MLE Lab 3 – 1/27/17

Andy Ballard

January 27, 2017

To do

- Omitted variable bias
- F test
- Likelihood ratio test
- Maximizing likelihood functions
- Basic regression with MLE

Setup

```
# Start with a clean workspace
rm(list=ls())

# Function to load packages
loadPkg <- function(toLoad){
  for(lib in toLoad){
    if(! lib %in% installed.packages()[,1])
      { install.packages(lib, repos='http://cran.rstudio.com/') }
    suppressMessages( library(lib, character.only=TRUE) ) }
}

# Load libraries
pkgs <- c("ggplot2", 'lmtest', 'car', 'gap', 'sandwich', 'reshape2', 'foreign', 'MASS')
loadPkg(pkgs)

# Set a theme for gg
theme_set(theme_bw())

# Functions that I use frequently
char <- function(x){ as.character(x) }
num <- function(x){ as.numeric(char(x)) }

## Relevant paths
if((Sys.info()['user']=='aob5' | Sys.info()['user']=='Andy')){
  lab3Path <- paste0(path.expand("~"), "MLE_LAB/Lab 3 - Introduction to Maximum Likelihood")
}
```

Muller & Seligson Regression Analytics

Omitted Variable Bias

Consequences of omitted variable bias?

- Biased parameter estimates and standard errors

In cases of more than two predictors direction of bias becomes hard to assess.

```
msrepBiRegData <- na.omit(msrep[,c('deaths75ln', 'upper20', 'sanctions75ln')])
cor(msrepBiRegData)

# Estimate coefficients
coeftest( lm(deaths75ln ~ upper20 + sanctions75ln, data=msrepBiRegData) )

# If we excluded sanctions75ln from the model what would you expect of the
# coefficient on upper20
coeftest( lm(deaths75ln ~ upper20, data=msrepBiRegData) )

# If we excluded upper20 from the model what would you expect of the
# coefficient on sanctions75ln
coeftest( lm(deaths75ln ~ sanctions75ln, data=msrepBiRegData) )

# More complicated model
ivs <- c('upper20', 'energypcln', 'intensep',
      'sanctions70ln', 'sanctions75ln', 'deaths70ln')
msrepRegData <- na.omit(msrep[,c('deaths75ln', ivs)])
olsForm1 <- formula(paste0('deaths75ln ~ ', paste(ivs, collapse=' + ')))
olsForm2 <- formula(paste0('deaths75ln ~ ', paste(ivs[c(2:length(ivs))], collapse=' + ')))
mod1 <- lm(olsForm1, data=msrepRegData)
mod2 <- lm(olsForm2, data=msrepRegData)
```

```
# View model results
summary(mod1)
```

```
##
## Call:
## lm(formula = olsForm1, data = msrepRegData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0584 -0.6402 -0.0818  0.8242  4.4539
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -7.51197    2.06031  -3.646 0.000592 ***
## upper20        0.10878    0.02385   4.561 2.89e-05 ***
## energypcln     0.27170    0.16737   1.623 0.110231
## intensep       1.17761    0.57111   2.062 0.043948 *
## sanctions70ln -0.32637    0.27122  -1.203 0.234005
## sanctions75ln  0.89604    0.23304   3.845 0.000315 ***
## deaths70ln     0.47747    0.11469   4.163 0.000111 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.429 on 55 degrees of freedom
## Multiple R-squared:  0.7063, Adjusted R-squared:  0.6743
## F-statistic: 22.04 on 6 and 55 DF,  p-value: 5.034e-13
```

```r
summary(mod2)
```

```
##
## Call:
## lm(formula = olsForm2, data = msrepRegData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3515 -0.8606 -0.1965  0.7870  4.8276
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.6866     1.1710   0.586  0.56000
## energypcln     -0.1151     0.1679  -0.686  0.49569
## intensep        1.1556     0.6644   1.739  0.08748 .
## sanctions70ln  -0.3487     0.3155  -1.105  0.27376
## sanctions75ln   0.9073     0.2711   3.347  0.00147 **
## deaths70ln      0.4328     0.1330   3.256  0.00192 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.663 on 56 degrees of freedom
## Multiple R-squared:  0.5952, Adjusted R-squared:  0.5591
## F-statistic: 16.47 on 5 and 56 DF,  p-value: 5.684e-10
```

F-test

A partial F-test (also called an incremental F-test or an extra sum of squares F-test) is the appropriate test to use when the simultaneous test of the statistical significance of a group of variables is needed.

