" library in R.
- Using our previous Poisson example for hours worked, let's test for the joint effect of all education dummy categories.

```
library(lmtest)
poissonreg<-glm(hourslw~age+educ,subd,family=poisson(link="log"))
summary(poissonreg)
lrtest(poissonreg,"educ")
```

- The results indicate the two models being tested, the log-likelihood for each, and the p-value from the LR test.
- Small p-values indicate that one can reject the joint restrictions.

Economics 217 - Multinomial Choice Models

- So far, most extensions of the linear model have centered on either a binary choice between two options (work or don't work) or censoring options.
- Many questions in economics involve consumers making choices between more than two varieties of goods
  - Ready-to-eat cereal
  - Vacation destinations
  - Type of car to buy
- Firms also have such multinomial choices
  - In which country to operate
  - Where to locate a store
  - Which CEO to hire
- Techniques to evaluate these questions are complex, but widely used in practice. Generally, they are referred to as **Discrete Choice Models**, or **Multinomial Choice Models**

Multinomial Choice - The basic framework

- Suppose there are individuals, indexed by i
- They choose from J options of a good, and may only choose one option.
- If they choose option k, then individual i receives U_{ik} in utility, where

$$U_{ik} = V_{ik} + \epsilon_{ik}$$

  - V_{ik} is observable utility (to the econometrician). This can be linked to things like product characteristics, demographics, etc..
  - ϵ_{ik} is random utility. The econometrician doesn't see this, but knows its distribution. This actually makes the problem a bit more reasonable to characterize empirically

- Utility maximization - individual i chooses option k if

$$U_{ik} > U_{ij} \quad \forall \quad j \neq k$$

- This maximization problem involves comparing observable utility for each option, while accounting for random utility.

Multinomial Choice - The basic framework

- From here, there are a variety of techniques that one can use to estimate multinomial choice models
- Multinomial Logit is the easiest, and will be derived below
  - Assumes a particular functional form that has questionable properties, but produces closed form solutions
  - There are two ways to derive the multinomial logit - we will go over the easier approach, though I have also derived the second approach in the notes.
- Nested Logit is more realistic:
  - Consumers choose between larger groups (car vs. truck) before making more refined choices (two-door vs. four-door)
  - Also yields closed form solutions, but results can depend on choices over "nests"
- Additional extensions to multinomial choice are beyond this course, but can be used if you understand the basic assumptions
  - Multinomial Probit (requires heavy computation)
  - Random coefficients logit (variation in how agents value attributes of choices)

Multinomial Distribution

- Recall the binomial distribution:

$$f(y;p) = \frac{n!}{y!\,(n-y)!} p^y (1-p)^{n-y}$$

- Remember that p is the probability some event (eg. unemployment) occurs, and y is the number of times the event occurs after n attempts.
  - $n-y$ is the number of times the event does not occur.
- When there are more than two choices, the distribution is generalized as **multinomial**
- Defining π_j as the probability that option j is chosen, the multinomial distribution is written as:

$$\begin{aligned} f(y;p) &= n! \prod_{j=1}^{J} \frac{1}{y_j!} \pi_j^{y_j} \\ &= \frac{n!}{y_1!y_2!\cdots y_{J-1}!y_J!} \pi_1^{y_1} \pi_2^{y_2} \cdots \pi_{J-1}^{y_{J-1}} \pi_J^{y_J} \end{aligned}$$

- This is the PDF that is used for maximum likelihood. We wish to estimate J π_j's.
  - Do you think we can? Or do you think that we need to?
- Let's now take the next step and link the likelihood function to data.

Multinomial Logit - Derivation

- Recall for the Logit model we link the log odds ratio to data

$$\log\left(\frac{p_i}{1-p_i}\right) = \mathbf{x}_i^T\beta$$

- We exponentiate and rearrange to get:

$$p_i = \frac{\exp\left(\mathbf{x}_i^T\beta\right)}{1+\exp\left(\mathbf{x}_i^T\beta\right)}$$

- We must extend this link to having multiple options in the multinomial model.
  - Since there is a linear dependency in our probabilities (ie. they sum to one), we must choose a **reference group**
- We write the log odds ratio relative to the reference group ($j = 1$) as:

$$\log\left(\frac{\pi_{ij}}{\pi_{i1}}\right) = \mathbf{x}_i^T\beta_j$$

- Note that the relative probability is specific to j: β_j.
  - β_j: The effect of some covariate on the choice between j and 1 may vary by j.
  - Eg. Price matters for choice between compact cars, but not choice between compact and luxury.

Multinomial Logit - Derivation

- Exponentiate and solve for π_{ij}

$$\pi_{ij} = \pi_1 \exp\left(\mathbf{x}_i^T \beta_j\right)$$

- Next, use the requirement that all probabilities sum to 1

$$\pi_{i1} + \sum_{j=2}^{J} \pi_j = 1$$

- Substituting for π_{ij}, we get:

$$\pi_{i1} + \sum_{j=2}^{J} \pi_1 \exp\left(\mathbf{x}_i^T \beta_j\right) = 1$$

- Solving for π_{i1}

$$\pi_{i1} = \frac{1}{1 + \sum_{j=2}^{J} \exp\left(\mathbf{x}_i^T \beta_j\right)}$$

- Thus, the probability of option j, π_{ij}, is

$$\pi_{ij} = \frac{\exp\left(\mathbf{x}_i^T \beta_j\right)}{1 + \sum_{s=2}^{J} \exp\left(\mathbf{x}_i^T \beta_s\right)}$$

Multinomial Logit - Assumptions

- The multinomial logit formula is pretty simple

$$\pi_{ij} = \frac{\exp\left(\mathbf{x}_i^T \beta_j\right)}{1 + \sum_{s=2}^{J} \exp\left(\mathbf{x}_i^T \beta_s\right)}$$

- The multinomial logit has a pretty sharp property that is usually not good in practice: Independence of Irrelevant Alternatives (IIA)

- Precisely, when choosing between two goods, substitution with other goods does not matter

- To see IIA in practice, take the ratio of probabilities between some good j and another k

$$\frac{\pi_{ij}}{\pi_{ik}} = \frac{\exp\left(\mathbf{x}_i^T \beta_j\right)}{\exp\left(\mathbf{x}_i^T \beta_k\right)} = \exp\left(\mathbf{x}_i^T \left(\beta_j - \beta_k\right)\right)$$

- Thus, the relative probabilities of two outcomes do not depend on the other $J-2$ outcomes.

- Techniques such as multinomial probit, and nested logit, avoid this strong prediction.

Multinomial Logit - Estimation in R

- There are a few packages in R to estimate the multinomial logit.
  - **mlogit** is the best.
- The package also includes a number of datasets that we can use to demonstrate the model. Since it is pretty simple, we will use the dataset "Cracker".
- After loading mlogit, you can call the data internal to the package via the following command:

```
data("Cracker", package = "mlogit")
str(Cracker)
```

- Each row represents an individual, and "choice" represents the chosen brand. This will be the outcome variable.
- For each brand of cracker, the dataset contains the following information
  - **price** observed for individual i
  - Whether or not there was an in-store display observed by individual i, **disp**.
  - Whether or not there was a newspaper ad observed by individual i, **feat**.

Multinomial Logit - Estimation in R

- To setup the data.frame for estimation, you must create an mlogit data object.

```
data_c<-mlogit.data(Cracker, shape="wide", choice="choice",
varying=c(2:13))
```

- "data_c" is the mlogit data object in "wide" formate
  - "Cracker" is the original data frame
  - "shape="wide"" tells us to list the data in a format that I will describe with R.
  - "varying=c(2:13)" indicates the variables from the dataset that vary by individual (prices they observe, advertisements they see
- To estimate the model, run:

```
m <- mlogit(choice~price+disp+feat,data_cracker)
summary(m)
```

- Can estimate the model with product specific coefficients using

```
m2 <- mlogit(choice~0|price+disp+feat,data_cracker)
summary(m2)
```

Extra: Multinomial Logit from Extreme Value Distribution

- Choices are independent of one another, and ϵ_{ik} follows an extreme value I distribution (also known as the Gumbel distribution).

$$\begin{aligned} f(\epsilon_{ik}) &= \exp(-\epsilon_{ik})\exp(-\exp(-\epsilon_{ik})) \\ \Pr(\epsilon < \epsilon_{ik}) = F(\epsilon_{ik}) &= \exp(-\exp(-\epsilon_{ik})) \end{aligned}$$

- Recall that from utility maximization - individual i chooses option k if

$$U_{ik} > U_{ij} \quad \forall \quad j \neq k$$

- We now seek the probability that this outcome occurs, which can then be compared empirically to the share of agents that choose option k over all other j.

Extra: Multinomial Logit - Derivation

- First, let's consider option k against some other option j. The probability the consumer purchases k:

$$\Pr\left(U_{ik} > U_{ij}\right) = \Pr\left(V_{ik} + \epsilon_{ik} > V_{ij} + \epsilon_{ij}\right)$$

- Rearranging to isolate ϵ_{ij}

$$\Pr\left(U_{ik} > U_{ij}\right) = \Pr\left(V_{ik} - V_{ij} + \epsilon_{ik} > \epsilon_{ij}\right)$$

- This simply says that the difference in observable utility plus ϵ_{ik} is greater than ϵ_{ij}. Put differently, unobserved utility in option j is not sufficient to make-up for the other factors influencing the decision between k and j.

- Imposing the CDF of the Gumbel distribution, and treating ϵ_{ik} as a conditioning variable, we have:

$$\Pr\left(U_{ik} > U_{ij} \middle| \epsilon_{ik}\right) = F\left(V_{ik} - V_{ij} + \epsilon_{ik}\right)$$

- Given ϵ_{ik}, what is the probability that this occurs for all $j \neq k$?

Extra: Multinomial Logit - Derivation

- Since unobserved utility is independent across goods, the intersection of these events is just their probabilities multiplied together

- So, the probability that k is chosen over j for all $j \neq k$, conditional on ϵ_{ik}, is:

$$\Pr\left(U_{ik} > U_{ij} \ \forall\, j \neq k \;\middle|\; \epsilon_{ik}\right) = \prod_{j \neq k} F\left(V_{ik} - V_{ij} + \epsilon_{ik}\right)$$

- For the final step before some algebra, recall that this is a *conditional probability*. We still need to account for the possible values of ϵ_{ik}

- Formally, the unconditional probability that k is chosen, P_{ik}, is written as:

$$P_{ik} = \Pr\left(U_{ik} > U_{ij} \ \forall\, j \neq k\right) = \int \Pr\left(U_{ik} > U_{ij} \ \forall\, j \neq k \;\middle|\; \epsilon_{ik}\right) f\left(\epsilon_{ik}\right) d\epsilon_{ik}$$

- Basically, what we're doing is taking each $\int \Pr\left(U_{ik} > U_{ij} \ \forall\, j \neq k \;\middle|\; \epsilon_{ik}\right)$, and then weighting by the pdf $f\left(\epsilon_{ik}\right)$.

Extra: Multinomial Logit - Derivation

- Imposing the solution for the choice of k conditional on ϵ_{ik}:

$$P_{ik} = \int_{-\infty}^{\infty} \prod_{j \neq k} F\left(V_{ik} - V_{ij} + \epsilon_{ik}\right) f\left(\epsilon_{ik}\right) d\epsilon_{ik}$$

- Imposing the parameterization of the extreme value distribution, we have:

$$P_{ik} = \int_{-\infty}^{\infty} \prod_{j \neq k} \exp\left(-\exp\left(-\left(V_{ik} - V_{ij} + \epsilon_{ik}\right)\right)\right) \exp\left(-\epsilon_{ik}\right) \exp\left(-\exp\left(-\epsilon_{ik}\right)\right) d\epsilon_{ik}$$

- Note that since $\exp\left(-\exp\left(-\epsilon_{ik}\right)\right) = \exp\left(-\exp\left(-\left(V_{ik} - V_{ik} + \epsilon_{ik}\right)\right)\right)$, we can simply as:

$$P_{ik} = \int_{-\infty}^{\infty} \prod_{j} \exp\left(-\exp\left(-\left(V_{ik} - V_{ij} + \epsilon_{ik}\right)\right)\right) \exp\left(-\epsilon_{ik}\right) d\epsilon_{ik}$$

- Simplifying this is not too hard, once you note a few convenient features of the extreme value distribution.

Extra: Multinomial Logit - Derivation

- Remember that the product of exponentials is just the exponential of the sums of the exponents

$$\prod_j \exp x_j = \exp\left(\sum_j x_j\right)$$

- Thus,

$$\begin{aligned} P_{ik} &= \int_{-\infty}^{\infty} \prod_j \exp\left(-\exp\left(-\left(V_{ik} - V_{ij} + \epsilon_{ik}\right)\right)\right) \exp\left(-\epsilon_{ik}\right) d\epsilon_{ik} \\ &= \int_{-\infty}^{\infty} \exp\left(-\sum_j \exp\left(-\left(V_{ik} - V_{ij} + \epsilon_{ik}\right)\right)\right) \exp\left(-\epsilon_{ik}\right) d\epsilon_{ik} \end{aligned}$$

- Using a similar rule, we can we can factor out $\exp(\epsilon_{ik})$

$$P_{ik} = \int_{-\infty}^{\infty} \exp\left(-\exp\left(-\epsilon_{ik}\right) \sum_j \exp\left(-\left(V_{ik} - V_{ij}\right)\right)\right) \exp\left(-\epsilon_{ik}\right) d\epsilon_{ik}$$

- The next step is tricky. What is the relationship between $-\exp\left(-\epsilon_{ik}\right)$ and $\exp\left(-\epsilon_{ik}\right) d\epsilon_{ik}$?

