Beta Regression in S-Plus and R

Michael Smithson
School of Psychology, The Australian National University

(email: Michael.Smithson@anu.edu.au)

```
#This code is for S-PLUS (but the functions also run in R):
#Be sure to load the MASS library before using this.
#The IVs and DVs are defined in the four statements below.
#Substitute your dependent variable name for 'DV' in the ydata statement
#and
#your two sets of independent variables for 'IV, …'
#in the xdata and wdata statements.
##
ydata <- cbind(DV);
const <- rep(1,length(ydata));
xdata <- cbind(const, IV, …);
wdata <- cbind(const, IV, …);
##
#The initial starting values for the null model are
#generated by using the method of moments on ydata.
start <- c(log(mean(ydata)/(1-mean(ydata))), -log((mean(ydata) -
(mean(ydata))^2 - var(ydata))/var(ydata)))
##
#The betareg function computes the log-likelihood. It removes missing data.
##
betareg <- function(h, y, x, z)
{
      hx = x%*%h[1: length(x[1,])]
      mu = exp(hx)/(1+exp(hx))
      gz = z%*%h[length(x[1,])+1: length(z[1,])]
      phi = exp(-gz)
      loglik = lgamma(phi) - lgamma(mu*phi) – lgamma(phi – mu*phi) +
mu*phi*log(y) + (phi – mu*phi)*log(1 – y) – log(y) – log(1 – y)
      -sum(loglik, na.rm = TRUE)
}
##
#The grad function computes the gradient.
##
grad <- function(h, y, x, z)
{
      hx = x%*%h[1: length(x[1,])]
      gz = z%*%h[length(x[1,])+1: length(z[1,])]
      gd <- cbind(x*rep(exp(-gz+hx)*(log(y/(1-y)) + digamma(exp(-
gz)/(1+exp(hx)))- digamma(exp(-gz+hx)/(1+exp(hx))))/(1+exp(hx))^2,
length(x[1,])), -z*rep(exp(-gz)*(log(1-y) + exp(hx)*log(y) +
(1+exp(hx))*digamma(exp(-gz)) – digamma(exp(-gz)/(1+exp(hx)))-
exp(hx)*digamma(exp(-gz+hx)/(1+exp(hx))))/(1+exp(hx)), length(z[1,])))
      colSums(gd, na.rm = TRUE)
}
##
#This is the optimizer function:
##
betafit <- nlminb(start, betareg, x = xdata, y = ydata, z = wdata)
##
#These are the parameters, standard errors, z-stats and significance-
levels.
##
outfit <- rbind(estim <- betafit$parameters, serr <-
sqrt(diag(vcov.nlminb(betafit))), zstat <- estim/serr, prob <- 1-
```

```
pnorm(abs(zstat))); row.names(outfit) <- c("estim", "serr", "zstat",
"prob"); outfit
##
#These are the log-likelihood and convergence message, followed by the
gradient at the solution.
##
c(betafit$objective,betafit$message)
grad(betafit$parameters,ydata,xdata,zdata)
```