A partial F-test requires fitting two regression models.

- A full model (F) that includes all the variables currently of interest
- A reduced model (R) that includes all the variables currently of interest except those whose statistical significance we wish to test.

The partial F-test assesses whether the improvement in model fit (as assessed by a reduction in prediction error) using the full model is too large to be ascribed to chance alone.

- To determine this we look at the average difference in SSE between the two models and compare this to the average prediction error in the best model we have (the full model).
- If these two measures of error are roughly the same, the added regressors in the full model are not contributing very much.
- But if the the average decrease in prediction error due to adding the regressors is quite large, then these regressors are contributing a good deal of explanatory power and thus should be retained.

The formula for the of the partial F-test is:

```r
# How would we do a partial F-test
ssr1 <- sum(resid(mod1)^2)
ssr2 <- sum(resid(mod2)^2)
chgReg <- ncol(model.matrix(mod1)) - ncol(model.matrix(mod2))
# F statistic
Fstat <- ((ssr1-ssr2)/chgReg)/(ssr1/df.residual(mod1))
1-pf(abs(Fstat), chgReg, df.residual(mod1))

# Using anova function from base R
anova(mod1, mod2)
```

Likelihood Ratio Test

In statistics, a likelihood ratio test is a statistical test used to compare the fit of two models, one of which (the null model) is a special case of the other (the alternative model). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one model than the other.

```r
# How would we run a likelihood ratio test
ltest <- nrow(model.matrix(mod2))*log(ssr1/ssr2)
pchisq(abs(ltest), df=chgReg, lower.tail=FALSE)

# Using lrtest from lmtest library
lrtest(mod1, mod2)
```