Extra: Multinomial Logit - Derivation

- Time for a change of variables, where

$$\begin{aligned} t &= -\exp(-\epsilon_{ik}) \\ dt &= \exp(-\epsilon_{ik})\, d\epsilon_{ik} \\ \textit{where}\ \ t &\in (-\infty, 0) \end{aligned}$$

- Thus,

$$\begin{aligned} P_{ik} &= \int_{-\infty}^{\infty} \exp\left(-\exp(-\epsilon_{ik}) \sum_j \exp\left(-\left(V_{ik} - V_{ij}\right)\right)\right) \exp(-\epsilon_{ik})\, d\epsilon_{ik} \\ &= \int_{-\infty}^{0} \exp\left(t \sum_j \exp\left(-\left(V_{ik} - V_{ij}\right)\right)\right) dt \end{aligned}$$

- Completing the integral:

$$P_{ik} = \left(\frac{\exp\left(t \sum_j \exp\left(-\left(V_{ik} - V_{ij}\right)\right)\right)}{\sum_j \exp\left(-\left(V_{ik} - V_{ij}\right)\right)} \right|_{-\infty}^{0}$$

Extra: Multinomial Logit - Derivation

- And finally, simplify

$$
\begin{aligned}
P_{ik} &= \frac{1}{\sum_j \exp\left(-\left(V_{ik} - V_{ij}\right)\right)} \\
&= \frac{1}{\sum_j \exp\left(-V_{ik}\right)\exp\left(V_{ij}\right)} && \text{use exponent rule} \\
&= \frac{1}{\exp\left(-V_{ik}\right)\sum_j \exp\left(V_{ij}\right)} && \text{factor out } \exp\left(-V_{ik}\right) \\
&= \frac{\exp\left(V_{ik}\right)}{\sum_j \exp\left(V_{ij}\right)} && \text{multiply top and bottom by } \exp\left(V_{ik}\right)
\end{aligned}
$$

- From here, we usually assume that observed utility is a function of covariates

$$V_{ij} = X_{ij}\beta$$

- Thus,

$$\boxed{P_{ik} = \frac{\exp\left(X_{ik}\beta\right)}{\sum_j \exp\left(X_{ij}\beta\right)}}$$

Lecture 4 - Survival Models

- Survival Models
  - Definition and Hazards
  - Kaplan Meier
  - Proportional Hazards Model
- Estimation of Survival in R

GLM Extensions: Survival Models

- Survival Models are a common and incredibly useful extension of the generalized linear model.
  - They are linked on a basic level to Poisson arrivals, which as we learned earlier, yield an exponential distribution of arrival times.
- Survival models are used across many fields
  - Medicine and biostatistics: Many drugs are used to prolong life in the face of serious illness. So, the applicability of "survival" analysis is pretty obvious
  - Firm survival and death. How long do businesses live? Eg: conditional on entering a market (or new market) today, what is the probability of bankruptcy in 12 months?
  - One can imagine survival being used to model time spent on webpages, shopping, Facebook, etc...
- In this part of the course, we'll learn the basics of survival models using the GLM methodology, and then discuss extensions for more realistic survival analysis

GLM Extensions: Survival Models

- Let y be survival time, and $f(y)$ be the pdf of survival times.

- Probability of surviving less than y is:

$$F(y) = \Pr(Y < y) = \int_0^y f(t)dt$$

- By the property of complements, the probability of surviving longer then y is the *survivor function*

$$S(y) = 1 - F(y)$$

- With the survivor function, we can define the *hazard function*, $h(y)$, which is the probability of death within a small period between y and δy, *given they have survived until* t.

$$\begin{aligned} h(y) &= \lim_{\delta \to 0} \frac{F(y+\delta y) - F(y)}{\delta y} \cdot \frac{1}{S(y)} \\ &= \frac{f(y)}{S(y)} \end{aligned}$$

- This is essentially a conditional probability. Conditional on surviving up to y or later, S(y), what is the instantaneous probability of death?

GLM Extensions: Survival Models

- For a few more definitions, it is straightforward to show that the hazard function is linked to the survivor function:

$$h(y) = -\frac{d}{dy}\log(S(y)) = -\frac{\frac{dS(y)}{dy}}{S(y)} = \frac{f(y)}{S(y)}$$

- Finally, the cumulative hazard function, $H(y)$ is written as

$$H(y) = -\log(S(y))$$

- Example: Exponential Distribution

$$\begin{aligned} f(y) &= \theta \exp(-\theta y) \\ F(y) &= \int_0^y \theta \exp(-\theta t)\, dt = (-\exp(-\theta t)\Big|_0^y = 1 - \exp(-\theta y) \end{aligned}$$

- Exponential Survivor function and Hazard:

$$S(y) = \exp(-\theta y) \quad , \quad h(y) = \theta$$

- Note that the hazard does not depend on age. Thus, the exponential distribution is "*memoryless*". When might this be a good or bad property?

GLM Extensions: Survival Models

- The memoryless property makes the exponential distribution unsuitable for a number of applications.

- The Weibull distribution is a nice alternative that nests the exponential distribution.

$$f(y) = \lambda\phi y^{\lambda-1}\exp\left(-\phi y^{\lambda}\right)$$

- Under what condition is this identical to the exponential distribution?

- The survival function of Weibull:

$$\begin{aligned} S(y) &= \int_t^{\infty} \lambda\phi t^{\lambda-1}\exp\left(-\phi t^{\lambda}\right)dt \\ &= \exp\left(-\phi y^{\lambda}\right) \end{aligned}$$

- Hence, the hazard is written as:

$$h(y) = \lambda\phi y^{\lambda-1}$$

- This is a much better property to have. The link between y and the hazard may be either positive of negative. What are some economic examples of each?

Simple Estimation: Survival Models

- One way to estimate survival models is to construct a *Kaplan-Meier* estimate of the survivor function
- For this, individuals are ordered by time of death from 1 to n
  - $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(k)}$, where n_j is the number of individuals alive just before $y_{(j)}$ and, d_j the number of deaths that occur at time $y_{(j)}$
- First, consider the probability of survival just before $y_{(1)}$.

$$\widehat{S}\left(y \in [0, y_{(1)})\right) = 1$$

- Next, probability of survival just before $y_{(2)}$.

$$\widehat{S}\left(y \in [y_{(1)}, y_{(2)})\right) = 1 \times \frac{n_1 - d_1}{n_1}$$

- Next, probability of survival just before $y_{(3)}$.

$$\widehat{S}\left(y \in [y_{(2)}, y_{(3)})\right) = 1 \times \frac{n_1 - d_1}{n_1} \times \frac{n_2 - d_2}{n_2}$$

Simple Estimation: Survival Models

- In general, the Kaplan-Meier estimate of the survivor function at time $y_{(s)}$ is the following:

$$\widehat{S}\left(y_{(s)}\right) = \prod_{j=1}^{s}\left(\frac{n_j - d_j}{n_j}\right)$$

- This can be compared to the survivor function for Exponential and Weibull distributions

$$\begin{aligned} \textit{Exponential}: \quad S(y) &= \exp(-\theta y) \\ \textit{Weibull}: \quad S(y) &= \exp\left(-\phi y^{\lambda}\right) \end{aligned}$$

- How should we go about choosing between exponential and Weibull using the KM estimates?

- Take logs of the survivor functions:

$$\begin{aligned} \textit{Exponential}: \quad \log(S(y)) &= -\theta y \\ \textit{Weibull}: \quad \log(S(y)) &= -\phi y^{\lambda} \end{aligned}$$

- Log of KM estimate should be approximately linear for exponential, non-linear for Weibull

Example: Kaplan-Meier

- To study survival models, we will use an influential study, the "Gehan-Freirich" Survival Data
- The data show the length of remission in weeks for two groups of leukemia patients, treated and control
- The library "survival" contains many function that were useful for survival models.
- In the dataset:
  - **weeks**: Weeks in remission (effectively survival)
  - **relapse**: 1 if a relapse observed, 0 otherwise (this is censoring)
  - **group**: 1 if respondent was in treatment group, 0 if in control
- To construct Kaplan-Meier Estimates:

```
fit <- survfit(Surv(weeks, relapse)~group, data = g)
plot(fit, lty = 2:3)
legend(23, 1, c("Control", "Treatment"), lty = 2:3)
```

- Try removing "relapse" and see what happens.

Estimation: Survival Models

- The importance of the survival function and hazard function become apparent when estimating the model rigorously by maximum likelihood.
- For survival analysis, the data are recorded by subject j
  - y_j is the survival time of individual j
  - $\delta_j = 1$ is a variable identifying uncensored observations, $\delta_j = 0$ if censored.
  - $\mathbf{x}_j$ a vector of explanatory variables for j.
  - Order individuals such that $j = 1..r$ are uncensored, and $j = r+1..n$ are censored
- A way to think of censoring is that individuals are still "surviving" at the end of the data collection. We do not observe when censored individuals actually die, so we must account for this in the likelihood function.
- Typically, the likelihood function is written as:

$$L = \prod_{j=1}^{n} f\left(y_j\right)$$

- This is in fact the likelihood function for estimation with uncensored data.

Estimation: Survival Models

- With censored data, the likelihood function is written as:

$$L = \prod_{j=1}^{r} f\left(y_j\right) \prod_{j=r+1}^{n} S\left(y_j\right)$$

- $f\left(y_j\right)$ is the pdf at y_j, which is appropriate for uncensored data.

- $S\left(y_j\right)$ is the probability that we observe y_j or greater, which is the appropriate likelihood to consider for censored observations.

  - We know that a censored individual j survives y_j or longer, so the likelihood of this event is $S\left(y_j\right)$

- Rearranging the likelihood function, we get:

$$L = \prod_{j=1}^{r} f\left(y_j\right)^{\delta_j} S\left(y_j\right)^{1-\delta_j}$$

- We can now place this in log-likelihood form, and impose the distributional assumptions.

Estimation: Survival Models

- In log-likelihood form:

$$
\begin{aligned}
l &= \sum_{j=1}^{n} \delta_j \log\left(f\left(y_j\right)\right) + \left(1-\delta_j\right)\log\left(S\left(y_j\right)\right) \\
&= \sum_{j=1}^{n} \delta_j \log\left(f\left(y_j\right)\right) + \log\left(S\left(y_j\right)\right) - \delta_j \log\left(S\left(y_j\right)\right) \\
&= \sum_{j=1}^{n} \delta_j \left(\log\left(f\left(y_j\right)\right) - \log\left(S\left(y_j\right)\right)\right) + \log\left(S\left(y_j\right)\right) \\
&= \sum_{j=1}^{n} \delta_j \log\left(h\left(y_j\right)\right) + \log\left(S\left(y_j\right)\right)
\end{aligned}
$$

- Intuition:
  - All individuals survive until y_j. This is accounted for in $\log\left(S\left(y_j\right)\right)$
  - However, for individuals with $\delta_j = 1$, they die at y_j. So, we account for this within the likelihood function using the hazard function, $\log\left(h\left(y_j\right)\right)$

Estimation: Exponential Survival

- Recall that exponential distribution has convenient forms for the hazard and survivor functions:

$$h(y_j) = \theta \quad , \quad S(y_j) = \exp(-\theta y_j)$$

- Thus, log-likelihood is:

$$l = \sum_{j=1}^{n} \delta_j \log(\theta_j) - \theta_j y_j$$

- This looks *a lot* like a Poisson likelihood function, with δ_j as the dependent variable. To get it even closer, write:

$$\begin{aligned} l &= \sum_{j=1}^{n} \delta_j \log\left(\theta_j \frac{y_j}{y_j}\right) - \theta_j y_j \\ &= \sum_{j=1}^{n} \delta_j \log(\theta_j y_j) - \theta_j y_j - \delta_j \log(y_j) \end{aligned}$$

- Defining $\mu_j = \theta_j y_j$, we have

$$l = \sum_{j=1}^{n} \delta_j \log(\mu_j) - \mu_j - \delta_j \log(y_j)$$

- We choose μ_j to maximize the log-likelihood.

Estimation: Exponential Survival

- Often, we assume a *proportional hazards model*, where the hazard function is related to observables, $\theta_j = \exp(\mathbf{x}\beta)$
  - While exponential is memoryless, the probability of dying at y is a function of observables (treatment vs control, for example).
- Thus, substituting into $\mu_j = \theta_j y_j$, we have

$$\mu_j = \exp(\mathbf{x}\beta) y_j$$

- Taking logs:

$$\log(\mu_j) = \mathbf{x}\beta + \log(y_j)$$

- Exponential with proportional hazards can be estimated by
  - glm in R, Poisson as family
  - log link (μ to $x\beta$)
  - Offset (of the log mean) by $\log(y_j)$

Estimation: Proportional Hazards Model in R

- Using the same data as before, we can estimate the simple exponential survival model using R

```
haz_glm<-glm(relapse~group+offset(log(weeks)),family=poisson("log"),data=g)
summary(haz_glm)
```

- "offset(log(weeks))" adjusts the link to the mean to include the offset term.
- To interpret, note that the hazard is estimated as:

$$\theta_{treat} = \exp(\beta_0 + \beta_1 Treat)$$

- Breaking up the exponential

$$\theta_{treat} = \exp(\beta_0)\exp(\beta_1 Treat)$$

- Note that $\theta_{control} = \exp(\beta_0)$. Hence:

$$\begin{aligned}
\theta_{treat} &= \theta_{control}\exp(\beta_1 Treat) \\
\frac{\theta_{treat}}{\theta_{control}} &= \exp(\beta_1) \\
\frac{\theta_{treat}}{\theta_{control}} - 1 &= \exp(\beta_1) - 1 \\
\frac{\theta_{treat} - \theta_{control}}{\theta_{control}} &= \exp(\beta_1) - 1 = \exp(-1.53) - 1 = -0.783
\end{aligned}$$

- 78% reduction in the hazard of relapse relative to control.

