Arizona State University

From the SelectedWorks of Joseph M Hilbe

September 18, 2011

Suppliment to Logistic Regression Models

Joseph Hilbe, *Arizona State University*

Available at: https://works.bepress.com/joseph_hilbe/26/

Supplement to *Logistic Regression Models*
Joseph M. Hilbe

Updated to 18 September, 2011

This document is intended to provide additional code, information, and updates to *Logistic Regression Models* (2009, Chapman & Hall/CRC). I will be adding to it on a periodical basis. Stata and R code may be copied and pasted into a text editor, and saved in ASCII or text format. Following the supplied code, which is displayed between double lines, I give sample output, which may come from the text.

A series of "0000000"'s is given for half the page between sections. I have many more additions to make, which I intend to do by the end of November, 2011.

Converting Stata files to R data frames

TASK: Create an R data frame from a Stata *dta* file. The *Hmisc* package can convert Stata files versions 5-10 to R files. Use the command *saveold* to convert version 11 and 12 Stata files to version10, which can be used by *Hmisc*.

```
. use heartlrm
. saveold heartlrm2
```

Below we assume that the Stata *heartlrm* file is stored in the c://ado directory. I save it to version 10 format, using a slightly different name. If I want to save this data as an R data frame in the c://rfiles directory, I can use the code below.

NOTE*:* *heartlrm2* is the same as *heart01*, but with no missing values and ordered in a meaningful way. *heartt01* is used frequently in *Logistic Regression Models*.

```
library("Hmisc")
heart01 <- stata.get("c://ado/heartlrm2.dta")
save.image(file="c://rfiles/heart01.Rdata")
head(heart01)    # Check to make certain the data is correct
```

You may convert any Stata file to an R data fame in this manner.

CHAPTER 7

R code for Hosmer-Lemeshow test

Credit: Contributed by Robert A. LaBudde (c) 2007

```
=========================================================================
iQuantile <- function (x, breaks=15) {
 #indices in x[] of percentile steps by 1/breaks
 xo <- order(x)  #sort indices
 n <- length(x)
 r <- rep(0, breaks+1)
 r[1]<- 1
 r[breaks+1]<- n
 r[2:breaks]<- round(1:(breaks-1)*n/breaks)
 return(list(index=r,cuts=x[xo[r]]))
}
 hlGOF.test <- function (observed, predicted, breaks=15) {
 #H-L GOF test for logistic regression
 #observed and predicted should not have missing values and match by index
 cat('\n', 'Hosmer-Lemeshow GOF test', '\n')
 ndata <- length(predicted)
 cuts <- c(round(.75*breaks), breaks, round(1.25*breaks))
 pvals <- rep(1,cuts[3]) #p-values
 for (nCuts in cuts) {
  ip <- order(predicted)
  iq <- iQuantile(predicted, nCuts) #indices for cuts
  iqInd<- iq$index
  cat('\n','For # Cuts =',nCuts,'  # Data =',ndata,'\n')
  cat('Cut  # Total #Patterns # Resp.    # Pred.  Mean Resp. Mean Pred.','\n')

  x2 <- 0
  ntot <- 0
  for (i in 1:nCuts) {
   if (i==1) {
    isubs <- ip[1:iqInd[2]]
   } else {
    isubs <- ip[(iqInd[i]+1):iqInd[i+1]]
   }
   nsubs <- length(isubs)
   ntot <- ntot + nsubs
   aobs <- mean(observed[isubs])
   mobs <- sum(observed[isubs])
   ncvp <- length(unique(predicted[isubs]))
   apred <- mean(predicted[isubs])
   mpred <- apred*nsubs
   x2 <- x2 + (mobs-mpred)^2/mpred + ((nsubs-mobs) - (nsubs-mpred))^2/(nsubs-mpred)
   cat(sprintf('%3d',i), sprintf('%8d', nsubs), sprintf('%8d', ncvp), sprintf('%8d', mobs),
    sprintf('%10.2f',mpred), sprintf('%8.5f', aobs), sprintf('%8.5f',apred), '\n')
  }
  cat('Total # Data:',ndata,' Total over cuts:',ntot,'\n')
  pvals[nCuts] <- pchisq(x2,nCuts-2,lower.tail=FALSE)
  cat('Chisq:', x2, '  d.f.:', sprintf('%d',nCuts-2), ' P-value:',
   sprintf('%8.5f', pvals[nCuts]),'\n')
 }
 cat('\n','Minimum P-value: ',sprintf('%8.5f',min(pvals)),'\n')
}
===============================================================================
```