Maximum Likelihood Estimation

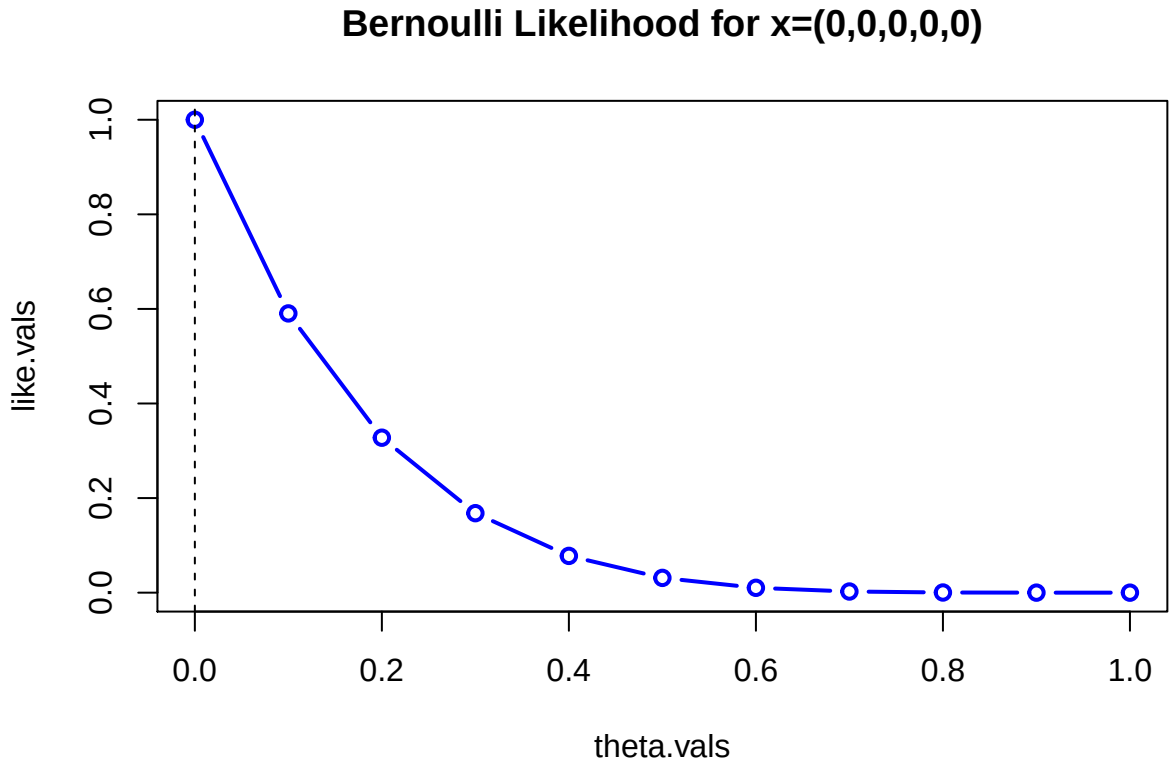
Bernoulli likelihood

Let's look at the Bernoulli likelihood function:

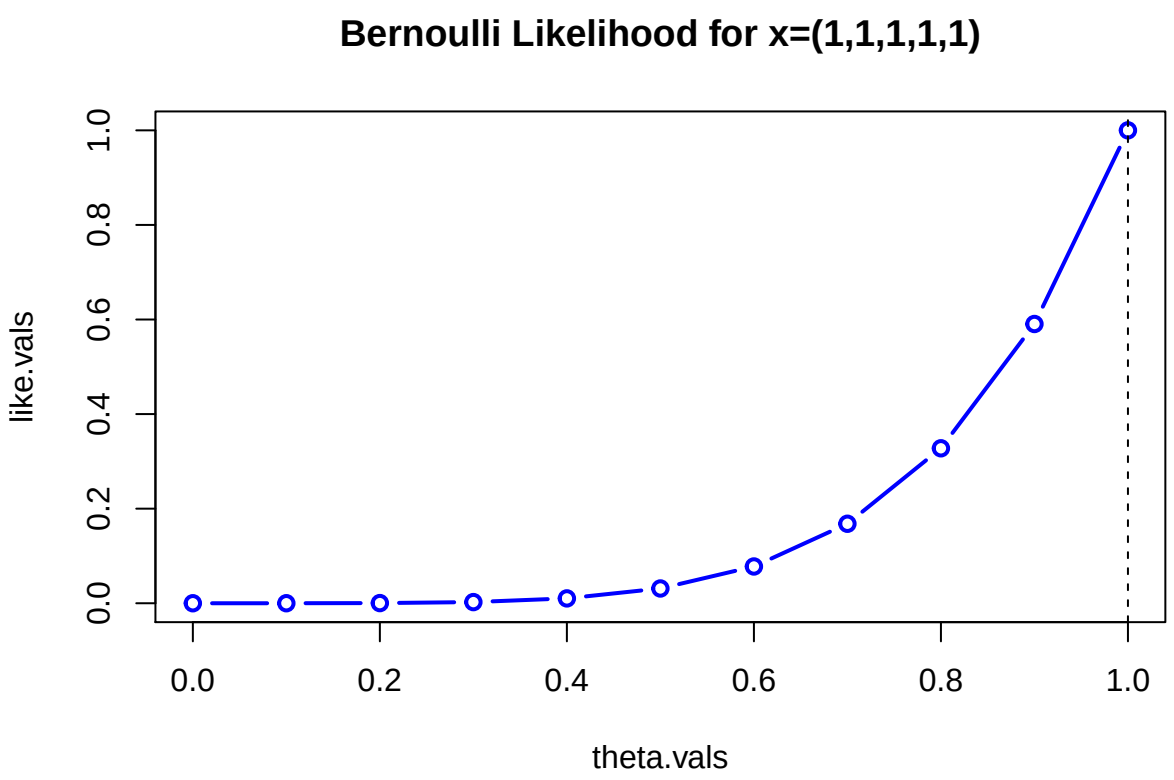
$$f(X, \theta) = (\theta)^x * (1-\theta)^{1-x}$$

```r
#
# Bernoulli example
#

# Bernoulli likelihood
# x = 0, 1
# f(x,theta) = (theta^x)*(1-theta)^(1-x)

likelihood.Bernoulli <- function(theta, x) {
  # theta   success probability parameter
  # x       vector of data
  n <- length(x)
  ans <- theta^sum(x) * (1-theta)^(n-sum(x))
  return(ans)
}

# plot Bernoulli likelihood (x = c(0,0,0,0,0))
x <- rep(0,5)
theta.vals <- seq(0,1, length.out=11)
```