Economics 217 - Nonparametric Econometrics

- Topics covered in this lecture
  - Introduction to the nonparametric model
  - The role of bandwidth
  - Choice of smoothing function
  - R commands for nonparametric models
- Much of these notes are inspired by Prof. Bruce Hansen's PhD Econometrics Text.

Linear models to non-parametric models

- What is a non-parametric model?
  - A model that does not assume a strong parametric form of the relationship between independent variables and dependent variables
  - Simple OLS adopts the assumption "linear in parameters". That is, a parametric function that is linear in things we estimate
  - Non-parametric models are occasionally called semi-parametric models, though these can refer to other techniques as well so we will use non-parametric.
- Recall that the linear model can be written as:

$$y_i = \mathbf{x}_i\beta + u_i$$

- In general, the non-parametric model is written as:

$$y_i = s\left(x_{i1}, x_{i2}, \ldots x_{ip}\right) + u_i$$

- The key is choosing the particular form of $s()$, subject to a variety of practical constraints. What are the issues in choosing these functions?

Top-level issues with non-parametric models

- **Issue #1:** Functional Form
  - Ultimately, we *must* choose a form for $s\left(x_{i1}, x_{i2}, ...x_{ip}\right)$. And, there are an infinite number of choices that we have.
- For example, we could simplify:

$$y_i = s\left(x_{i1}, x_{i2}, ..., x_{ip}\right) + u_i$$

as

$$y_i = s_1\left(x_{i1}\right) + s_2\left(x_{i2}\right) + \cdots + s_p\left(x_{ip}\right) + u_i$$

- And still, even under the last form, we'd have to assume something about $s_k()$. But why didn't we use:

$$y_i = s_1\left(x_{i1}, x_{i2}\right) + s_3\left(x_{i3}\right) + \cdots + s_p\left(x_{ip}\right) + u_i$$

- The choices are (literally) endless.

Top-level issues with non-parametric models

- One option available to the researcher is to choose a parametric function that is ridiculously rich and flexible.
- For example, let's consider the univariate non-parametric model

$$y_i = s(x_i) + u_i$$

- Again, there are a lot of choices for $s()$. One (parametric) choice is the following:

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 \\ &+ \beta_5 x_i^5 + \beta_6 x_i^6 + \beta_7 \log(x_i) + \beta_8 \cos(x_i) + u_i \end{aligned}$$

- Positives for this specification:
  - Can estimate with OLS, get standard errors easily, generate predictions
- Negatives for this specification:
  - Good luck interpreting any coefficients.
  - Functional form is arbitrary.
- We will return to these types of models after we discuss the most basic non-parametric estimation procedures.

Top-level issues with non-parametric models

- A common, very simple and intuitive alternative to a flexible functional form is called "binned estimation".
- Intuitively, we break-up the data into bins, and find the best fit within these bins. Then we aggregate the results and produce a plot describing the relationship between y and x.
  - A popular technique in data science, "k-nearest neighbors", is an extended version of "binned" estimation.
- Formally, this is accomplished through the following equation

$$\widehat{s}(x) = \frac{\sum_i^n \mathbf{1}(|x_i - x| < h) y_i}{\sum_i^n \mathbf{1}(|x_i - x| < h)}$$

- In this equations, we have:
  - $\widehat{s}(x)$ The estimate form $s()$ at x
  - h: The bandwidth - the region of $x's$ over which we estimate $s()$ at x
- In this binned approach, at a given x, we take values no more than h above or below x to estimate $\widehat{s}(x)$

Top-level issues with non-parametric models

- This approach can be re-written as a function of a general weighting function.

$$\begin{aligned}
\widehat{s}(x) &= \frac{\sum_i^n \mathbf{1}\left(|x_i - x| < h\right) y_i}{\sum_i^n \mathbf{1}\left(|x_i - x| < h\right)} \\
&= \sum_i^n \underbrace{\frac{\mathbf{1}\left(|x_i - x| < h\right)}{\sum_j^n \mathbf{1}\left(|x_j - x| < h\right)}}_{w_i(x)} y_i \\
&= \sum_i^n w_i(x) y_i
\end{aligned}$$

- What is the primary issue with estimating this function?
- **Issue #2:** Weighting and Bandwidth
  - Non-parametric estimates may depend heavily on choice of weighting function $w(x)$
- We will later examine the issues with the weighting function with a few R examples

Nadaraya-Watson Estimator

- Generally, binned-estimation is called either a "local-constant estimator" or the "Nadaraya-Watson" estimator.

$$\widehat{s}(x) = \sum_{i}^{n} w_i(x) y_i$$

- The choice of $w_i(x)$ is crucial. It is helpful to redefine the weighting function as a *Kernel Function*, $k(u)$, where

$$u = \frac{x_i - x}{h}$$

and $k(u)$ has the following properties:

$$\begin{aligned} k(u) &= k(-u) \\ 0 &\leq k(u) < \infty \\ \int_{-\infty}^{\infty} k(u)du &= 1 \\ \int_{-\infty}^{\infty} u^2 k(u)du &< \infty \end{aligned}$$

- That is, $k(u)$ is a bounded pdf and symmetric about zero, with finite variance.

Nadaraya-Watson Estimator

- Choice of $k(u)$ is crucial to any non-parametric study. There are three common choices:

- **Uniform (or "box"):**
  Just as we've described above for the binned estimation

$$k(u) = \frac{1}{2}\mathbf{1}(|u| \leq 1)$$

- **Epanechnikov:**
  Like uniform, but declining weights in u^2

$$k(u) = \frac{3}{4}\left(1-u^2\right)\mathbf{1}(|u| \leq 1)$$

- **Gaussian:**
  Weighted as a standard normal distribution

$$k(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$$

R Examples: Nadaraya-Watson and Binned Estimation

- Basic Nadaraya-Watson estimation can be accomplished in R using the `ksmooth` command

- Syntax: ksmooth(x,y, type, bandwidth)
  - x: the running variable
  - y: the outcome variable
  - kernel: type of smoothing ("box" or "normal")
  - bandwidth: exactly as it sounds.

- Evaluate smooth relationship between age and labor force participation

```
plot(ksmooth(subd$age,subd$nilf, "box", bandwidth = 1), col = 1)
lines(ksmooth(subd$age,subd$nilf, "box", bandwidth = 10), col = 2)
lines(ksmooth(subd$age, subd$nilf, "box", bandwidth = 20), col = 3)
lines(ksmooth(subd$age, subd$nilf, "box", bandwidth = 40), col = 4)
```

- With the normal kernel instead of boxed

```
plot(ksmooth(subd$age,subd$nilf, "normal", bandwidth = 1), col = 1)
lines(ksmooth(subd$age,subd$nilf, "normal", bandwidth = 10), col = 2)
lines(ksmooth(subd$age, subd$nilf, "normal", bandwidth = 20), col = 3)
lines(ksmooth(subd$age, subd$nilf, "normal", bandwidth = 40), col = 4)
```

Locally linear regression

- A common alternative to Nadaraya-Watson (NW), though not necessarily better, is the locally linear regression (often called "loess" smoothing)
- Like NW, we produce an estimate for each x
- Unlike NW, we run a full linear regression rather than just estimate an intercept (though we still use an intercept)
- Formally, we solve the following for each x

$$\begin{aligned} \widehat{s}(x) &= \widehat{\alpha}(x) \\ \textit{where} & \\ \left\{\widehat{\alpha}(x), \widehat{\beta}(x)\right\} &= \underset{\alpha,\beta}{\operatorname{argmin}} \sum_{i}^{n} k\left(\frac{x_i - x}{h}\right)\left(y_i - \alpha - \beta\left(x_i - x\right)\right)^2 \end{aligned}$$

- Then, after we do this, we plot $\widehat{s}(x)$
- When do you think that the Loess regression works better than NW?

R Examples: Loess Estimator

- Again, let's evaluate the relationship between age and labor force participation.
- The "loess" function is one way to execute local-linear estimation in R.
- "loess" allows for both first and second degree polynomial smoothing.

```
fit.lm<-lm(subd$nilf~subd$age)
fit.loess1<-loess(subd$nilf~subd$age,span=1, degree=1)
fit.loess2<-loess(subd$nilf~subd$age,span=1, degree=2)
```

- And now we plot it.

```
plot(subd$age,predict(fit.lm),type="l",lwd=2,ylim=c(0,1))
lines(subd$age,predict(fit.loess1),col=1,lty=2)
lines(subd$age,predict(fit.loess2),col=4)
```

- Now let's evaluate the role of "span" which is the commands bandwidth control

```
fit.loess1<-loess(subd$nilf subd$age,span=1, degree=1)
fit.loess2<-loess(subd$nilf~subd$age,span=10, degree=1)
fit.loess3<-loess(subd$nilf~subd$age,span=.1, degree=1)
```

- Again, we plot:

```
plot(subd$age,predict(fit.lm),type="l",lwd=2,ylim=c(0,1))
lines(subd$age,predict(fit.loess1),col=1,lty=2)
lines(subd$age,predict(fit.loess2),col=3)
lines(subd$age,predict(fit.loess3),col=4)
```

Optimal Bandwidth Selection

- How do we choose optimal bandwidth?
- The tradeoffs are fairly straightforward.
  - Large h: reduces variance but increases bias and oversmoothing
  - Small h: reduces bias but increases noise
- Need a technique to systematically balance these objectives.
- **Cross Validation** is the general technique that is used for choosing bandwidth
- **Leave-one-out** bandwidth selection is a type of cross-validation, the standard approach
  - The technique itself drops an observation, generates the model, and predicts the outcome for the dropped observation using the model.
  - We choose the bandwidth that minimizes any out-of-sample prediction errors.

Optimal Bandwidth Selection

- Cross-Validation procedure
  1. Choose h
  2. Estimate $\widehat{s}(x)$ without observation i. Label this estimate $\widehat{s}_{-i}(x,h)$
  3. Calculate prediction error for i: $\tilde{e}_i = y_i - \widehat{s}_{-i}(x,h)$
  4. Repeat for all i
  5. Calculate $CV(h) = \sum_i^n \tilde{e}_i^2$
  6. Repeat for all other h.
- Choose h that minimizes $CV(h)$
- This technique could obviously take a while. For example, with a dataset of 1000 observations and 100 choices of bandwidth, 100,000 regressions are run in total.

R: Optimal Bandwidth Selection

- To demonstrate leave-one-out, let's first create some fake data

```
x<-seq(-10,10,length=1000)
y<-sin(x)+rnorm(1000,0,1)
```

- Then, let's have a look at a few loess plots and talk about what we see.

```
fit.loess1<-loess(y~x, family="gaussian",span=1, degree=1)
fit.loess2<-loess(y~x, family="gaussian",span=.05, degree=1)
plot(x,predict(fit.loess1),type="l",lwd=2,ylim=c(-2,2))
lines(x,predict(fit.loess2),col=1,lty=2)
```

- For the leave-one-out estimator, let's create a fake data frame to use

```
small<-data.frame(y,x)
```

R: Optimal Bandwidth Selection

- Again, the basic process for leave-one-out is the following
  - For each h, iterate through each observation i
    - Drop i, estimate the model with the rest
    - Use model to predict i
    - Calculate squared error of prediction $\Rightarrow$ save
  - Choose h that minimizes sum of squared residuals of the out of sample predictions
- Precisely:

```
for(h in 1:20){
  for(i in 1:nrow(small)){
    smalldrop<-small[i,]
    smallkeep<-small[-i,]
    fit<-loess(y~x,smallkeep, family="gaussian",span=(h/20), degree=1)
    dropfit<-predict(fit,smalldrop,se=FALSE)
    sqrerr<-(smalldrop$y-as.numeric(dropfit))^2
    if(i*h==1){results<-data.frame(h,i,sqrerr)}
    if(i*h>1){results<-rbind(results,data.frame(h,i,sqrerr))}
  }
}
```

- Use tapply (or some other function) to find the minimizing h

```
tapply(results$sqrerr,results$h,FUN=sum,na.rm=TRUE)
```

Series estimation

- Series estimation involves using a flexible polynomial to estimate an unknown function.
- Though earlier I mentioned that the choice of polynomial is arbitrary, there is a science behind it.
- **Stone-Weierstrass Theorem** (1885, 1937, 1948)
  - Any continuous function can be well approximated by a polynomial of a sufficiently high order.
- How do we choose such a function?
- Two main considerations:
  - Do we interact variables of interest?
  - What order polynomial should we use?

