Maximum Likelihood and Hypothesis Testing with R

```
>
> # Multnomial Example
> ThetaHat <- c(0.2615,0.6014,0.8333) ; ThetaHat0 <- 0.5755
> n <- c(65,429,36) ; nn <- sum(n)
> # Break the calculation up into parts
> kore <- n * (ThetaHat*log(ThetaHat) + (1-ThetaHat)*log(1-ThetaHat) )
> kore0 <- nn * (ThetaHat0*log(ThetaHat0) + (1-ThetaHat0)*log(1-ThetaHat0) )
> G <- 2 * ( sum(kore) - kore0 )
> pval <- 1 - pchisq(G,2)
> cat("Chi-square = ",G,"   df = 2    p = ", pval, "\n")
Chi-square =  38.5068    df = 2    p =  4.348651e-09
>

> # Beta Example: Likelihood Ratio Test
>
> bloglike <- function(theta,xx)    # - log likelihood for beta distribution
+                                   # with parameter theta = (alpha,beta). Data in
xx.
+     {
+     alpha <- theta[1] ; beta <- theta[2]  ; nn <- length(xx)
+     if(alpha<0) {cat("alpha=",alpha,"\n") ; stop("Alpha must be positive!") }
+     if(beta<0) {cat("beta=",beta,"\n") ; stop("Beta must be positive!") }
+     bloglike <- -1*nn*(
+                 lgamma(alpha+beta) - lgamma(alpha) - lgamma(beta) +
+                 (alpha-1)*mean(log(xx)) + (beta-1)*mean(log(1-xx))
+                      )
+     bloglike
+     }# End Function bloglike
>
> n <- 1000 ; a <- 2 ; b <- 2.1 # True alpha & beta
> set.seed(9999) # Set seed for the random number generator (optional)
> datta <- rbeta(n,a,b) # Simulate the data
> # Students will usually read data from a file. First write it out.
> write(datta,file="beta.dat.txt")
> # Now read it
> bdat <- scan("beta.dat.txt")
Read 1000 items
> print(bdat) # Always look at your data
   [1] 0.772887100 0.715413200 0.624496500 0.282058000 0.731617000 0.237205100
   [7] 0.435583700 0.842859500 0.750242000 0.364663400 0.770057000 0.373551000
  [13] 0.370197200 0.455377400 0.405387600 0.438078100 0.959874500 0.405276800
  [19] 0.623807200 0.731510900 0.531346000 0.455420000 0.478678500 0.356962700
  [25] 0.141350700 0.778775900 0.424710600 0.542864600 0.297572100 0.375114200
  [31] 0.794531600 0.734867600 0.584070800 0.446863600 0.321399000 0.148559800
  [37] 0.272621900 0.629465500 0.526445000 0.528217100 0.231862100 0.556230900
  [43] 0.348106800 0.439590100 0.207724700 0.264911900 0.731896600 0.390316600
  [49] 0.731221700 0.482853300 0.039466730 0.583882100 0.273375000 0.676954900
  [55] 0.736502600 0.819758300 0.061926400 0.632920200 0.451868500 0.469018800
  [61] 0.037041060 0.197020900 0.243244000 0.569664700 0.404670900 0.247045600
  [67] 0.489595800 0.670667000 0.414184000 0.265394900 0.944094400 0.431725900
  [73] 0.401168900 0.738832900 0.614826600 0.470829100 0.140540900 0.366207300
```

. . . Skipping . . .

```
 [991] 0.310637100 0.561382600 0.257007400 0.467186300 0.534698700 0.097042120
 [997] 0.382139400 0.822015500 0.104756600 0.344154000
> m <- mean(bdat) ; v <- var(bdat)
> cat("Mean = ",m,"   Variance = ", v, "\n")
Mean =  0.4869979    Variance =  0.05004784
>
> # We will maximize the log likelihood numerically, but where should
> # we start looking?
>
> ############################################################
> # Mathematica                                              #
> # eqn = {m == a/(a+b),v == a b / ((a+b)^2 (a+b+1))}        #
> # Solve[eqn,{a,b}]                                         #
> ############################################################
> # Yielding Method of Moments estimators as starting values for numerical MLE
> astart <- m*(m-m^2-v)/v ; bstart <- (m-2*m^2+m^3-v+m*v)/v
> cat("Astart = ",astart,"   Bstart = ", bstart, "\n")
Astart =  1.944019    Bstart =  2.047824
> start1 <- c(astart,bstart) # The nlm function wants a vector of starting values
>                            # for a vector parameter.
> Full <- nlm(bloglike,start1,xx=bdat) # Full (unrestricted) model
> # Look at the "model object" Full
> print(Full)
$minimum
[1] -123.9845

$estimate
[1] 1.934915 2.041719

$gradient
[1] 8.078875e-07 7.830272e-06

$code
[1] 1

$iterations
[1] 4

>
> # Now fit a REDUCED model in which alpha=beta
> # bwrapper is a function for reduced model alpha = beta
> bwrapper <- function(ab,xx)
+     {
+     bwrapper <- bloglike(c(ab,ab),xx)
+     bwrapper
+     }
> starteq <- (astart+bstart)/2
> Red <- nlm(bwrapper,starteq,xx=bdat) ; print(Red)
alpha= -0.3588073
Error in bloglike(c(ab, ab), xx) : Alpha must be positive!