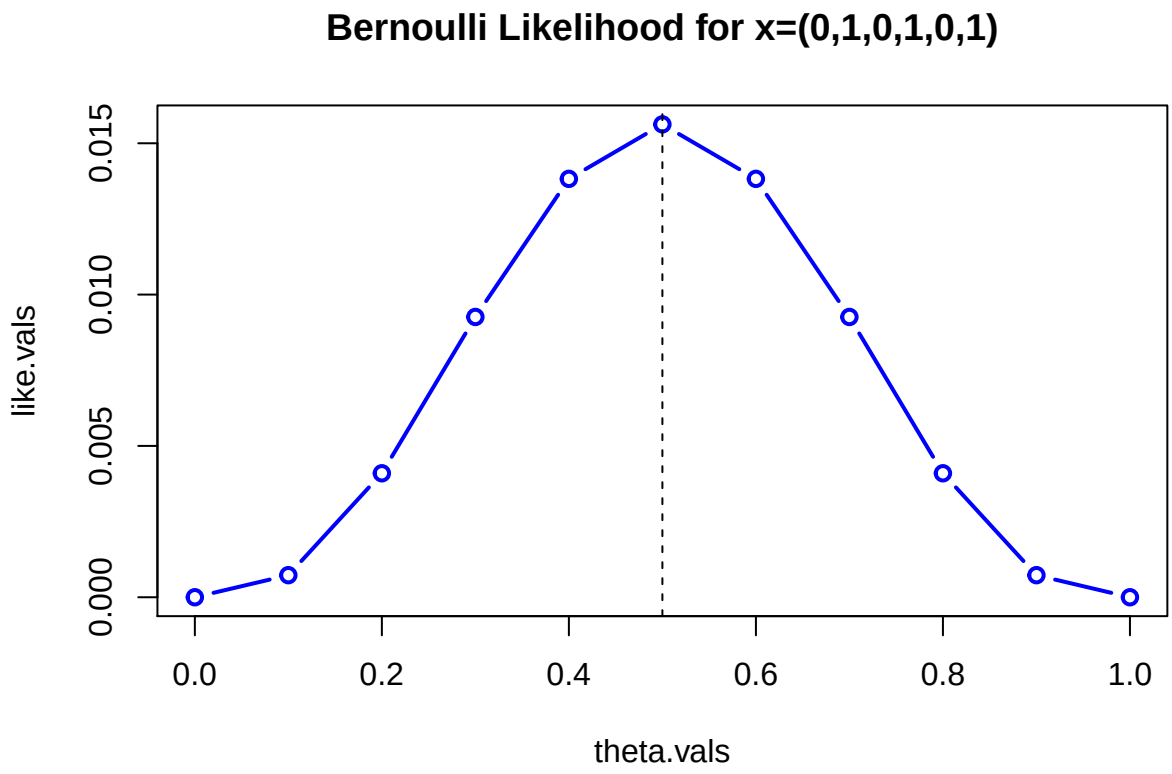
```r
like.vals <- likelihood.Bernoulli(theta.vals, x)
plot(theta.vals, like.vals, type="b", col="blue", lwd=2,
     main="Bernoulli Likelihood for x=(0,0,0,0,0)")
abline(v=0, lty=2)
```




```r
# plot Bernoulli likelihood (x = c(1,1,1,1,1))
x <- rep(1,5)
like.vals <- likelihood.Bernoulli(theta.vals, x)
plot(theta.vals, like.vals, type="b", col="blue", lwd=2,
     main="Bernoulli Likelihood for x=(1,1,1,1,1)")
abline(v=1, lty=2)
```




```
# plot Bernoulli likelihood (x = c(0, 1, 0, 1, 0, 1))
x <- c(0,1,0,1,0,1)
like.vals <- likelihood.Bernoulli(theta.vals, x)
plot(theta.vals, like.vals, type="b", col="blue", lwd=2,
     main="Bernoulli Likelihood for x=(0,1,0,1,0,1)")
abline(v=0.5, lty=2)
```