OUTPUT: Chapter 7.2

```
> #SECTION 7.2: HOSMER-LEMESHOW GOF TEST

> fit7_2a<- glm(death ~ anterior + hcabg + kk2 + kk3 + kk4 +
+       age3 + age4, data=heart, family=binomial)

> summary(fit7_2a)
Call:
glm(formula = death ~ anterior + hcabg + kk2 + kk3 + kk4 + age3 +
age4, family = binomial, data = heart)

Deviance Residuals:
Min 1Q Median 3Q Max
-1.3724 -0.3257 -0.1745 -0.1270 3.1061

Coefficients:
               Estimate   Std. Error    z value    Pr(>|z|)
(Intercept)     -4.8159       0.1934    -24.896     < 2e-16 ***
anteriorAnterior 0.6387       0.1675      3.812    0.000138 ***
hcabg           0.7864       0.3527      2.229    0.025786 *
kk2             0.8249       0.1804      4.572    4.84e-06 ***
kk3             0.7967       0.2692      2.959    0.003083 **
kk4             2.6837       0.3565      7.529    5.11e-14 ***
age3            1.2668       0.2006      6.316    2.68e-10 ***
age4            1.9409       0.2080      9.329     < 2e-16 ***
---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1486.2 on 4502 degrees of freedom
Residual deviance: 1276.3 on 4495 degrees of freedom
(885 observations deleted due to missingness)
AIC: 1292.3

Number of Fisher Scoring iterations: 7

> exp(coef(fit7_2a)) #ORs
  (Intercept)  anteriorAnterior         hcabg           kk2
  0.008099558       1.894095884   2.195519114   2.281692192
          kk3               kk4          age3          age4
  2.218198595      14.639836908   3.549576517   6.964846706

> source('ralhlGOFtest.r') #macro for H-L GOF test
> heart2<- na.omit(heart) #drop rows with missing data

> hlGOF.test(heart2$death, predict(fit7_2a, heart2, type='response'), breaks=10)

Hosmer-Lemeshow GOF test
   For # Cuts = 8 # Data = 4503
   Cut # Total #Patterns # Resp. # Pred.  Mean Resp. Mean Pred.
    1   563      1         7        4.52  0.01243    0.00803
    2   563      1         2        4.52  0.00355    0.00803
    3   563      2         6        7.77  0.01066    0.01380
    4   563      4         6        8.87  0.01066    0.01575
    5   562      2         8       13.79  0.01423    0.02455
    6   563      7        25       21.07  0.04440    0.03742
    7   563      8        38       32.68  0.06750    0.05805
    8   563     24        84       82.77  0.14920    0.14702
Total # Data: 4503 Total over cuts: 4503
Chisq: 8.334831 d.f.: 6 P-value: 0.21458
```

```
  For # Cuts = 10 # Data = 4503
  Cut # Total #Patterns # Resp. # Pred.   Mean Resp. Mean Pred.
    1   450      1         6        3.62  0.01333    0.00803
    2   451      1         3        3.62  0.00665    0.00803
    3   450      2         3        4.47  0.00667    0.00994
    4   450      1         3        6.80  0.00667    0.01511
    5   451      4         6        7.17  0.01330    0.01591
    6   450      2         7       10.66  0.01556    0.02370
    7   450      4        12       13.95  0.02667    0.03099
    8   450      5        24       21.94  0.05333    0.04875
    9   451      7        43       31.90  0.09534    0.07073
   10   450     24        69       71.87  0.15333    0.15970
Total # Data: 4503 Total over cuts: 4503
Chisq: 10.59860 d.f.: 8 P-value: 0.22550
```