>
> # nlm does unconstrained optimization, and it is looking outside the
> # parameter space. Here's a trick.
> Red <- nlm(bwrapper,starteq,stepmax=1,xx=bdat) ; print(Red)
```

```
$minimum
[1] -122.1350

$estimate
[1] 1.980880

$gradient
[1] -1.721763e-07

$code
[1] 1

$iterations
[1] 3

>
> # G = -2 log(lambda) = -2 log( f(x|ThetaHat0) / f(x|ThetaHat0) )
> #   = -2 [ log( f(x|ThetaHat0) ) - log( f(x|ThetaHat) ) ]
> #   =  2 [ -log( f(x|ThetaHat0) ) - -log( f(x|ThetaHat) ) ]
>
> G <- 2 * (Red$minimum - Full$minimum)
> pval <- 1 - pchisq(G,1)
> cat("Chi-square = ",G,"   df = 1    p = ", pval, "\n")
Chi-square =  3.698825    df = 1    p =  0.0544508
>
```

```
>
> # Univariate Normal
>
> noglike <- function(theta,xx) # - log likelihood for normal
+     {
+     mu <- theta[1] ; sigsq <- theta[2] ; nn <- length(xx)
+     noglike <- ( nn*log(2*3.14159) + nn*log(sigsq) + sum((xx-mu)^2)/sigsq  )/2
+     noglike
+     }
> n <- 32
> datta <- rnorm(n,4,4)
> print(datta) # Always look at your data
 [1]  6.0708792  1.7203642  7.6776538  0.8065815  9.6825623  5.5192257 10.1815426
 [8]  6.8528688 -4.4985706  1.4869934  4.0063852  4.1263288  7.8512422  2.3970816
[15]  5.3738598  1.6216669  4.7168758 -5.0836198  5.1577467  3.5623270 13.0000002
[22]  3.4629611  1.3662684 -0.6605780  8.6537018 -6.8355418 -0.1510760  1.2349386
[29] 11.0410987  3.7648193  6.9024097  6.2700249

> # Get the estimated asymptotic variances exactly
> m <- mean(datta) ; v <- var(datta) ; vhat <- v * (n-1)/n  # MLE is biased
> cat("Mean = ",m,"   Variance = ", v, "   Vhat = ",vhat, "\n")
Mean =  3.977469    Variance =  20.62260    Vhat =  19.97814

> cat("Estimated asymptotic variances:  \n")
Estimated asymptotic variances:
> v1 <- vhat/n  ; v2 <- 2 * vhat^2 / n
> cat("For the mean = ",v1,"   For the variance = ", v2, "\n")
For the mean =  0.6243169    For the variance =  24.94538
> # Now get it numerically
> start1 <- c(1,1)
> Fullmod <- nlm(noglike,start1,hessian=T,xx=datta) ; print(Fullmod)
Warning messages:
1: NaNs produced in: log(x)
2: NA/Inf replaced by maximum positive value
3: NaNs produced in: log(x)
4: NA/Inf replaced by maximum positive value
$minimum
[1] 93.32024

$estimate
[1]  3.977463 19.978187

$gradient
[1] -6.406109e-06  2.279065e-06

$hessian
              [,1]          [,2]
[1,]  1.601747e+00 -1.546942e-05
[2,] -1.546942e-05  4.007117e-02

$code
[1] 1

$iterations
[1] 23

> # Estimated Fisher information is just the Hessian, so inverse of the Hession
> # is the estimated asymptotic variance-covariance matrix.
```

```
> print(solve(Fullmod$hessian))
             [,1]         [,2]
[1,] 0.6243183135 2.410172e-04
[2,] 0.0002410172 2.495560e+01
>

> # Back to the Beta: Wald Test of H0: alpha=beta
> # Get the Hessian this time

> Full2 <- nlm(bloglike,start1,hessian=T,xx=bdat) ; print(Full2)
$minimum
[1] -123.9845

$estimate
[1] 1.934915 2.041718

$gradient
[1]  2.495638e-05 -2.511951e-05

$hessian
          [,1]      [,2]
[1,]  386.5496 -285.6894
[2,] -285.6894  342.7270

$code
[1] 1

$iterations
[1] 8

> ThetaHat <- Full2$estimate
> V <- solve(Full2$hessian) # Estimated inverse of Fisher information
> n <- 1000
> CC <- t(as.matrix(c(1,-1))) ; print(CC)
     [,1] [,2]
[1,]    1   -1
> kore <- solve(CC %*% V %*% t(CC) )
> W <- as.numeric(t(CC %*% ThetaHat) %*% kore %*% (CC %*% ThetaHat))
> pval <- 1 - pchisq(W,1)
> cat("Wald Chi-square = ",W,"   df = 1    p = ", pval, "\n")
Wald Chi-square =  3.674431    df = 1    p =  0.05525311
>
```

The likelidood ratio chisquare was 3.698825