Normal likelihood

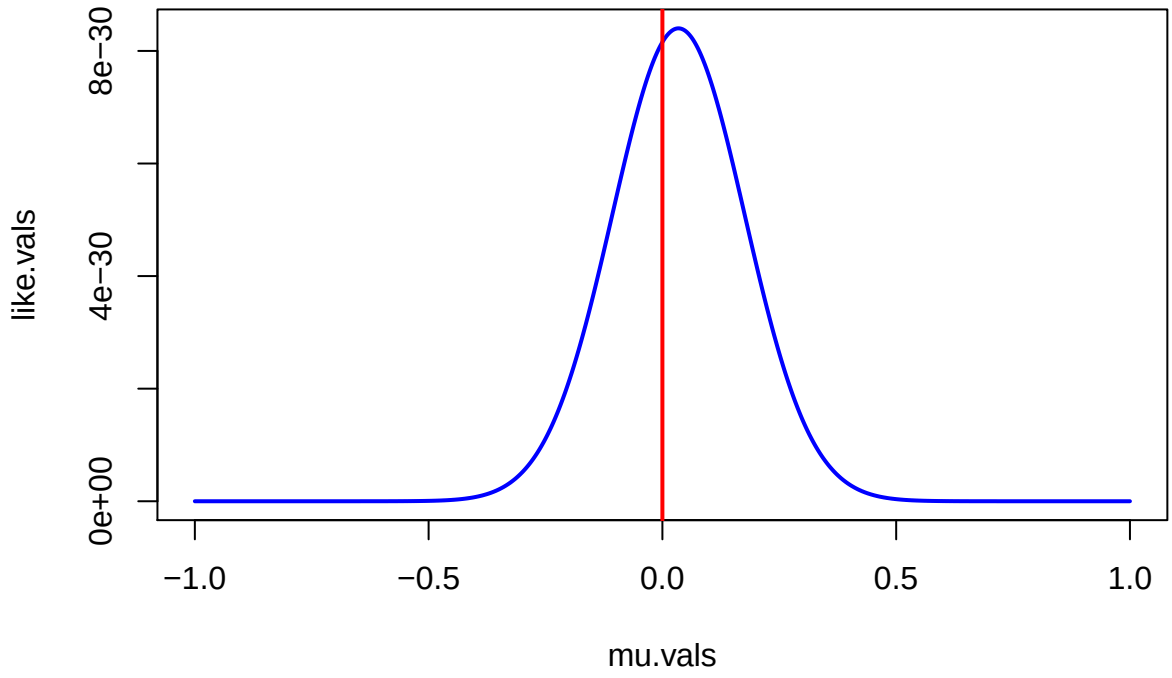
Now let's look at the Normal distribution likelihood function, which is given by:

$$f(X, \mu, \theta^2) = (2\pi\sigma^2)^{-n/2} exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j - \mu)^2\right)$$