<other tables produced for 12, 6, and 4 groups. Look for consistency of p-values across tables>

000000000000000000000000000000000000000

CHAPTER 10

Synthetic Multinomial Logit Regression - R

```
===============================================================
library(MASS)
library(nnet)
x1 <- rnorm(50000)
x2 <- rnorm(50000)
x3 <- rnorm(50000)
denom = 1+exp(.4*x1 -.5*x2 +1 ) + exp(-.3*x1+.25*x2 +2) + ///
    exp(-.25*x1 + .1*x2 +2.5)
p1 <- 1/denom
p2 <- exp( .4*x1  - .5*x2  + 1)/denom
p3 <- exp(-.3*x1  + .25*x2 + 2)/denom
p4 <- exp(-.25*x1 + .1*x2  + 2.5)/denom
u <- runif(50000)
y <- rep(1, length(u)) #start with all level 1 u<= p1)
p12 <- p1 + p2
y <- ifelse(u>p1 & u<= p12, 2, y)
p13 <- p1 + p2 +p3
y <- ifelse (u>p12 & u<= p13, 3, y)
y <- ifelse(u>p13, 4, y)
mlogit <- multinom( y ~ x1 + x2 )
summary(mlogit)
================================================================
```

OUTPUT

Paste the above code into the New Script Editor in R's menu system and run. The output is displayed as below:

```
Coefficients:
  (Intercept)         x1          x2
2   0.9781753  0.4350747 -0.54372770
3   1.9781182 -0.2901101  0.22701225
4   2.4782944 -0.2052511  0.07339072

Std. Errors:
  (Intercept)         x1         x2
2  0.02652228 0.02556832 0.02576985
3  0.02349313 0.02378512 0.02391790
4  0.02289410 0.02319644 0.02332391

Residual Deviance: 107575.2
AIC: 107593.2
```

000000000000000000000000000000000000000

Synthetic Multinomial Logit Regression - Stata

```
============================================================
qui {
  clear
  set mem 50m
  set obs 50000
  gen x1 =  invnorm(runiform())
  gen x2 =  invnorm(runiform())
}
  /* coefficients for y= 2 are .4,  -.5 and 1
     coefficients for y= 3 are -.3, .25 and 2
     coefficients for y= 4 are -.25, .10 and 2.5 */
 qui {
  gen denom = 1+exp(.4*x1 -.5*x2 +1 ) + exp(-.3*x1+.25*x2 +2) + ///
       exp(-.25*x1 + .1*x2 +2.5)
  gen p1 = 1/denom
  gen p2 = exp(.4*x1-.5*x2 + 1) / denom
  gen p3 = exp(-.3*x1+.25*x2 + 2) / denom
  gen p4 = exp(-.25*x1+.1*x2 + 2.5) / denom
  gen u = runiform()
  gen y = 1 if u <= p1
  gen p12 = p1 + p2
  replace y = 2 if y==. & u<=p12
  gen p13 = p1 +p2 + p3
  replace y = 3 if y==. & u<=p13
  replace y = 4 if y==.
 }
============================================================
. do synmultinom4   /* 4 levels */
. mlogit y x1 x2 ,  baseoutcome(1) nolog

Multinomial logistic regression            Number of obs   =      50000
                                           LR chi2(6)      =    4979.77
                                           Prob > chi2     =     0.0000
Log likelihood = -53562.977                Pseudo R2       =     0.0444
---------------------------------------------------------------------
    y |      Coef.   Std. Err.      z    P>|z|   [95% Conf. Interval]
------+--------------------------------------------------------------
1     |  (base outcome)
------+--------------------------------------------------------------
2     |
   x1 |    .430633   .0259663    16.58   0.000     .37974     .481526
   x2 |  -.4866157    .025879   -18.80   0.000  -.5373375   -.4358939
_cons |   .9732471   .0267927    36.33   0.000   .9207345     1.02576
------+--------------------------------------------------------------
3            |
   x1 |  -.2740491   .0240936   -11.37   0.000  -.3212717   -.2268265
   x2 |   .2761934   .0240118    11.50   0.000   .2291312    .3232557
_cons |   2.004453   .0236947    84.59   0.000   1.958012    2.050893
------+--------------------------------------------------------------
4     |
   x1 |  -.2312814   .0235239    -9.83   0.000  -.2773875   -.1851753
   x2 |    .122648   .0234227     5.24   0.000   .0767404    .1685557
 cons |   2.500548   .0231124   108.19   0.000   2.455249    2.545848
---------------------------------------------------------------------
```