Series estimation

- Two techniques:
  - Approximation by **series**
  - Approximation by **spline**
- In the former, we essentially choose a flexible polynomial, including all powers of variables and cross-products of variables and their powers.
- Two defining features of approximation by series
  - p number of variables:
  - k order of the polynomial.
- A simple series regression, $p = 2$ and $k = 1$, is the following:

$$s(x) = \beta_0 + \beta_1 x_1 + \beta_1 x_2 + \beta_{12} x_1 x_2$$

- Assuming $p = 2$ and $k = 2$ we get:

$$\begin{aligned} s(x) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 \\ &+ \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{122} x_1 x_2^2 + \beta_{112} x_1^2 x_2 + \beta_{1122} x_1^2 x_2^2 \end{aligned}$$

- Just by going from $k = 1$ to $k = 2$, dimension more than doubled

Series estimation

- In general, series estimation has a dimension $K = (1+k)^p$
  - This can obviously get pretty big depending on the dataset and desire for a smooth fit
- There is also a downside to a polynomial fit of this type: **Runge's Phenomenon**
  - Polynomials can be very bad at interpolation.
  - In other words, they might do well predicting the actual data, but very poorly when generating out-of-sample predictions.
- To study this, let's plot the function

$$s(x) = \frac{1}{1+x^2}$$

And try to estimate it with a polynomial.

R Example: Runge's phenomenon

- Try this with linear regression, and 10th order polynomial
- First, let's create some fake data

```
x<-seq(-10,10,by=1)
y<-1/(1+x^2)
x2<-x^2
x3<-x^3
x4<-x^4
x5<-x^5
x6<-x^6
x7<-x^7
x8<-x^8
x9<-x^9
x10<-x^10
```

- Then let's plot:

```
plot(y~x,ylim=c(-0.25,1))
lines(predict(lm(y~x))~x,col=1,lwd=2)
lines(predict(lm(y~x+x2))~x,col=2,lwd=2)
lines(predict(lm(y~x+x2+x3+x4+x5+x6+x7+x8+x9+x10))~x,col=3,lwd=2)
```

R Example: Runge's phenomenon

- Next, let's generate a new dataset, and evaluate out-of-sample predictions

```
xnew<-data.frame(x=seq(-5,5,by=0.01))
xnew$x2<-xnew$x^2
xnew$x3<-xnew$x^3
xnew$x4<-xnew$x^4
xnew$x5<-xnew$x^5
xnew$x6<-xnew$x^6
xnew$x7<-xnew$x^7
xnew$x8<-xnew$x^8
xnew$x9<-xnew$x^9
xnew$x10<-xnew$x^10
ynew<-1/(1+xnew$x^2)
```

- Then let's plot:

```
plot(ynew~xnew$x,ylim=c(-0.25,1),cex=0.25)
lines(predict(lm(y~x+x2+x3+x4+x5+x6+x7+x8+x9+x10),xnew)~xnew$x,col=4,lwd=2)
```

- The original fit was created using 11 data points evenly spaced between -5 and 5. How did we do away from these points?

Spline estimation

- Spline estimation is an alternative to series estimation which is also based on polynomials, but allows for the polynomial to "evolve" with the value of the dependent variable.

- To develop a spline model, suppose that $s(x)$ is univariate, and that $x \in \left(\underline{x}, \overline{x}\right)$

- Further, suppose that we have chosen N "knots" $\{t_1, t_2, ..., t_N\} \in \left(\underline{x}, \overline{x}\right)$.

  - These knots split up the relevant range of x, and as you will see, are crucial to the estimation of spline functions.

- With these knots, a spline function is defined by the following:

$$s(x) = \sum_{j=0}^{k} \beta_k x^k + \sum_{z=1}^{N} \gamma_z \left(x - t_z\right)^k \mathbf{1}\left(x \geq t_z\right)$$

- Characteristics of the spline function

  - Conitinuous derivatives up to $k-1$

  - In practice k is usually 3 to have continuous second derivatives.

Spline estimation

$$s(x) = \sum_{j=0}^{k} \beta_k x^k + \sum_{z=1}^{N} \gamma_z (x - t_z)^k \mathbf{1}(x \geq t_z)$$

- There are two critical parts to the spline function:
  - $\sum_{j=0}^{k} \beta_k x^k$ is the basic polynomial
  - $\sum_{z=1}^{N} \gamma_k (x - t_z)^k \mathbf{1}(x \geq t_z)$ at the maximum degree
- Critical issues moving forward is choosing t_z's. Cross-validation is the technique that is typically used for this
- However, must choose either flexibility (location of t_z's), or depth of changes to the polynomial (number of t_z's).
  - If you limit yourself to very small number of t_z's, can grid search over a few t_z's.
  - If you want to possibly have a lot of flexibility in the spline, evenly space

R Example: Spline estimation

- Let's do another example with $s(x) = \frac{1}{1+x^2}$. We'll compare the 10th-order polynomial with a third-degree spline with four knots at -3, -1, 1, and 3.
  - Since the data lie between -5 and 5, the knots are evenly spaced
- To construct the spline, let's first write out the equation:

$$\begin{aligned} s(x) \quad = \quad & \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \gamma_1 (x-(-3))^3 \mathbf{1}(x \geq -3) + \gamma_2 (x-(-1))^3 \mathbf{1}(x \geq -1) \\ & + \gamma_3 (x-1)^3 \mathbf{1}(x \geq 1) + \gamma_4 (x-3)^3 \mathbf{1}(x \geq 3) + u \end{aligned}$$

- To code in R, let's create the knots at the original and new data

```
k1<-ifelse(x>(-3),(x-(-3))^3,0)
k2<-ifelse(x>(-1),(x-(-1))^3,0)
k3<-ifelse(x>(1),(x-1)^3,0)
k4<-ifelse(x>(3),(x-(3))^3,0)
xnew$k1<-ifelse(xnew$x>(-3),(xnew$x-(-3))^3,0)
xnew$k2<-ifelse(xnew$x>(-1),(xnew$x-(-1))^3,0)
xnew$k3<-ifelse(xnew$x>(1),(xnew$x-1)^3,0)
xnew$k4<-ifelse(xnew$x>(3),(xnew$x-(3))^3,0)
```

- Then plot

```
plot(ynew~xnew$x,ylim=c(-0.25,1))
lines(predict(lm(y~x+x2+x3+x4+x5+x6+x7+x8+x9+x10),xnew)~xnew$x,col=4,lwd=2)
lines(predict(lm(y~x+x2+x3+k1+k2+k3+k4),xnew)~xnew$x,col=1,lwd=2)
```

R Example: GAM package in R

- There are a number of non-parametric econometrics packages in R
  - library "gam" is the easiest to use
  - library "mvcv" has more bells and whistles - check it out on your own as you wish.
- We will estimate (again) labor force participation with gam, as a function of age:

```
gamresults<-gam(nilf ~s(age,4), data=subd)
summary(gamresults)
plot(gamresults,se=TRUE,rug=FALSE,terms="s")
```

- In the first line, "s(age,4)" specifies a smooth function of the variable "age" with a smoothing parameter of 4.
  - This smoothing parameter goes into a complicated procedure called "backfitting", but the entire procedure is based on third-order splines.
- "s(age,1)" would yield a linear regression.
- The dependent variable is always demeaned to zero before estimation. So, $\mathbb{E}(s(age,)) = 0$. This is useful for inference.

R Example: GAM package in R

- Now we add-in education, which is a factor variable.

```
gamresults<-gam(nilf ~s(age,4)+educ, data=subd)
summary(gamresults)
par(mfrow=c(1,2))
plot(gamresults,se=TRUE,rug=FALSE,terms="s")
abline(v=0)
abline(h=0)
plot(gamresults,se=TRUE,rug=FALSE,terms="educ")
abline(v=0)
abline(h=0)
```

- The use of the function "abline" places a horizontal and vertical intercept at zero, with the former being the benchmark for being different from the sample average.
  - That is, if the two standard deviation confidence bands do not include zero, we reject zero as a hypothesized value at that point.
  - Since $\mathbb{E}(s(age,)) = 0$, we conclude that the estimate at that point is significantly different from the sample average.
- GLM restrictions (eg. families, links) can be used with gam.

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 of 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
  - "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))
}
```

Economics 217 - "Modern" Data Science

- Topics covered in this lecture
  - K-nearest neighbors
  - Decision Trees
- There is no reading for these lectures. Just notes. However, I have copied a number of online websites that may help to the course schedule.
- I hired a PhD student who works in this area to prepare some extra notes on the website, which provide more examples for those who are interested.

Modern data science

- In 216 and 217, we have (mostly) evaluated empirical relationships using parametric models
    - Parametric models almost surely have some form of model mis-specification, but are helpful in that the techniques to analyze the models are well sussed-out and interpretations of the model are fairly straightforward (ie. take a derivative)
    - In new data science lingo, we are "supervising" the data with a model
- In this last few lectures, we have been more flexible with our modeling choices
    - More flexible models that are non-parametric
    - Resampling procedures to conduct inference and choose smoothing parameters
- Practically, much of the new data science literature, learning and otherwise, isn't all that different from what we're doing already.
    - The main difference is the choice of non-parametric model, and the goal is to improve prediction.

Modern data science (cont.)

- Whether you adopt new techniques or old techniques is usually a function of your research objective

- In economics, we often wish to understand the mechanisms behind behaviors, as opposed to the collection of attributes that lead to behaviors.

  - Example: Knowing that graduates from Harvard are more likely to own a new house than graduates of Cabrillo college might be interesting from a marketing perspective, but it tells us nothing of *why* this is the case

  - If we are constructing policy, we want to know why. That is the big difference between modern data science and econometrics as I see it (even though in principle the techniques are very similar)

  - To be sure, the techniques can be complementary.

- In this lecture, we will study two techniques:

  - **K-Nearest Neighbors**: Similar people do similar things.

  - **Decision Trees**: Individuals adopt a heuristic to make choices.

K-Nearest Neighbors

- K-Nearest Neighbors is extremely similar to the Nadaraya-Watson binned estimator
  - In NW, we take a bandwith of h on either side of a given x, and average the behaviour within the region to generate a prediction for y.
  - This technique can be extended to more than one dimension of x by using a measure of "Euclidean Distance".
- K-Nearest Neighbors (KNN) also measures average (or modal) behavior around a particular point.
  - Instead of a fixed distance of h around a particular x, KNN, uses k nearest neighboring observations to measure behavior.
- The key inputs to a basic KNN model
  - The choice of k (obviously)
  - The distance function
  - The outcome variable (eg. unemployment)
  - The input variables (which will be used to determine who is nearest)

K-Nearest Neighbors - Distance

- Euclidean Distance is a common measure of distance.
- In P dimensions, Euclidean distance of two observations, $\mathbf{x}_i = \left(x_{i1}, x_{i2}, \cdots, x_{ip}\right)$, and $\mathbf{x}_j = \left(x_{j1}, x_{j2}, \cdots, x_{jp}\right)$, is:

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sqrt{\sum_{l=1}^{p} \left(x_{il} - x_{jl}\right)^2}$$

- In one dimension, it is just absolute distance
- In two dimensions, this is basically the Pythagorean theorem.
- Other distance functions exist, but we'll just use Euclidean distance

K-Nearest Neighbors - Outcomes

- In the NW estimator, we averaged outcomes within the bandwidth.
  - Eg. Average real wage
  - Averages might be weighted by a kernel function
- In data science jargon, outcomes can also be "classifications"
  - Unemployed, part-time, employed, out of workforce
  - Classifications are hard to average
- For KNN, the prediction is:
  - Average value if outcome is numeric
  - The modal value if outcome is a classification (this is called "majority rule" in data science lingo)
- Similar to h being chosen by cross-validation in NW, k can be chosen by a similar technique for KNN.

R example: K-Nearest Neighbors

- Load the necessary libraries

```
library(caret)
library(foreign)
```

- Load and clean data

```
d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")
d<-subset(d,is.na(nilf)==FALSE)
d<-subset(d,is.na(educ)==FALSE)
d<-subset(d,is.na(age)==FALSE)
d<-subset(d,is.na(female)==FALSE)
```

- Construct the "training" and "testing" samples:

```
subtrain<-subset(d,year==2013&state=="CA")
subtest<-subset(d,year==2013&state!="CA")
```

- Run your model:

```
model.knn <- train(nilf ~age+educ+female, data = subtrain, method =
"knn")
```

R example: K-Nearest Neighbors

- After the regression, check accuracy using the training sample

```
val.pred <- predict(model.knn, subtrain)
```

- Calculate the share of predictions that match the actual values in the training sample

```
val.acc <- sum(val.pred ==
subtrain$nilf,na.rm=TRUE)/length(subtrain$nilf)
print(val.acc)
```

- Now to the same with the testing sample

```
pred <- predict(model.knn2, subtest)
accuracy <- sum(pred == subtest$nilf2,na.rm=TRUE)/length(subtest$nilf2)
print(accuracy)
```

- By comparing "acc" and "accuracy", we can compare how well the model does within sample and out of sample.

R example: K-Nearest Neighbors (cont.)

- The results look pretty poor. So, let's redefine our outcome variable as non-numeric

```
d$nilf2<-ifelse(d$nilf==1,"Out of Labor Force", "In Labor Force")
```

- Re-construct the "training" and "testing" samples:

```
subtrain<-subset(d,year==2013&state=="CA")
subtest<-subset(d,year==2013&state!="CA")
```

- Run the model:

```
model.knn <- train(nilf2 ~age+educ+female, data = subtrain, method =
"knn")
```

- And compare accuracy:

```
obsr<-subtrain$nilf2
val.pred <- predict(model.knn2, subtrain)
val.acc <- sum(val.pred == obsr,na.rm=TRUE)/length(obsr)
pred <- predict(model.knn2, subtest)
accuracy <- sum(pred == obsr,na.rm=TRUE)/length(obsr)
print(val.acc)
print(accuracy)
```

Decision Trees

- Decision Trees are a form of classification, and map nicely into a "heuristic" approach of decision making by individuals.
- An example: Buying a car
  - Car or Truck
    - Domestic or Foreign
- Decision Trees can also be used to categorize outcomes by defining thresholds
- Suppose the outcome is "employed"
  - White or Non-White
    - Education greater than X, or less than X
- These are very complex models, but they general require (1) an order of "sub-trees", (2) splitting variables and (3) splitting points.
  - All three components can be chosen by cross-validation.
- The technique that is used for estimation is called recursive partitioning.

R example: Decision Trees

- Let's evaluate employment outcomes as a function of education and demographics.
- Load the required libraries

```
library(rpart)
library(foreign)
library(rattle)
```

- Reload and prepare outcome variable

```
d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")
d<-subset(d,is.na(educ)==FALSE)
d<-subset(d,is.na(age)==FALSE)
d<-subset(d,is.na(female)==FALSE)
```

- Take "lfstat", which is labor force status, and create a dichotomous variable for whether or not the respondent is employed

```
d$lfstat2<-ifelse(d$lfstat=="Employed","Employed","Not Employed")
```

R example: Decision Trees (cont)

- Just like with the KNN, create the training and testing samples

```
subtrain<-subset(d,year==2013&state=="CA")
subtest<-subset(d,year==2013&state!="CA")
```

- Run the classification tree

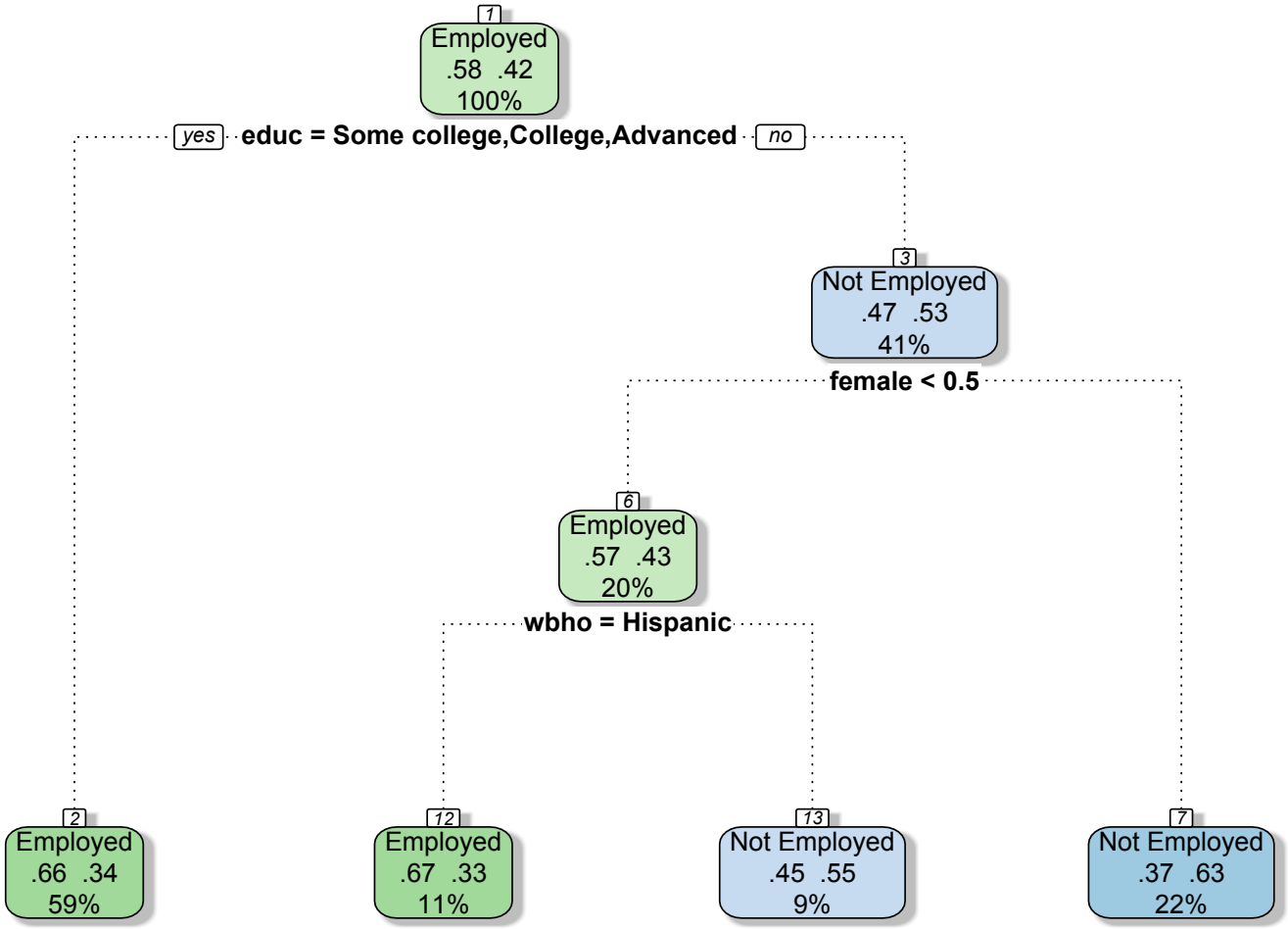
```
tree <- rpart(lfstat2 ~educ+wbho+female,data = subtrain, method =
"class")
```

- Use "fancyRpartPlot" from the library "rattle" to visualize the results

```
fancyRpartPlot(tree, main = "Labor Force Status on Education, Race, and
Gender")
```

- There are a host of other plotting functions

Labor Force Status on Education, Race, and Gender



Rattle 2016-Feb-10 10:22:50 acspearot

R example: Decision Trees (cont)

- Calculate accuracy for the training model

```
obsr<-subtest$lfstat2
outcomes <- predict(tree, subtest)
preds <- ifelse(outcomes[,1] >= .5, "Employed", "Not Employed")
accuracy <- round(sum(preds == obsr,na.rm=TRUE)/length(preds))
print(accuracy)
```

- And compare with the testing model

```
obsr<-subtrain$lfstat2
outcomes <- predict(tree, subtrain)
preds <- ifelse(outcomes[,1] >= .5, "Employed", "Not Employed")
accuracy <- round(sum(preds == obsr,na.rm=TRUE)/length(preds), digits =
4)
print(accuracy)
```

Time Series Econometrics

- Typical data that has been covered up to this point:
  - Cross-sectional data - many individuals at one point in time
  - Panel data - many individuals sampled repeatedly
- Time series data consists of one individual observed in multiple periods
- Why would we step back from panel data to study time series data?
  - Some "individuals" are singular (eg. global temperatures)
  - Decomposing an individual time-series can be helpful for analyses
- One way to look at time-series is the following
  - There are a variety of models that can explain the data. We wish to find the model of best fit without going overboard.
  - For example: Are observables correlated over time, or are unobservables correlated over time (or both)?

ARIMA models

- ARIMA models are a general form of time-series model that incorporate three features:
- **AR:** Autoregressive
  - Outcomes are explicitly correlated over-time
  - GDP this year depends on GDP last year
- **I:** Integrated
  - Not stationary series - drift in average of the outcome
- **MA:** Moving Average
  - The average of an outcome in a current period is a weighted sum of past noise.

Stationarity

- A time series is considered *covariance stationary* if
  - mean reversion around a constant long-run mean
  - finite variance that is time invariant
  - a theoretical correlogram that decreases in lag length (will explain this later)
- A key idea here is that shocks to a stationary system will only be temporary.
  - Without these assumptions, it's very hard to characterize the properties of a variable, as we'll see.
- For these slides the key assumption for our derivations is that all unobserved shocks are Gaussian white noise
  - Normally distributed with **mean zero** and **constant variance**.

The AR(1) process

- The basic AR(1) process is written as follows:

$$Y_t = \phi Y_{t-1} + u_t$$

  - Y_t is the observed time-series variable at time t
  - u_t is the unobserved shock at time t - gaussian white noise
  - ϕ determines how much of Y_t is based on past values

- The key (necessary) condition for stationarity is $|\phi| < 1$. To see this, solve for the variance of Y_t and simplify:

$$\begin{aligned} Var(Y_t) &= Var(\phi Y_{t-1}) + Var(u_t) \\ &= \phi^2 Var(Y_{t-1}) + Var(u_t) \\ &= \phi^2 Var(Y_t) + Var(u_t) \\ \Rightarrow \quad Var(Y_t) &= \frac{Var(u_t)}{(1-\phi^2)} \end{aligned}$$

- Variance of Y_t explodes as ϕ^2 approaches 1.

Code to generate an AR(1) process

- First, create a function to generate an AR1 process

```
AR1<-function(n,phi){
  es<-rnorm(n)
  Y<-rep(0,n)
  for(i in 2:length(Y)){
     Y[i]<-phi*Y[i-1]+es[i]
  }
  return(Y)
}
```

- Plot different time series of length 100 with different ϕ parameters.

```
par(mfrow=c(2,3))
plot(AR1(100,0.1),type='l',main="phi=0.1",xlab='t',ylab="Y")
plot(AR1(100,0.5),type='l',main="phi=0.5",xlab='t',ylab="Y")
plot(AR1(100,0.9),type='l',main="phi=0.9",xlab='t',ylab="Y")
plot(AR1(100,1),type='l',main="phi=1",xlab='t',ylab="Y")
plot(AR1(100,1.5),type='l',main="phi=1.5",xlab='t',ylab="Y")
plot(AR1(100,2),type='l',main="phi=2",xlab='t',ylab="Y")
```

- Try this repeatedly and see what happens to the non stationary plots

The AR(1) process and its properties

- For the stationary AR(1) process with Gaussian white noise for u_t it is straightforward to show that:

$$E(Y_t) = 0$$

- A key property is how the correlation between period t and $t-k$ values decay as k increases. The correlation that we seek is the following:

$$ACF = Cor(Y_t, Y_{t-k}) = \frac{Cov(Y_t, Y_{t-k})}{\sqrt{Var(Y_t)}\sqrt{Var(Y_{t-k})}}$$

- This metric is often called a "correlogram" when plotted against k.

- How do we refine the denominator?

- Since process is stationary, $Var(Y_t) = Var(Y_{t-k})$. So, we have:

$$ACF = \frac{Cov(Y_t, Y_{t-k})}{Var(Y_t)}$$

The AR(1) process and its properties (cont.)

- To simplify the numerator, note that:

$$Cov(Y_t, Y_{t-k}) = E\Big[\,(Y_t - E(Y_t))\,(Y_{t-k} - E(Y_{t-k}))\,\Big]$$

- Expanding:

$$Cov(Y_t, Y_{t-k}) = E\Big[Y_t Y_{t-k} - E(Y_t)Y_{t-k} - E(Y_t)E(Y_{t-k}) + Y_t E(Y_{t-k})\Big]$$

- Noting that $E(Y_t) = E(Y_{t-1}) = 0$, we have:

$$Cov(Y_t, Y_{t-k}) = E\Big[Y_t Y_{t-k}\Big]$$

- Now, we need to solve for Y_t precisely determine the covariance. Recursively using the AR(1) equation k times, we have:

$$\begin{aligned}
Y_t &= \phi Y_{t-1} + u_t \\
&= \phi\left(\phi Y_{t-2} + u_{t-1}\right) + u_t \\
&= \phi\left(\phi\left(\phi Y_{t-3} + u_{t-2}\right) + u_{t-1}\right) + u_t \\
&\vdots \\
&= \phi^k Y_{t-k} + u_t + \phi u_{t-1} + \cdots + \phi^{k-2} u_{t-k+2} + \phi^{k-1} u_{t-k+1}
\end{aligned}$$

The AR(1) process and its properties (cont.)

- Plugging in to $Cov(Y_t, Y_{t-k})$:

$$\begin{aligned} Cor(Y_t, Y_{t-k}) &= \phi^k E\left[Y_{t-k}Y_{t-k}\right] \\ &\quad + E\left[u_t Y_{t-k}\right] \\ &\quad + \phi E\left[u_{t-1} Y_{t-k}\right] \\ &\quad \vdots \\ &\quad + \phi^{k-2} E\left[u_{t-k+2} Y_{t-k}\right] \\ &\quad + \phi^{k-1} E\left[u_{t-k+1} Y_{t-k}\right] \end{aligned}$$

- Since future shocks do not determine past Y's, we are left with:

$$Cor(Y_t, Y_{t-k}) = \phi^k E\left[Y_{t-k}Y_{t-k}\right]$$

- Finally, since $E\left[Y_{t-k}\right] = 0$, we have:

$$\begin{aligned} Cor(Y_t, Y_{t-k}) &= \phi^k E\left[\left(Y_{t-k} - E[Y_{t-k}]\right)\left(Y_{t-k} - E[Y_{t-k}]\right)\right] \\ &= \phi^k Var(Y_t) \end{aligned}$$

The AR(1) process and its properties (cont.)

- Overall, we have:

$$ACF = \frac{\phi^k Var(Y_t)}{Var(Y_t)} = \phi^k$$

- Since ϕ is between zero and one, ACF asymptotes to zero with higher k.

- Let's now construct some AR(1) processes in R that are stationary and otherwise and construct their correlogram.

ACF of AR(1) processes in R

- The 'acf' function in R simply constructs and plots ACF functions