Example 2 from Smithson and Verkuilen (2005) in an S-Plus session

```
> attach(Example2)
> ydata <- cbind(Anxiety);
> const <- rep(1,length(ydata));
> start <- c(log(mean(ydata)/(1-mean(ydata))), -log((mean(ydata) -
(mean(ydata))^2 - var(ydata))/var(ydata)))
> start
[1] -2.299012 -1.313792
> #We enter the betareg and grad functions.
> betareg <- function(h, y, x, z)
+ {
+ hx = x%*%h[1: length(x[1,])]
+ mu = exp(hx)/(1+exp(hx))
+ gz = z%*%h[length(x[1,])+1: length(z[1,])]
+ phi = exp(-gz)
+     loglik = lgamma(phi) - lgamma(mu*phi) - lgamma(phi - mu*phi) +
mu*phi*log(y) + (phi - mu*phi)*log(1 - y) - log(y) - log(1 - y)
+     -sum(loglik, na.rm = TRUE)
+ }
> grad <- function(h, y, x, z)
+ {
+ hx = x%*%h[1: length(x[1,])]
+ gz = z%*%h[length(x[1,])+1: length(z[1,])]
+ gd <- cbind(x*rep(exp(-gz+hx)*(log(y/(1-y)) + digamma(exp(-
gz)/(1+exp(hx)))- digamma(exp(-gz+hx)/(1+exp(hx))))/(1+exp(hx))^2,
length(x[1,])), -z*rep(exp(-gz)*(log(1-y) + exp(hx)*log(y) +
(1+exp(hx))*digamma(exp(-gz)) - digamma(exp(-gz)/(1+exp(hx)))-
exp(hx)*digamma(exp(-gz+hx)/(1+exp(hx))))/(1+exp(hx)), length(z[1,])))
+ colSums(gd, na.rm = TRUE)
+ }
> #We start with the null model.
> xdata <- cbind(const); wdata <- cbind(const);
> betafit <- nlminb(start, betareg, x = xdata, y = ydata, z = wdata)
> #This is the negative log-likelihood and convergence message.
> c(betafit$objective,betafit$message)
[1] "-239.448005712066"            "RELATIVE FUNCTION CONVERGENCE"
> #These are the estimates, asymptotic standard errors and z-statistics.
> outfit <- rbind(estim <- betafit$parameters, serr <-
sqrt(diag(vcov.nlminb(betafit))), zstat <- estim/serr, prob <- 1-
pnorm(abs(zstat))); row.names(outfit) <- c("estim", "serr", "zstat",
"prob"); outfit
              x.1         x.2
estim  -2.24396066  -1.7956442
 serr   0.09757744   0.1240222
zstat -22.99671641 -14.4784088
 prob   0.00000000   0.0000000
> #Now we estimate a model with Stress in the location submodel.
> #We also specify new values for "start".
> xdata <- cbind(const, Stress); start <- c(-2.244, 0.1, -1.796);
> betafit <- nlminb(start, betareg, x = xdata, y = ydata, z = wdata)
> c(betafit$objective,betafit$message)
[1] "-283.006657335412"            "RELATIVE FUNCTION CONVERGENCE"
```

```
> #Check that the gradient is 0.
> grad(betafit$parameters,ydata,xdata,wdata)
       const        Stress        const
 1.5892e-006 8.478869e-007 1.07663e-006
> outfit <- rbind(estim <- betafit$parameters, serr <-
sqrt(diag(vcov.nlminb(betafit))), zstat <- estim/serr, prob <- 1-
pnorm(abs(zstat))); row.names(outfit) <- c("estim", "serr", "zstat",
"prob"); outfit
              x.1        x.2        x.3
estim  -3.4790186  3.7495930  -2.457410
 serr   0.1529186  0.3379374   0.126031
zstat -22.7507931 11.0955267 -19.498451
 prob   0.0000000  0.0000000   0.000000
> #Finally, we estimate a model that includes Stress in the dispersion
submodel.
> wdata <- cbind(const, Stress); start <- c(-3.479, 3.75, -2.46, 0.1);
> betafit <- nlminb(start, betareg, x = xdata, y = ydata, z = wdata)
> c(betafit$objective,betafit$message)
[1] "-301.9600873253"               "RELATIVE FUNCTION CONVERGENCE"
> grad(betafit$parameters,ydata,xdata,wdata)
         const        Stress          const         Stress
 8.977042e-007 2.552195e-007 -1.098784e-006 -2.781475e-007
> outfit <- rbind(estim <- betafit$parameters, serr <-
sqrt(diag(vcov.nlminb(betafit))), zstat <- estim/serr, prob <- 1-
pnorm(abs(zstat))); row.names(outfit) <- c("estim", "serr", "zstat",
"prob"); outfit
              x.1        x.2         x.3           x.4
estim  -4.0237159  4.9413672  -3.9608469 4.273312e+000
 serr   0.1438478  0.4480357   0.2121417 6.360746e-001
zstat -27.9720401 11.0289595 -18.6707588 6.718256e+000
 prob   0.0000000  0.0000000   0.0000000 9.195644e-012
```

#This code is for R (but the optimizer also runs in S-Plus via the MASS library).

```
#It's a good idea to load the MASS library before using this.
#You have to run the betareg and grad functions before this next step.
#This is the optimizer function using using BFGS
#(which is what we would recommend):
##
betaopt <- optim(start, betareg, hessian = T, x = xdata, y = ydata, z =
wdata, method = "BFGS")
##
#These are the parameters, standard errors, z-stats and significance-
levels.
##
outopt <- rbind(estim <- betaopt$par, serr <-
sqrt(diag(solve(betaopt$hessian))), zstat <- estim/serr, prob <- 1-
pnorm(abs(zstat))); row.names(outopt) <- c("estim", "serr", "zstat",
"prob"); outopt
##
#These are the log-likelihood, gradient, and convergence message.
##
betaopt$value
grad(betaopt$par,ydata,xdata,wdata)
betaopt$convergence
```