000000000000000000000000000000000000000

Convert Observation-level Multinomial Data to Grouped

```
=======================================================================
clear
use medpar
keep type died white hmo
mlogit type white died hmo, rrr
egen cp = group(died white hmo)
sort type cp
gen n = 1
by type cp: gen cnt = sum(n)
sort type cp
by type cp: keep if _n==_N
drop cp n
mlogit type died white hmo [fw=cnt], rrr
=======================================================================
. do multinom_obs_grp
```

OBSERVATION-LEVEL DATA

```
. mlogit type white died hmo, rrr
---------------------------------------------------------------------------
           type |      RRR   Std. Err.    z    P>|z|   [95% Conf. Interval]
----------------+----------------------------------------------------------
Elective_Admit  |(base outcome)
----------------+----------------------------------------------------------
Urgent_Admit    |
          white | .4689032   .1000123  -3.55   0.000   .3086957    .7122555
           died | 1.404759   .1997158   2.39   0.017   1.063129     1.85617
            hmo | .6174288   .1279701  -2.33   0.020   .4113053      .92685
          _cons | .4388146   .0898531  -4.02   0.000   .2937556    .6555047
----------------+----------------------------------------------------------
Emergency_Admit |
          white | .6629189   .2366041  -1.15   0.249   .3293487    1.334335
           died | 1.905411   .4113054   2.99   0.003   1.248093    2.908912
            hmo | .1470197   .0870809  -3.24   0.001   .0460471    .4694064
          _cons | .1130222   .0392496  -6.28   0.000   .0572222    .2232357
---------------------------------------------------------------------------
```

GROUPED FORMAT

```
. mlogit type white died hmo [fw=cnt], rrr
---------------------------------------------------------------------------
           type |      RRR   Std. Err.      z    P>|z|   [95% Conf. Interval]
----------------+----------------------------------------------------------
Elective_Admit  |(base outcome)
----------------+----------------------------------------------------------
Urgent_Admit    |
          white | .4689032   .1000123  -3.55   0.000   .3086957    .7122555
           died | 1.404759   .1997158   2.39   0.017   1.063129     1.85617
            hmo | .6174288   .1279701  -2.33   0.020   .4113053      .92685
          _cons | .4388146   .0898531  -4.02   0.000   .2937556    .6555047
----------------+----------------------------------------------------------
Emergency_Admit |
          white | .6629189   .2366041  -1.15   0.249   .3293487    1.334335
           died | 1.905411   .4113054   2.99   0.003   1.248093    2.908912
            hmo | .1470197   .0870809  -3.24   0.001   .0460471    .4694064
          _cons | .1130222   .0392496  -6.28   0.000   .0572222    .2232357
---------------------------------------------------------------------------
```

0000000000000000000000000000000000000

SYNTHETIC LOGIT and PROBIT REGRESSION

R code

SYNTHETIC LOGISTIC REGRESSION

```
nobs <- 5000
x1  <- runif(nobs)
x2  <- runif(nobs)
xb  <- 2 + 0.75*x1 - 1.25*x2
exb <- 1/(1+exp(-xb))
by  <- rbinomial(nobs, size=1, p=exb)
lreg <- glm(by ~ x1 + x2, family=binomial(link="logit"))
summary(lreg)
```

SYNTHETIC PROBIT REGRESSION

```
nobs <- 5000
x1  <- runif(nobs)
x2  <- runif(nobs)
xb  <- 2 + 0.75*x1 - 1.25*x2
exb <- pnorm(xb)
py  <- rbinomial(nobs, size=1, p=exb)
preg <- glm(py ~ x1 + x2, family=binomial(link="probit"))
summary(preg)
```

