Statistics is the inverse problem of probability.

Probability: we know the parameters, we figure out what the generated data are likely to look like.

Statistics: we know the data, we figure out what the parameters are likely to be.

One way of doing statistics, i.e. figuring out the unknown parameters, is to use Maximum Likelihood Estimation (MLE). (There are many other ways, e.g. OLS, Bayesian, non-parametric methods).

The idea of MLE is intuitive. Given observed data X, we find an estimate $\hat{\theta}$ that makes X most likely to be observed, and we say that $\hat{\theta}$ is our best guess for θ.

For example, given 5 coin flips, $X = \{H, T, T, T, T\}$, with θ is the chance of flipping head. We find the $\hat{\theta}$ that maximizes the likelihood that we see this sequence of X. Doing the math, $\hat{\theta} = \frac{1}{n}\sum X = 0.2$, and that is our best guess for θ.

While this idea makes intuitive sense, we also want to have formal proof that it's a good idea. So far we have not managed to prove anything about how good of a guess is $\hat{\theta}$, i.e. how close it is to θ. So we need to learn more about the properties of MLE, by the way of learning about the expectation and variance of this estimator.

What exactly do we mean by expectation and variance of an estimator? Recall that only a random variable has an expectation and a variance. (Indeed, a number will just be ... a number). So what is random about the estimator?

Recall the coin flipping example. $\hat{\theta} = \frac{1}{n}\sum X$. What's random in here? X is random. Once X is observed, we got an estimate, a number. An estimator is a rule to be applied to any set of data that we will observe.

Statisticians are interested in estimators that have good properties, i.e. estimators that will give estimates that are close to the true value most of the time. Notice that I say "most of the times," because for a given dataset we have no way to know for sure how close it is to the true value.

For example, let's say the true value of $\theta = 0.4$.

```
theta <- 0.4
(X <- rbinom(5, size = 1, prob = theta))

## [1] 1 1 1 1 0

(MLE_estimate <- mean(X))

## [1] 0.8
```

There is no way to know for a particular dataset whether the MLE estimate is close to θ or not. We can only know about the properties of the MLE estimate across many datasets. Some "good" properties of an estimator is unbiasedness (mean of estimator equals the true value), consistent (explain here: A sequence of $\hat{\theta}_1, \ldots, \hat{\theta}_n$ converges in probability to true value $limPr|\hat{\theta} - \theta| > \epsilon \leftarrow 0$), efficient (an estimator that has small variance, i.e. it always stays rather close to the true value, no matter what particular dataset we get). When you go on and read methods paper on your own, when people propose new estimators, they also come up with an intuitive idea, and then have to prove that their estimator has these nice properties. So if your models are not showing significant results, you may want to search for a more efficient estimator, for example.

Side note: You've heard that MLE is not always unbiased. You wonder why we use it. It's because MLE is more efficient that OLS. (Note that OLS is BLUE, best linear UNBIASED estimator. So OLS is best among unbiased estimators, but MLE doesn't belong to that class).

Finally, talk about Fisher's information. Fisher's information quantifies how much information we have about the true value of the parameter There are three definitions that can be shown to be algeabrically equivalent:

$$I(\theta) = E\left(\left[\frac{\partial}{\partial\theta}LL\right]^2\right) \tag{1}$$

$$I(\theta) = -E\left(\frac{\partial^2}{\partial\theta\partial\theta}LL\right) \tag{2}$$

$$I(\theta) = Var\left(\frac{\partial}{\partial\theta}LL\right) \tag{3}$$

Most of the times, you use (2) to calculate $I(\theta)$, and (2) and (3) are the most intuitive.

(2) is the definition you see in class. The second derivative of the log likelihood quantifies the curvature of the log likelihood. When we zoom into the area around the estimator (which is also pretty close to the true value) we can say that it's "locally quadratic". A "blunt" support curve (one with a shallow maximum) would have a low negative expected second derivative, and thus low information; while a sharp one would have a high negative expected second derivative and thus high information.

1 Cobb Doublas example

$$Y = AL^{\alpha}K^{\beta} \tag{4}$$

$$logY = logA + \alpha logL + \beta logK \tag{5}$$

We can estimate α, β, but how would we test the hypothesis that $H_0 : \alpha+\beta > 1$. We can use simulation.