```
acf(series,lag.max=10,type="correlation")
```

  - "series" is your data
  - "lag.max=10" is the number of lags you want on the plot
  - "type="correlation"" indicates that you want a correllogram

- Use our AR1 function from the previous example within this plotting function:

```
par(mfrow=c(2,3))
acf(AR1(100,0.1),lag.max=10,type="correlation")
acf(AR1(100,0.5),lag.max=10,type="correlation")
acf(AR1(100,0.9),lag.max=10,type="correlation")
acf(AR1(100,1),lag.max=10,type="correlation")
acf(AR1(100,1.5),lag.max=10,type="correlation")
acf(AR1(100,2),lag.max=10,type="correlation")
```

- This plotting function will also indicate 2 standard deviation confidence intervals for a simple test of significant lags.

The AR(p) process

- The basic AR(p) process is simply an expanded AR(1) to include p lags of the dependent variable

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_{p-1} Y_{t-p+1} + \phi_p Y_{t-p} + u_t$$

- We can program an AR(p) process in R quite easily

```
ARp<-function(n,phi){

  p<-length(phi)
  es<-rnorm(n+p)
  Y<-rep(0,n+p)

  for(i in (p+1):length(Y)){
      Y[i]<-t(phi)%*%Y[(i-p):(i-1)]+es[i]
  }
  Y<-Y[-(1:p)]
  return(Y)
}
```

The Moving Average Models

- MA(1): Current values are a function of current white noise and some function of the last period's white noise

$$Y_t = u_t + \theta u_{t-1}$$

- MA(q): Current values are a function of current white noise and some function prior period noise

$$\begin{aligned} Y_t &= u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_{q-1} u_{t-q+1} + \theta_q u_{t-q} \\ &= u_t + \sum_{j=1}^{q} \theta_j u_{t-j} \end{aligned}$$

- MA(1) and AR(p) models are similar in the limit, which is a complication for estimation. We now take a look at this.

AR or MA? Theoretical identification issues

- Rearrange MA(1) for u_t.

$$u_t = Y_t - \theta u_{t-1}$$

- For u_{t-1} this is:

$$u_{t-1} = Y_{t-1} - \theta u_{t-2}$$

- Substitute repeatedly into $Y_t = u_t + \theta u_{t-1}$ for the lag error:

$$\begin{aligned} Y_t &= u_t + \theta\left(Y_{t-1} - \theta u_{t-2}\right) \\ &= u_t + \theta\left(Y_{t-1} - \theta\left(Y_{t-2} - \theta u_{t-3}\right)\right) \\ &\vdots \\ &= u_t + \theta Y_{t-1} + \theta^2 Y_{t-2} + \cdots + \theta^q Y_{t-q} + \cdots \end{aligned}$$

- But, an AR(p) looks like

$$Y_t = u_t + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_{p-1} Y_{t-p+1} + \phi_p Y_{t-p}$$

- Understanding the subtle differences is central to identification.

MA(1) correlogram

- We once again turn to evaluating the correlation of the observed value in time t with the observed value at time $t-k$.
- For AR(1), the decay of this correlation was log-linear with lag length k: ϕ^k
- To derive for MA(1), we note that:

$$ACF = \frac{Cov(Y_t, Y_{t-k})}{Var(Y_t)}$$

- The variance of Y_t is straightforward

$$\begin{aligned} Var(Y_t) &= Var(u_t) + Var(\theta u_{t-1}) \\ &= Var(u_t) + \theta^2 Var(u_t) \\ &= Var(u_t)\left(1 + \theta^2\right) \end{aligned}$$

- $Cov(Y_t, Y_{t-k})$ depends on the length of the lag.

MA(1) correlogram (cont.)

- For $k = 1$

$$Cov(Y_t, Y_{t-1}) = E\Big[(Y_t - E(Y_t))(Y_{t-1} - E(Y_{t-1})) \Big]$$

- Since our white noise processes are mean zero, so are $E(Y_t)$'s

$$Cov(Y_t, Y_{t-1}) = E\Big[Y_t Y_{t-1} \Big]$$

- Substituting using the MA(1) equation

$$\begin{aligned} Cov(Y_t, Y_{t-1}) &= E\Big[(u_t + \theta u_{t-1})(u_{t-1} + \theta u_{t-2}) \Big] \\ &= E\Big[u_t u_{t-1} + \theta u_{t-1} u_{t-1} + \theta u_t u_{t-2} + \theta^2 u_{t-1} u_{t-2} \Big] \end{aligned}$$

- Distributing the expectation, $E[u_t u_{t-k}] = 0 \ \ \forall \ \ k \neq 0$ since shocks are not serially correlated. Hence:

$$\begin{aligned} Cov(Y_t, Y_{t-1}) &= E\Big[\theta u_{t-1} u_{t-1} \Big] \\ &= \theta E\Big[u_{t-1} u_{t-1} \Big] \\ &= \theta Var(u_t) \end{aligned}$$

MA(1) correlogram (cont.)

- For $k > 1$

$$Cov(Y_t, Y_{t-k}) = E\Big[(Y_t - E(Y_t))(Y_{t-k} - E(Y_{t-k})) \Big] = E\Big[Y_t Y_{t-k} \Big]$$

- Substituting using the MA(1) equation

$$\begin{aligned} Cov(Y_t, Y_{t-k}) &= E\Big[(u_t + \theta u_{t-1})(u_{t-k} + \theta u_{t-k-1}) \Big] \\ &= E\Big[u_t u_{t-k} + \theta u_{t-1} u_{t-k} + \theta u_t u_{t-k-1} + \theta^2 u_{t-1} u_{t-k-1} \Big] \end{aligned}$$

- Since $E[u_t u_{t-k}] = 0 \ \ \forall \ \ k \neq 0$, we have that

$$Cov(Y_t, Y_{t-k}) = 0$$

- Overall, the ACF for the MA(1) process is written as:

$$\begin{aligned} ACF &= \frac{\theta^2}{(1+\theta^2)} \quad if \ \ k = 1 \\ &= 0 \qquad\quad if \ \ k > 1 \end{aligned}$$

- For AR processes, the correlations persist. For MA they vanish sharply after some period determined by the lag length of errors.

MA(p) process in R

- MA(p) process is similar in coding to the AR(p) process.

```
MAp<-function(n,theta){
  d<-length(theta)
  es<-rnorm(n+d)
  Y<-rep(0,n+d)
  for(i in (d+1):length(Y)){
    Y[i]<-es[i]+t(theta)%*%es[(i-d):(i-1)]
  }
  Y<-Y[-(1:d)]
  return(Y)
 }
```

- Then, generate a 3X2 correlograms similar to before

```
par(mfrow=c(2,3))
acf(MAp(100,0.1),lag.max=10,type="correlation")
acf(MAp(100,0.5),lag.max=10,type="correlation")
acf(MAp(100,0.9),lag.max=10,type="correlation")
acf(MAp(100,1),lag.max=10,type="correlation")
acf(MAp(100,1.5),lag.max=10,type="correlation")
acf(MAp(100,2),lag.max=10,type="correlation")
```

ARMA(p,q) and ARIMA(p,d,q)

- Not surprisingly, the ARMA model is the combination of AR(p) and MA(q) models

$$Y_t = u_t + \sum_{j=1}^{q} \theta_j u_{t-j} + \sum_{j=1}^{p} \phi_j Y_{t-j}$$

- ARIMA(p,d,q) models incorporate an "integrated" term, which is a trend in the average over time.

- Assuming δ is constant, the following model is integrated of order 1:

$$Y_t = \delta + u_t + \sum_{j=1}^{q} \theta_j u_{t-j} + \sum_{j=1}^{p} \phi_j Y_{t-j}$$

- To make stationary, take differences (using the Δ operator).

$$\Delta Y_t = \Delta u_t + \sum_{j=1}^{q} \theta_j \Delta u_{t-j} + \sum_{j=1}^{p} \phi_j \Delta Y_{t-j}$$

- the 'd' in ARIMA(p,d,q) is the number of times the data must be differenced to make stationary.

Estimating ARIMA models in R

- To estimate ARIMA(p,d,q) models in R, let's first load data with the "quantmod" package

```
install.packages("quantmod")
library(quantmod)
```

- With quantmod, you can download stock price information from Yahoo finance.
- For example, downloading 7 years of data for google (before the stock split)

```
getSymbols("GOOG",from="2007-01-01",to="2014-01-01")
prices<-GOOG$GOOG.Open
rets <- dailyReturn(GOOG)
```

- Use str(GOOG) too see all the information after using getSymbols

Estimating ARIMA models in R

- Use the package "forecast" to provide the optimal ARIMA model

```
install.packages("forecast")
library(forecast)
```

- The function "auto.arima" gives us the optimal ARIMA model
  - Tries to find the best fit without over parameterizing (eg. Penalized log-likelihood)
- Syntax is simple:

```
auto.arima(prices)
auto.arima(rets)
```

- The function "forecast.arima" provides predictions with forecast errors.

```
google.arima<-auto.arima(prices)
forecast.Arima(google.arima,40,level=c(80,95)),
plot(google.forecast)
```

Modeling Variance - ARCH and GARCH

- Volatility is a key concept in finance
  - Calm vs. Turbulent periods of returns
  - Crucial for modeling and accounting for risk over a particular investing time horizon.
- Heteroskedastic errors are crucial for accounting for volatility.
  - Unlike the first lecture, the variance of white noise will not be constant over time
  - Obviously complicates many of our derivations.

ARCH models

- ARCH stands for "Autoregressive Conditional Heteroskedasticity"
- Consider the following simply time series model

$$Y_t = \beta_0 + \mathbf{X_t}\beta + u_t$$

  - outcome variable Y_t
  - vector of explanatory variables $\mathbf{X_t}$.
  - u_t again is the error term.

- For a ARCH(1), process, we assume that

$$\begin{aligned} u_t | \Omega_t &\sim N(0, h_t) \\ where \;\; h_t &= \gamma_0 + \gamma_1 u_{t-1}^2 \end{aligned}$$

- For ARCH(1), the variance of the error term depends on a constant, γ_0, and a function of the past squared error, $\gamma_1 u_{t-1}^2$.

ARCH (p) and testing

- The ARCH(p) model can be characterized as:

$$\begin{aligned} Y_t &= \beta_0 + \mathbf{X_t}\beta + u_t \\ where \;\; u_t | \Omega_t &\sim N(0, h_t) \\ h_t &= \gamma_0 + \sum_{j=1}^{p} \gamma_j u_{t-j}^2 \end{aligned}$$

- Testing these models is potentially very simple:
  - $Y_t = \beta_0 + \mathbf{X_t}\beta + u_t$
  - Collect residuals, square them
  - Estimate $u_t^2 = \gamma_0 + \sum_{j=1}^{p} \gamma_j u_{t-j}^2 + w_t$
  - Evaluate parameters, conduct exclusion test

GARCH(p,q)

- The ARCH(p) in many ways is more of a moving average than autoregressive
  - There is no explicit persistence in h_t - just some lagged function of past observables.
- "Generalized Autoregressive Conditional Heteroskedasticity" allows for dependences on past unobservables, as well as persistence in the variance
- The GARCH(p,q) model is written as:

$$\begin{aligned} Y_t &= \beta_0 + \mathbf{X_t}\beta + u_t \\ \textit{where}\ \ u_t \mid \Omega_t &\sim N(0, h_t) \\ h_t &= \gamma_0 + \sum_{j=1}^{p} \delta_j h_{t-j} + \sum_{j=1}^{q} \gamma_j u_{t-j}^2 \end{aligned}$$

- $\mathbf{X_t}$ could include lags of Y_t, or residuals like in the ARIMA framework.

Estimating ARCH(p) and GARCH(p,q) models

- In R, there are many functions to estimate ARCH and GARCH models.
- Unsurprisingly, the package "tseries" is one that contains many functions for time series analysis
- To demonstrate, download S&P500 data for 2001 to 2015

```
library(quantmod)
getSymbols('ĜSPC',from='2001-01-01',to='2015-01-01')
rets = dailyReturn(GSPC)
plot(rets)
```

- To make ourselves sick, let's plot the price series and daily returns.