It may look scary, but it's really your best friend.

```
# normal likelihood
#

likelihood.normal <- function(theta, x) {
  # theta   vector of normal distribution parameters
  #         theta = (mu, sig2)'
  # x       vector of data
  mu <- theta[1]
  sig2 <- theta[2]
  n <- length(x)
  a1 <- (2*pi*sig2)^-(n/2)
  a2 <- -1/(2*sig2)
  y <- (x-mu)^2
  ans <- a1*exp(a2*sum(y))
  return(ans)
```

```
}

likelihood.normal.mu <- function(mu, sig2=1, x) {
  # mu      mean of normal distribution for given sig
  # x       vector of data
  n <- length(x)
  a1 <- (2*pi*sig2)^-(n/2)
  a2 <- -1/(2*sig2)
  y <- (x-mu)^2
  ans <- a1*exp(a2*sum(y))
  return(ans)
}


# generate N(0,1) data
n <- 50
set.seed(123)
x <- rnorm(n, mean=0, sd=1)

# compute normal likelihood as function of mu
mu.vals <- seq(-1,1, length.out=10001)
like.vals <- rep(0,length(mu.vals))
for (i in 1:length(like.vals)) {
  like.vals[i] <- likelihood.normal.mu(mu.vals[i], sig2=1, x=x)
}

plot(mu.vals, like.vals, type="l", col="blue", lwd=2)
abline(v=0, col="red", lwd=2)
```




```
mean(x)
```

```
## [1] 0.03440355
```

Regression with MLE

```
######################################################
# Load Data
data <- read.dta("http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp2.dta")
    # DV: female labor force participation (lfp)
    # IVs: age (age), log wage (lwg), household income (inc)
table(data$lfp)
```

```
##
## NotInLF    inLF
##     325     428
```

```
data$lfp <- as.numeric(data$lfp)-1
######################################################


######################################################
```

```
### BUILDING YOUR OWN MLE ESTIMATOR, STEP BY STEP

# 1. what do we need?
# input - data: y, X
# input - model components: stochastic/family, systematic/link
# output - vector of betas

binreg <- function(y, X){
    X <- cbind(1,X)
    neg.loglik <- function(b, X, y){
        pi <- as.vector(1/(1+exp(-X %*% b))) # the link function
        - sum(y*log(pi) + (1-y)*log(1-pi)) # the neg log likelihood
    }
    results <- optim(rep(0,ncol(X)), neg.loglik, hessian=T, method="BFGS", X=X, y=y)
    list(coefficients=results$par,
        varcovariance=solve(results$hessian),
        converged=results$convergence==0)
}

# Run function on data
dv <- data$lfp
iv <- cbind(data$age, data$lwg, data$inc)
m1 <- binreg(dv, iv)

m1$coefficients
```

```
## [1]  0.81255880 -0.01976440  0.75369499 -0.02534331
```

```
sqrt(diag(m1$varcovariance))
```

```
## [1] 0.442990257 0.009444098 0.143352147 0.006827226
```

```
m1$converged
```

```
## [1] TRUE
```

```
#######################################################

#######################################################
### GLM function
m2 <- glm(lfp ~ age + lwg + inc, data=data, family=binomial(link="logit"))
summary(m2)
```

```
##
## Call:
## glm(formula = lfp ~ age + lwg + inc, family = binomial(link = "logit"),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6622  -1.2106   0.7918   1.0173   1.9462
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.813310   0.442895   1.836 0.066306 .
## age         -0.019783   0.009440  -2.096 0.036111 *
## lwg          0.753688   0.143352   5.258 1.46e-07 ***
## inc         -0.025338   0.006826  -3.712 0.000206 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance:  984.0  on 749  degrees of freedom
## AIC: 992
##
## Number of Fisher Scoring iterations: 4

#####################################################
```