2 R stuff

```
########## Normally Distributed DVs 1 ############

library(bbmle)

## Loading required package:  stats4

library(arm)

## Loading required package:  MASS
## Loading required package:  Matrix
## Loading required package:  lme4
##
## arm (Version 1.8-6, built:  2015-7-7)
## Working directory is /home/anh/projects/ps630_lab/ps733_s16/W2

# Data from 2012 American National Election Study (~3100 non-Latino-white-identifying respo

anesdata <- na.omit(read.delim("2012 ANES_Economic Prefs.txt", header=TRUE))
summary(anesdata)

##      south             male             age01             unemp
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.3333   1st Qu.:0.00000
##  Median :0.0000   Median :1.0000   Median :0.5833   Median :0.00000
##  Mean   :0.3232   Mean   :0.5008   Mean   :0.5768   Mean   :0.05109
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.8333   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##      union           income01          auth01          extrav01
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.2963   1st Qu.:0.2500   1st Qu.:0.3333
##  Median :0.0000   Median :0.5185   Median :0.5000   Median :0.5000
##  Mean   :0.1665   Mean   :0.5232   Mean   :0.5643   Mean   :0.5221
##  3rd Qu.:0.0000   3rd Qu.:0.7778   3rd Qu.:0.7500   3rd Qu.:0.6667
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##     agree01          consc01          stable01        openness01
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.5833   1st Qu.:0.6667   1st Qu.:0.5000   1st Qu.:0.5000
##  Median :0.6667   Median :0.8333   Median :0.6667   Median :0.6667
##  Mean   :0.6971   Mean   :0.7818   Mean   :0.6557   Mean   :0.6320
##  3rd Qu.:0.8333   3rd Qu.:0.9167   3rd Qu.:0.8333   3rd Qu.:0.7500
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      econ01           educ1             educ2            educ3
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.2333   1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.4333   Median :0.00000   Median :0.000   Median :0.0000
##  Mean   :0.4306   Mean   :0.06465   Mean   :0.245   Mean   :0.3198
##  3rd Qu.:0.6000   3rd Qu.:0.00000   3rd Qu.:0.000   3rd Qu.:1.0000
```

```r
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.000   Max.   :1.0000
##      educ4            educ5
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.2208   Mean   :0.1498
##  3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000

### Simple linear regression of social welfare support on income (coded 0-1) and union membe

X <- array(NA, c(length(anesdata$econ01),3)) #Initialize the design matrix
X[ ,1] <- 1 #column of 1s for constant
X[ ,2] <- anesdata$income01 #household income, coded from 0-1
X[ ,3] <- anesdata$union #dichotomous indicator for family member union membership
y <- anesdata$econ01 #additive scale of several social welfare policy items, coded from 0-1

# Define the LL function in general terms (this code can be used for any # of predictors):

LL_normreg = function(params, y, X){
  B = matrix(NA, nrow = length(params) - 1, ncol = 1)
  B[,1] = params[-length(params)]
  sigma    = params[[length(params)]]
  minusll  = -sum(dnorm(y, X %*% B, sigma, log=T))
  return(minusll)
}

# Declare the names of the parameters (from B0 to B[# of predictors], and sigma):

parnames(LL_normreg) <- c("B0", "B1", "B2", "sigma")

# Fit the model using mle2 ('vecpar=TRUE' tells mle2 that the first argument passed to the
  # LL function is a vector of all parameters with names declared in 'parnames' above and i

fit <- mle2(LL_normreg, start = c(B0 = mean(y), B1 = 0, B2 = 0, sigma = sd(y)),
            data=list(y=y,X=X), vecpar = TRUE, control=list(maxit=5000))

## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced
```

```
## Warning in dnorm(y, X %*% B, sigma, log = T): NaNs produced

summary(fit)

## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = LL_normreg, start = c(B0 = mean(y), B1 = 0,
##     B2 = 0, sigma = sd(y)), data = list(y = y, X = X), vecpar = TRUE,
##     control = list(maxit = 5000))
##
## Coefficients:
##         Estimate Std. Error z value     Pr(z)
## B0     0.4686993  0.0085633 54.7333 < 2.2e-16 ***
## B1    -0.0931828  0.0143522 -6.4926 8.439e-11 ***
## B2     0.0639140  0.0111607  5.7267 1.024e-08 ***
## sigma  0.2322673  0.0029167 79.6349 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: -259.6571

str(summary(fit))

## Formal class 'summary.mle2' [package "bbmle"] with 3 slots
##   ..@ call  : language mle2(minuslogl = LL_normreg, start = c(B0 = mean(y), B1 = 0, B2 =
##   ..@ coef  : num [1:4, 1:4] 0.4687 -0.09318 0.06391 0.23227 0.00856 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:4] "B0" "B1" "B2" "sigma"
##   .. .. ..$ : chr [1:4] "Estimate" "Std. Error" "z value" "Pr(z)"
##   ..@ m2logL: num -260

summary(fit)@m2logL

## [1] -259.6571
```