SYNTHETIC BINOMIAL OR GROUPED LOGISTIC REGRESSION

```
nobs <- 5000
x1  <- runif(nobs)
x2  <- runif(nobs)
d   <- rep(1:5, each=10000, times=1) *100
xb  <- 2 + 0.75*x1 - 1.25*x2
exb <- 1/(1+exp(-xb))
by  <- rbinomial(nobs, size=d, p=exb)
dby <- d - by
mylogit <- data.frame(by, dby, x1, x2)
gby <- glm(cbind(by,dby) ~ x1 + x2, family=binomial(link="logit"), data=mylogit)
summary(gby)

or
library(msme)   # forthcoming
mylogit <- data.frame(by, d, x1, x2)
lreg <- irls(by ~ x1 + x2, family=binomial, m=d, data=mylogit)
summary(lreg)
```

# MONTE CARLO LOGISTIC REGRESSION

```
# mc.logit.r // Joseph Hilbe  31Jul 2011 100 replications
# Displays vectors of coefficients and of standard errors
logitmc <- function()
{
  nobs <- 50000
  x1 <- runif(nobs)
  x2 <- runif(nobs)
  xb <- 2 + .75*x1 - 1.25*x2
  exb <- 1/(1+exp(-xb))
  by <- rbinom(nobs, size = 1, prob =exb)
  lry <- glm(by ~ x1 + x2, family=binomial(link="logit"))
    beta <- lry$coef
    pr <- sum(residuals(lry, type="pearson")^2)
    prdisp <- pr/lry$df.residual
    return(list(coef = coef(lry),
            se = sqrt(diag(vcov(lry)))/ sqrt(prdisp)))
}
B <- replicate(100, logitmc())
apply(matrix(unlist(B[1,]),3,100),1,mean)
apply(matrix(unlist(B[2,]),3,100),1,mean)
```

# MONTE CARLO GROUPED LOGISTIC REGRESSION

```
# mc.glogit.r // Joseph Hilbe  31Jul 2011
glogitmc <- function()
{
  nobs <- 50000
  x1 <- runif(nobs)
  x2 <- runif(nobs)
  d <- rep(1:5, each=10000, times=1)*100    # denominator
  xb <- 2 + .75*x1 - 1.25*x2
  exb <- 1/(1+exp(-xb))
  by <- rbinom(nobs, size = d, p = exb)
  dby = d - by
  gby <- glm(cbind(by,dby) ~ x1 + x2, family=binomial(link="logit"))
    beta <- gby$coef
    pr <- sum(residuals(gby, type="pearson")^2)
    prdisp <- pr/gby$df.residual
    se <- sqrt(diag(vcov(gby)))/prdisp
    list(beta,se)
}
B <- replicate(100, glogitmc())
apply(matrix(unlist(B[1,]),3,100),1,mean)
mean(unlist(B[2,]))
```

SYNTHETIC LOGIT and PROBIT REGRESSION

Stata code

SYNTHETIC BERNOULLI-LOGIT DATA

```
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 13579
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
gen xb = 2 + 0.75*x1 - 1.25*x2
gen exb = 1/(1+exp(-xb))
gen by = rbinomial(1, exb)
glm by x1 x2, nolog fam(bin 1)
```

SYNTHETIC BERNOULLI-PROBIT DATA

```
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 19375
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
gen xb = 2 + 0.75*x1 - 1.25*x2
gen double exb = normprob(xb)
* replace exb=.99999999 if exb>.99999999  // add if need 50000 obs
gen double py = rbinomial(1, exb)
glm py x1 x2, nolog fam(bin 1) link(probit)
```

SYNTHETIC BINOMIAL-LOGIT DATA

```
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 13579
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
* ================================================
* Select one: user specified or random  denominator
* gen d = 100+100*int((_n-1)/10000)
gen d = ceil(10*runiform())
* ================================================
gen xb = 2 + 0.75*x1 - 1.25*x2
gen exb = 1/(1+exp(-xb))
gen by = rbinomial(d, exb)
glm by x1 x2, nolog fam(bin d)
```