Example 3 from Smithson and Verkuilen (2005) in an R session

```
#This code shows how to obtain the final model in Example 3.
> ex3 <- read.table("Example3.txt", header = TRUE); attach(ex3)
> library(MASS)
> ydata <- cbind(accur01);
> const <- rep(1,length(ydata));
> xdata <- cbind(const, grpctr, ziq, grpctr*ziq);
> wdata <- cbind(const, grpctr, ziq);

[SNIP—FUNCTIONS ENTERED AS IN S-PLUS EXAMPLE]

> start <- c(1.0, -1.0, 0.5, -0.5, -3.0, -1.5, -1.5);
> betaopt <- optim(start, betareg, hessian = T, x = xdata, y = ydata, z =
wdata, method = "BFGS")
> c(betaopt$value, betaopt$convergence)
[1] -65.90186   0.00000
> grad(betaopt$par,ydata,xdata,wdata)
     const       grpctr          ziq                      const
grpctr          ziq
0.019330881  0.017862284  0.001995802 -0.002906731 -0.005810820
-0.005330392  0.003144999
> outopt <- rbind(estim <- betaopt$par, serr <-
sqrt(diag(solve(betaopt$hessian))), zstat <- estim/serr, prob <- 1-
pnorm(abs(zstat))); row.names(outopt) <- c("estim", "serr", "zstat",
"prob"); outopt
             [,1]          [,2]        [,3]          [,4]        [,5]          [,6]
estim 1.123240e+00 -7.417003e-01 0.486274215 -0.5811408452 -3.3040923 -1.746202e+00
serr  1.508897e-01  1.514561e-01 0.167118121  0.1726061937  0.2265501  2.940534e-01
zstat 7.444110e+00 -4.897130e+00 2.909763548 -3.3668597450 -14.5843774 -5.938384
prob  4.884981e-14  4.862333e-07 0.001808511  0.0003801467   0.0000000 1.439227e-09
              [,7]
estim -1.228857217
serr   0.459702865
zstat -2.673155443
prob   0.003757071
```

Example 1 from Smithson and Verkuilen (2005) in an R session

```
#This code shows how to obtain the final model in Example 1.
> ex1 <- read.table("Example1.txt", header = TRUE); attach(ex1)
> library(MASS)
> ydata <- cbind(crc99);
> const <- rep(1,length(ydata));
> xdata <- cbind(const, vert, confl, vert*confl);
> wdata <- cbind(const, vert, confl, vert*confl);

[SNIP—FUNCTIONS ENTERED AS IN S-PLUS EXAMPLE]

> start <- c(.88, -.01, .15, .19, -1.1, .34, -.22, .1);
> betaopt <- optim(start, betareg, hessian = T, x = xdata, y = ydata, z =
wdata, method = "BFGS")
> c(betaopt$value, betaopt$convergence)
[1] -40.11692   0.00000
> grad(betaopt$par,ydata,xdata,wdata)
        const          vert         confl                       const
vert
-1.177995e-04 -8.656216e-05 -1.401339e-04 -1.319796e-04 -5.783755e-05
-5.033170e-05
        confl
-4.503639e-05 -2.251637e-05
> outopt <- rbind(estim <- betaopt$par, serr <-
sqrt(diag(solve(betaopt$hessian))), zstat <- estim/serr, prob <- 1-
```

```
pnorm(abs(zstat))); row.names(outopt) <- c("estim", "serr", "zstat",
"prob"); outopt
         [,1]       [,2]       [,3]        [,4]       [,5]        [,6]        [,7]
estim 0.9124050 0.00503580 0.16857467 0.280011363 -1.1733126 0.329879607 -0.21960807
serr  0.1039788 0.10397881 0.10397881 0.103978810  0.1278120 0.127812019  0.12781202
zstat 8.7749129 0.04843102 1.62124063 2.692965659 -9.1799866 2.580974844 -1.71821143
prob  0.0000000 0.48068637 0.05248302 0.003540978  0.0000000 0.004926088  0.04287903
             [,8]
estim -0.31629474
serr   0.12781202
zstat -2.47468696
prob   0.00666765
```