```
par(mfrow=c(2,1))
plot(GSPC$GSPC.Open) plot(rets)
```

- Install and load the package "tseries" `install.packages("tseries")`

```
library(tseries)
```

Estimating ARCH(p) and GARCH(p,q) models

- The function "garch" estimates both ARCH and GARCH models, assuming no covariates in the regression equation
  - Thus, this is pure variance estimation
- Syntax is as follows

```
garch(data,order=c(p,q))
```

- p is the GARCH component, and q is the ARCH component
- Using the data we downloaded, estimate a ARCH(1) model and summarize

```
sp500.g=garch(rets,order=c(0,1))
summary(sp500.g)
```

- Can predict bounds on values using the predict function

```
u=predict(sp500.g)
```

- This vector has two elements - upper and lower bounds

Estimating ARCH(p) and GARCH(p,q) models

- Can plot the data and estimated bounds using the following code

```
rets_upper<-rets
rets_upper$daily.returns<-u[,1]
rets_lower<-rets
rets_lower$daily.returns<-u[,2]
par(mfrow=c(2,1))
plot(rets, type="l", xlab="time", ylab="daily change",
main="SP500 index 2001-2015")
lines(rets_upper,col="red", lty="dashed",lwd=1.5)
lines(rets_lower,col="red", lty="dashed",lwd=1.5)
```

- What do you notice about this plot, and how might a GARCH model improve the simple ARCH framework?

Estimating ARCH(p) and GARCH(p,q) models

- Can plot the data and estimated bounds using the following code

```
sp500.g2=garch(rets,order=c(1,1))
u2=predict(sp500.g2)
rets_upper2<-rets
rets_upper2$daily.returns<-u2[,1]
rets_lower2<-rets
rets_lower2$daily.returns<-u2[,2]
```

- Finally, plot the old bounds (in black) against the new bounds (blue)

```
plot(rets_upper, type="l", xlab="time", ylab="daily change",
main="SP500 index 2001-2015",ylim=c(-.07,.07))
lines(rets_lower)
lines(rets_upper2,col="blue", lty="dashed",lwd=1)
lines(rets_lower2,col="blue", lty="dashed",lwd=1)
```

Estimating ARCH(p) and GARCH(p,q) models

- The package fGarch contains more bells and whistles
  - GARCH error estimation as above
  - ARMA effects in the original regression equation
- Install the package and load the library

```
install.packages("fGarch")
library(fGarch)
```

- Syntax for GARCH estimation with ARMA components

```
garchFit(~arma(ar,ma)+garch(p,q),data,trace=FALSE)
```

- Let's run four different models to see how they look

```
garch1<-garchFit(~garch(1,1),data=rets,trace=FALSE)
garch2<-garchFit(~arma(1,1)+garch(2,1),data=rets,trace=FALSE)
garch3<-garchFit(~arma(2,1)+garch(2,1),data=rets,trace=FALSE)
garch4<-garchFit(~arma(2,2)+garch(2,2),data=rets,trace=FALSE)
```

Estimating ARCH(p) and GARCH(p,q) models

- Summarize works similarly to other regressions

```
summary(garch4)
```

- Predict generates predictions in 'n.ahead' future periods, as well as plots the last 'nx' observations

```
par(mfrow=c(2,2))
predict(garch1,n.ahead=10,plot=TRUE,nx=20)
predict(garch2,n.ahead=10,plot=TRUE,nx=20)
predict(garch3,n.ahead=10,plot=TRUE,nx=20)
predict(garch4,n.ahead=10,plot=TRUE,nx=20)
```

Estimating ARCH(p) and GARCH(p,q) models

- fGarch also has some cool ploting functions. The main feature is interactive

```
plot(garch4)
```

- Or, you can choose the difference elements using "which"

```
par(mfrow=c(2,2))
plot(garch4,which=1)
plot(garch4,which=3)
plot(garch4,which=4)
plot(garch4,which=10)
```

GARCH-M Models

- In many models of finance, the expect return of an asset is some function of characteristics and then risk.
  - Risk is often viewed through the lens of standard deviation
- The GARCH-M Model allows for this explicit dependence:

$$\begin{aligned} Y_t &= \beta_0 + \mathbf{X_t}\beta + \theta\sqrt{h_t} + u_t \\ \textit{where}\ \ u_t|\Omega_t &\sim N(0, h_t) \\ h_t &= \gamma_0 + \sum_{j=1}^{q} \delta_j h_{t-q}^2 + \sum_{j=1}^{p} \gamma_j u_{t-j}^2 \end{aligned}$$

- Though $\sqrt{h_t}$ may look weird, I like it since it is on the same scale as the dependent variable.
  - Some prefer including variance h_t instead, so it's best to know the results from both.
- Finally, one can also add a set of regressors into the error equation as needed.

VARs and Granger Causality

- Economic theories usually consist of a set of endogenous variables that are determined, in equilibrium, by exogenous parameters
- We are almost always interested in these endogenous outcomes, though sometimes we are interested in fundamental parameters.
  - Eg. Demand and supply elasticities, capacity constraints.
- However, it is extremely difficult to determine causality
  - You know this already since 216 was focused on much of these identification problems
- Time series has a number of techniques that side-step or re-cast issues of causation
  - **VAR**: Vector autoregression
  - Granger Causality

Vector autoregression (VAR)

- VARs are basically systems of equations with outcome variables that depend on other outcome variables
- Consider the following simply time series model

$$\begin{aligned} y_t &= \beta_{10} - \beta_{12} x_t + \gamma_{11} y_{t-1} + \gamma_{12} x_{t-1} + u_{yt} \\ x_t &= \beta_{20} - \beta_{21} y_t + \gamma_{21} y_{t-1} + \gamma_{22} x_{t-1} + u_{xt} \end{aligned}$$

- VARs are meant to get around the obvious concerns in the above system of equations. What causes what?
- Instead, we again focus on predicting endogenous variables as a function of lags. The goal is exploiting useful information and making predictions.

Vector autoregression (VAR)

- VARs treat the x's and y's the same, and view the problem was one of forecasting as opposed to empirical identification.

- Again, consider the model:

$$\begin{aligned} y_t &= \beta_{10} - \beta_{12}x_t + \gamma_{11}y_{t-1} + \gamma_{12}x_{t-1} + u_{yt} \\ x_t &= \beta_{20} - \beta_{21}y_t + \gamma_{21}y_{t-1} + \gamma_{22}x_{t-1} + u_{xt} \end{aligned}$$

- Solve for y_t and x_t on the LHS:

$$\begin{aligned} y_t + \beta_{12}x_t &= \beta_{10} + \gamma_{11}y_{t-1} + \gamma_{12}x_{t-1} + u_{yt} \\ \beta_{21}y_t + x_t &= \beta_{20} + \gamma_{21}y_{t-1} + \gamma_{22}x_{t-1} + u_{xt} \end{aligned}$$

- Arranging the system of equations in matrix notation:

$$\begin{bmatrix} 1 & \beta_{12} \\ \beta_{21} & 1 \end{bmatrix} \begin{bmatrix} y_t \\ x_t \end{bmatrix} = \begin{bmatrix} \beta_{10} \\ \beta_{20} \end{bmatrix} + \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} u_{yt} \\ u_{xt} \end{bmatrix}$$

- Invert:

$$\begin{bmatrix} y_t \\ x_t \end{bmatrix} = \begin{bmatrix} 1 & \beta_{12} \\ \beta_{21} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \beta_{10} \\ \beta_{20} \end{bmatrix} + \begin{bmatrix} 1 & \beta_{12} \\ \beta_{21} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} \widetilde{u}_{yt} \\ \widetilde{u}_{xt} \end{bmatrix}$$

Vector autoregression (VAR)

- Define the vector of constants as

$$\mathbf{B}_0 = \begin{bmatrix} 1 & \beta_{12} \\ \beta_{21} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \beta_{10} \\ \beta_{20} \end{bmatrix}$$

- Define the matrix of lag coefficients as:

$$\mathbf{B}_1 = \begin{bmatrix} 1 & \beta_{12} \\ \beta_{21} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}$$

- Solving for $\begin{bmatrix} y_t \\ x_t \end{bmatrix}$, we have the "Reduced form VAR":

$$\begin{bmatrix} y_t \\ x_t \end{bmatrix} = \mathbf{B}_0 + \mathbf{B}_1 \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} \widetilde{u}_{yt} \\ \widetilde{u}_{xt} \end{bmatrix}$$

- Current observables, y_t and x_t, are functions of lagged observables.
- Some points about this VAR:
  - For purposes of forecasting, this is as convenient as anything we've done thus far.
  - This VAR cannot recover the original parameters.

Estimating a VAR in R

- Estimating a VAR in R is as simple as running a linear regression, but there are also handy packages to do this:
- First, let's download the data we need

```
library(quantmod)
getSymbols('GOOG',from='2014-11-12',to='2015-11-12')
vol<-GOOG$GOOG.Volume
price<-GOOG$GOOG.Open
```

- Question: Do prices drive volume or volume drive prices?
- We first run a reduced for VAR by directly programming into OLS

```
n<-length(price)
regPrice<-lm(price[2:n]~price[1:(n-1)]+vol[1:(n-1)])
regVol<-lm(vol[2:n]~price[1:(n-1)]+vol[1:(n-1)])
summary(regPrice)
summary(regVol)
```

- Make sure the indexing is correct!

Estimating a VAR in R

- The package MSBVAR has a load of functions that help in time-series analysis and forecasting
- First, load the required library

```
library(MSBVAR)
```

- Initiate the data as a time-series object

```
y2<-ts(data.frame(price,volume))
```

- Run the VAR in reduced form

```
pricevol_var<-reduced.form.var(y2,p=1)
```

  - $p = 1$ specifies one period lag. Increase for bigger lag lengths.

- Summarize like our earlier versions

```
summary(pricevol_var)
```

Causality Tests

- Causality in time series is defined differently from applied micro (as in 216)
- In applied micro "style", we would usually approach a problem as follows:
  - Find an "instrument" for x_t
  - Use IV to find the precise estimate for β_{12}.
  - Other techniques could also be used (randomly allocate x_t's, use discontinuities if they exist, etc..)
- "Causality" in time series usually refers to the "information" contained within a time series for some variable.
- "Granger Causality"
  - Controling for the history of y's, the history of x's help predict y.
  - x "Granger Causes" y.
  - This is not the same as casual inference - omitted variables could still affect this conclusion.

Granger Causality Test

- Granger Causality Test is performed by the following three-step procedure (which is essentially a F-test)
- Step 1: Regress y on y lags without x lags (restricted model)

$$y_t = a_1 + \sum_{j=1}^{m} \gamma_j y_{t-j} + e_t$$

- Step 2: Add in x lags and regress again (unrestricted model)

$$y_t = a_1 + \sum_{i=1}^{n} \beta_i x_{t-i} + \sum_{j=1}^{m} \gamma_j y_{t-j} + e_t$$

- Step 3: Test null hypothesis that $\beta_i = 0 \ \forall i$ using a F-test

Granger Causality Test in R

- Running a Granger test in R is quite simple
- Using the same time series object as in the previous example:

```
granger.test(y2,p=2)
```

- If $p = 1$, we only have one lag. Do we need a Granger testing this case?

Sims Causality Test

- "Sims Causality"
  - Controling for lag y's and x's, do future x's predict current y's?
- Sims Causality tests for the effect of "leading terms"
- **Step 1:** Regress y on y lags and x lags (restricted model)

$$y_t = a_1 + \sum_{i=1}^{n} \beta_i x_{t-i} + \sum_{j=1}^{m} \gamma_j y_{t-j} + e_{1t}$$

- **Step 2:** Add in x leading terms $(t+\rho)$ and regress again (unrestricted model)

$$y_t = a_1 + \sum_{i=1}^{n} \beta_i x_{t-i} + \sum_{j=1}^{m} \gamma_j y_{t-j} + \sum_{\rho=1}^{m} \xi_j x_{t+\rho} + e_{1t}$$

- **Step 3:** Test null hypothesis that $\xi_j = 0 \ \forall i$ using a F-test
- If we reject this null hypothesis, then y causes x since the future cannot predict the present.

Time Series Lecture Module 2

- Topics in this lecture
  - Stationarity and Unit Roots
  - Spurious Regressions
  - Cointegration
  - Error-Correction Models

Stationarity and Unit Roots

- We need a precise test to distinguish between stationarity and non-stationarity
    - Mean is unknown and variance explodes for non-stationary time series
- Graphical techniques were not based on any precise statistical test
- In this set of slides, we'll discuss the "unit root", and how to identify it
    - Though a unit root has a precise definition, it basically summarizes when a autoregressive relationship is non-stationary
    - Generally, we take differences, or differences of differences, or differences of differences of....of differences of differences to purge an autoregressive relationship of non-stationary properties.

What is a unit-root?

- Consider the following AR(1) model

$$y_t = \phi_1 y_{t-1} + u_t$$

- Three possible cases for this AR(1) model:
  1. $|\phi_1| < 1$, and the series is stationary
  2. $\phi_1 = 1$, and the series has a unit root and is non-stationary
  3. $\phi_1 = -1$, and the series is non-stationary without a unit root
  4. $|\phi_1| > 1$, and the series is explosive

- To test for unit root, first subtract y_{t-1} from both sides.

$$\begin{aligned} y_t - y_{t-1} &= (\phi_1 - 1) y_{t-1} + u_t \\ \Delta y_t &= \gamma y_{t-1} + u_t \end{aligned}$$

- $H_0 : \gamma = 0$ indicates a unit root.
  - For stationarity, we reject in favor of $\gamma < 0$. (note this is a one-sided test)
  - In this simple form, this test is known as the "Dickey-Fuller Test".
  - Test statistics are not based on a t-distribution - Table in book, correct p-values given in R.

What is a unit-root? (graphically)

- Create 3 different AR(1) time series, at or near a unit root.