* SYNTHETIC ORDERED LOGIT WITH 3 LEVELS

```
di in ye "Defined Coefficients/cuts:"
di in ye "b1 = .75; b2 = 1.25"
di in ye "Cut1=2; Cut2=3; Cut3=4"
qui {
drop _all
set obs 50000
set seed 13444
gen double x1 = 3*uniform()+1
gen double x2 = 2*uniform()-1
gen err = uniform()
gen y = .75*x1 + 1.25*x2 + log(err/(1-err))
gen int ys = 1 if y<=2
replace ys=2 if y<=3 & y>2
replace ys=3 if y<=4 & y>3
replace ys=4 if  y>4
* save syn_logit
}
ologit ys x1 x2, nolog
* predict double (olpr1 olpr2 olpr3 olpr4), pr
```

* SYNTHETIC ORDERED LOGIT MODEL WITH 4 LEVELS

```
qui {
drop _all
set obs 50000
set seed 13444
gen double x1 = 3*runiform()+1
gen double x2 = 2*runiform()-1
gen double x3 = runiform()
gen err = runiform()
gen y = .5*x1 + 1.75*x2 - 1.25*x3 + log(err/(1-err))
gen int ys = 1 if y<=.8
replace ys=2 if y<=1.6 & y>.8
replace ys=3 if y<=2.4 & y>1.6
replace ys=4 if y<=3.2 & y>2.4
replace ys=5 if  y>3.2
* save syn_logit
}
ologit ys x1 x2 x3, nolog
predict double (olpr1 olpr2 olpr3 olpr4 olpr5), pr
noi di in ye "Defined Coefficients/cuts:"
noi di in ye "b1 = .5; b2 = 1.75; b3 - -1.25"
noi di in ye "Cut1=.8; Cut2=1.6; Cut3=2.4; Cut4=3.2"
```

* SYNTHETIC BINOMIAL-PROBIT DATA AND MODEL

```
* x1=.75, x2=-1.25, _cons=2
clear
set obs 50000
set seed 070785
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
* ================================================
* Select one: user specified or random denominator
* gen d = 100+100*int((_n-1)/10000)
gen d = ceil(10*runiform())
* ================================================
gen xb = 2 + 0.75*x1 - 1.25*x2
gen double exb = normprob(xb)
* replace exb=.99999999 if exb>.99999999 and want exactly 50000
gen double by = rbinomial(d, exb)
glm by x1 x2, nolog fam(bin d) link(probit)
```

* SYNTHETIC ORDERED PROBIT DATA AND MODEL

```
di in ye "b1 = .75; b2 = 1.25"
di in ye "Cut1=2; Cut2=3,; Cut3=4"
qui {
drop _all
set obs 50000
set seed 12345
gen double x1 = 3*uniform()+1
gen double x2 = 2*uniform()-1
gen double y = .75*x1 + 1.25*x2 + invnorm(uniform())
gen int ys = 1 if y<=2
replace ys=2 if y<=3 & y>2
replace ys=3 if y<=4 & y>3
replace ys=4 if  y>4
* save syn_probit
}
oprobit ys x1 x2, nolog
* predict double (olpr1 olpr2 olpr3 olpr4), pr
```

* SYNTHETIC MULTINOMIAL LOGIT MODEL

```
* y=2: x1= 0.4, x2=-0.5, _cons=1.0
* y=3: x1=-3.0, x2=0.25, _cons=2.0
qui {
  clear
  set mem 50m
*  set seed 111322
  set obs 100000
  gen x1 = runiform()
  gen x2 = runiform()
  gen denom = 1+exp(.4*x1 - .5*x2 +1 ) + exp(-.3*x1+.25*x2 +2)
  gen p1 = 1/denom
  gen p2 = exp(.4*x1-.5*x2 + 1) / denom
  gen p3 = exp(-.3*x1+.25*x2 + 2) / denom
  gen u = runiform()
  gen y = 1 if u <= p1
  gen p12 = p1 + p2
  replace y = 2 if y==. & u<=p12
  replace y = 3 if y==.
}
  mlogit y x1 x2,  baseoutcome(1) nolog
```