```
Nobs<-100
x<-AR1(Nobs,0.25)
y<-AR1(Nobs,1)
z<-AR1(Nobs,1.02)
```

- Plot the time series

```
par(mfrow=c(1,3))
plot(x,type='l',main="phi=0.25",xlab='t',ylab="Y")
plot(y,type='l',main="phi=1",xlab='t',ylab="Y")
plot(z,type='l',main="phi=1.02",xlab='t',ylab="Y")
```

- What are the features of these three plots?

Integrated series

- From the previous series...

$$\Delta y_t = \gamma y_{t-1} + u_t$$

- Again, this is stationary when $\gamma < 0$.
  - This type of series is called "integrated of order 0": I(0)
- If not stationary, take differences and test again. If Δy_t is stationary, this type of series is called "integrated of order 1"
- In general, a series is integrated order d if d differences are required to make stationary.
- In R, for our previous series, take differences and plot:

```
plot(diff(x,lag=1),type='l',main="phi=0.25",xlab='t',ylab="Y")
plot(diff(y,lag=1),type='l',main="phi=1",xlab='t',ylab="Y")
plot(diff(z,lag=1),type='l',main="phi=1.02",xlab='t',ylab="Y")
```

- Do the differenced series look more stationary?

Testing for unit roots manually

- In R, we need to regress the differences of a time series on initial values and test the coefficient.
- Using our original time series x:

```
summary(lm(diff(x,lag=1)~x[1:(N-1)]))
```

- x[1:(N-1)] is the vector of matched initial time periods.
- Do the same for the other series

```
summary(lm(diff(y,lag=1)~y[1:(N-1)]))
summary(lm(diff(z,lag=1)~z[1:(N-1)]))
```

- These regressions are only suggestive in significance. Must use the DF significance table from the book, or the R code that I will present in a few slides.

Testing for unit roots manually

- When series appear to be non-stationary, we need find out how many differences we need to take for it to be stationary.
- To begin, test for stationarity in the differenced data
- Formally, we are testing where ϕ lies relative to 1 in the following

$$\Delta y_t = \phi \Delta y_{t-1} + u_t$$

- Subtracting Δy_{t-1} from both sides

$$\begin{aligned} \Delta y_t - \Delta y_{t-1} &= \phi \Delta y_{t-1} - \Delta y_{t-1} + u_t \\ \Delta^2 y_t &= \gamma \Delta y_{t-1} + u_t \end{aligned}$$

- In R, for some series z:

```
d1<-z[2:(N-1)]-z[1:(N-2)]
d2<-z[3:N]-z[2:(N-1)]
summary(lm(I(d2-d1)~d1))
```

Stationarity tests in R

- In the package "tseries", adf.test(x,k=0) runs the standard DF test.

```
library(tseries)
adf.test(x,k=0)
adf.test(y,k=0)
adf.test(z,k=0)
```

- The null hypothesis is that there is a unit root, and the alternative is "stationary".
- The augmented DF test runs the following regression

$$\Delta y_t = \beta_0 + \alpha t + \gamma y_{t-1} + \sum_{i=1}^{k} \beta_k \Delta y_{t-k} + e_t$$

- k adjusts the lag length in the regression.

```
adf.test(z,k=0)
adf.test(z,k=1)
adf.test(z,k=2)
```

- Again, the null is that there is a unit root.

Why we care - Spurious Regressions

- Recall from our earlier example that when $\phi = 1$

$$y_t = y_{t-1} + u_{yt}$$

- If we run this process enough times what do we notice?
  - The series usually trends somewhere.
- Suppose we have an independently constructed series of the same from:

$$x_t = x_{t-1} + u_{xt}$$

- If we regress y_t on x_t, what happens?
- Since both series have a tendency to trend somewhere, there will appear to be a relationship between the two series most of the time.
- This is called a **spurious relationship**. We must therefore identify unit roots when regressing time series on one another to prevent this issue.

Why we care - Spurious Regressions

- Regressing two independently created series should not show a systematic relationship
- But, when there is a unit root in both series, there may be a spurious relationship.
  - This is bad for macro data, since as we know aggregate variables usually trend somewhere.
- Run this code repeatedly and see how many times you get an insignificant relationship between the two series

```
x1<-AR1(N,1)
x2<-AR1(N,1)
summary(lm(x2~x1))
```

- Along with there clearly being no mechanical relationship between the series (since they are random), the standard errors are incorrect for classic OLS

Spurious Regressions (cont.)

- Why are standard errors incorrect?
- Suppose we wish to regress y_t on x_t using

$$y_t = \beta_0 + \beta_1 x_t + u_t$$

- Rearranging for u_t, we have:

$$u_t = y_t - \beta_0 - \beta_1 x_t$$

- Back-substituting for $y_t = y_{t-1} + u_{yt-1}$ and $x_t = x_{t-1} + u_{xt-1}$, we have:

$$u_t = \left(y_{t-1} + u_{yt-1}\right) - \beta_0 - \beta_1 \left(x_{t-1} + u_{xt-1}\right)$$

- Doing so repeatedly for back to period 1, we get:

$$u_t = y_1 + \sum_{i=1}^{t-1} u_{yi} + \beta_0 - \beta_1 x_1 - \beta_1 \sum_{i=1}^{t-1} u_{xi}$$

- Note that because of the $\sum_{i=1}^{t-1} u_{yi}$ and $\beta_1 \sum_{i=1}^{t-1} u_{xi}$ the variance explodes as t gets large relative to the initial state.

Cointegration and Error-Correction

- Trending time series cause problems due to the spurious regression
  - This tends to be a problem with any macro data
- Differencing helps, but there are drawbacks
  - Cannot speak to long-run changes, only short-run (since identifying variation is based on first differences or higher order differences)
- **Cointegration** provides a framework for identifying and estimating time series regressions

Definition of Cointegration

- Suppose we have the following time-series model

$$Y_t = \beta_0 + \beta_1 X_t + u_t$$

- Estimate the model to get $\widehat{\beta}_0$ and $\widehat{\beta}_1$. Constructing the residuals:

$$\widehat{u}_t = Y_t - \widehat{\beta}_0 - \widehat{\beta}_1 X_t$$

- If $\widehat{u}_t \sim I(0)$, then Y_t and X_t are **cointegrated**
- This is trivial if Y_t and X_t are both $I(0)$
- This more interesting when Y_t and X_t are both $I(1)$
- How often does this occur given random time series generated by a process with a unit-root?

Cointegrated series - Monte Carlo examples

- Insert examples, theory

```
for(i in 1:1000){
      x<-AR1(N,1)
      y<-AR1(N,1)
      errors<-resid(lm(y~x))
      adftest<-adf.test(errors)
      p<-adftest$p.value
      if(i==1){res<-data.frame(p)}
      if(i>1){res<-rbind(res,data.frame(p))}
}
```

- Calculate how many reject a unit root in favor of stationarity

```
mean(res$p<0.1,na.rm=TRUE)
```

Error Correction Model

- If Y_t and X_t are $I(1)$, but $\widehat{u}_t \sim I(0)$, then we can estimate using OLS the following "Error correction model"

$$\Delta Y_t \quad = \quad a_0 + b_1 \Delta X_t + \pi \widehat{u}_{t-1} + e_t$$

- This regression is **not** spurious because $\widehat{u}_{t-1}$, ΔY_t and ΔX_t are all $I(0)$
- Since, Y_t and X_t are also $I(0)$, we can obtain consistent estimates for b_1 using standard regression
- We can also obtain long-run equilibrium dynamics by focusing on π.
  - $\widehat{u}_{t-1} \neq 0$ indicates *disequilibrium* between Y and X in $Y_{t-1} = \beta_0 + \beta_1 X_{t-1} + u_{t-1}$
  - $\pi = -1$ equilibrium is reached immediately
  - $\pi \in (-1, 0]$ equilibrium is reached gradually
  - $\pi < -1$ suggests an over-correction

Error Correction Model to ARDL model

- The ECM model is equivalent to a ARDL model (Autoregressive Distributed Lag), which we presented a few lectures ago when talking about VARs. To see this, note that:

$$\Delta Y_t = a_0 + b_1 \Delta X_t + \pi \widehat{u}_{t-1} + e_t$$

- Expanding ΔY_t, ΔX_t, and $\widehat{u}_{t-1}$, we have:

$$Y_t - Y_{t-1} = a_0 + b_1 (X_t - X_{t-1}) + \pi (Y_{t-1} - \beta_0 - \beta_1 X_{t-1}) + e_t$$

- Bringing all lags to the RHS:

$$Y_t = a_0 + Y_{t-1} + \pi Y_{t-1} + b_1 X_t - b_1 X_{t-1} - \pi \beta_0 - \pi \beta_1 X_{t-1} + e_t$$

- Collecting terms

$$Y_t = a_0 - \pi \beta_0 + (1 + \pi) Y_{t-1} + b_1 X_t + (-b_1 - \pi \beta_1) X_{t-1} + e_t$$

- Thus, we have an ARDL model.

Engel-Granger Technique

- Engel and Granger have proposed a technique for evaluating data that may be spurious.
- **Step 1:** Determine whether X_t and Y_t are cointegrated.
  - If X_t and Y_t are $I(0)$, then use classic regression
  - If only one of X_t and Y_t are $I(1)$, and the other $I(0)$, then need a new techinque
  - If X_t and Y_t are $I(1)$, then run

    $$Y_t = \beta_0 + \beta_1 X_t + u_t$$

    and collect residuals. Go to step 2.
- **Step 2:** Check whether u_t is $I(0)$
  - If u_t is $I(0)$, move to step 3.
  - If u_t is $I(1)$, find a new model
- **Step 3:** Estimate and interpret:

$$\Delta Y_t = a_0 + b_1 \Delta X_t + \pi \widehat{u}_{t-1} + e_t$$

Example - Co-integration of investment funds

- Download a year of daily opening prices for SPY and VOO

```
getSymbols('SPY',from='2014-11-12',to='2015-11-12')
getSymbols('VOO',from='2014-11-12',to='2015-11-12')
prices.spy <- SPY$SPY.Open
prices.voo <- VOO$VOO.Open
```

- **Step 1:** Determine whether *SPY* and *VOO* have unit root.

```
adf.test(prices.spy)
adf.test(prices.voo)
```

Example - Co-integration of investment funds

- **Step 2:** Regress

$$SPY_t = a_0 + b_1 VOO_t + u_t$$

and test whether u_t is $I(0)$

```
coint <- lm(prices.spy~prices.voo)
summary(coint)
beta<-coint$coef
resid <- prices.spy - (beta[1] + beta[2]*prices.voo)
adf.test(resid)
```

- **Step 3:** Estimate and interpret:

$$\Delta SPY_t = a_0 + b_1 \Delta VOO_t + \pi \widehat{u}_{t-1} + e_t$$

```
dSPY<-prices.spy-lag(prices.spy,1)
dVOO<-prices.voo-lag(prices.voo,1)
ecm <-lm(dSPY~dVOO+lag.resid)
summary(ecm)
```

The LASSO

- Time series focuses on dynamic relationships within and across variables
  - We have a choice of how many lags to include, etc...
- The LASSO:
  - "**L**east **A**bsolute **S**hrinkage and **S**election **O**perator"
  - Regression-based model selection procedure popular in *data science*
- Suppose that we have N observations, P potential explanatory variables
- The Lasso Problem:

$$\max_{\beta_p} \sum_{i=1}^{N} \left(y_i - \sum_{p=1}^{P} \beta_p x_{ip} \right)^2 \tag{1}$$

$$s.t. \qquad \sum_{p=1}^{P} \left| \beta_p \right| < \lambda \tag{2}$$

- (1) is the OLS problem.
- (2) constrains the total absolute size of all coefficients

The LASSO (cont.)

- Within the context of an ARp model, the problem can be written as:

$$\max_{\beta_p} \sum_{i=1}^{N} \left(y_t - \sum_{p=1}^{P} \beta_p y_{t-p} \right)^2$$

$$s.t. \qquad \sum_{p=1}^{P} \left| \beta_p \right| < \lambda$$

- λ can be chosen by cross-validation. Let's first look at the procedure
- Load "lars" library and some SPY data.

```
library(lars)
getSymbols('SPY',from='2001-05-12',to='2015-11-12')
prices.spy <- (as.numeric(SPY$SPY.Open))
```

- Define a first difference and the length of the vector

```
dSPY0<-diff(prices.spy,lag=1)
len<-length(dSPY)
```

The LASSO (cont.)

- Define the current difference (dSPY) and five lags

```
dSPY<-dSPY0[6:(len)]
dSPY1<-dSPY0[5:(len-1)]
dSPY2<-dSPY0[4:(len-2)]
dSPY3<-dSPY0[3:(len-3)]
dSPY4<-dSPY0[2:(len-4)]
dSPY5<-dSPY0[1:(len-5)]
```

- Run a regression, a LASSO, and compare coefficients

```
lm.reg<-lm(dSPY~dSPY1+dSPY2+dSPY3+dSPY4+dSPY5-1)
d<-cbind(dSPY1,dSPY2,dSPY3,dSPY4,dSPY5)
lasso.reg<-lars(d,dSPY,type="lasso",normalize=FALSE)
coef(lm.reg)
coef(lasso.reg)
plot(lasso.reg)
```

The LASSO (cont.)

- The x-axis on the plot represents:

$$s = \frac{\sum_p |\beta_p|}{\sum_p |\beta_p^{ols}|}$$

- Choose the optimal λ via cross validation

```
CVlasso<-cv.lars(d,dSPY,type="lasso",normalize=FALSE)
str(CVlasso)
```

- Extract the optimal s using "which.min" and "index"

```
opt<-CVlasso$index[which.min(CVlasso$cv)]
predict(lasso.reg,s=opt,type="coef",mode="fraction")
```

- This particular prediction code gives the coefficient estimates under the CV-optimal solution.