September 2011

Negative Binomial Regression Extensions

Negative Binomial Regression Extensions:

Code, Functions, Derivations, and Examples

Joseph M. Hilbe

Jet Propulsion Laboratory
California Institute of Technology

and

Arizona State University

18 September, 2011 update

NOTE TO READER:

This text is intended as an electronic book, to be used as an ancillary text to **Hilbe, Joseph M. (2011), *Negative Binomial Regression*, second edition, Cambridge University Press**. It consists of additional code, derivations, functions, commands, and examples to the material that is in the aforementioned text. It also consists of updates to items in the book which have undergone enhancement.

I provide code and examples using Stata, R, SAS, LIMDEP, and Winbugs. The majority of examples in the text use Stata, with R code shown in associated tables as well as in the text. The examples used in Chapter 15 on Bayesian modeling employs SAS STAT/GENMOD due to its ease of use compared to R and Winbugs – particularly for more elementary models. Stata currently has no Bayesian modeling capability.

SAS code is given in this text to assist SAS users who wish to model count data. Wherever possible I give SAS code to duplicate the modeling examples in the text that use Stata and R.

This work will continually evolve, so check the edition and date of preparation. I suggest referring to the most current version.

Readers are encouraged to send me any related code, or discussion, that they have developed. If appropriate, I will include their work in the text, with a proper acknowledgement. The goal of the published text, as well as this electronic ancillary text, is to provide researchers with a reliable source of information and code for the modeling of count data in all of its varieties.

R functions, scripts, and data used in both the published and electronic texts are located in the **COUNT** package (capital letters), which is posted to CRAN. I will also attempt to keep it as up-to-date as possible.

Corrections, additions, and suggestions should be addressed to me at hilbe@asu.edu, jhilbe@aol.com, or j.m.hilbe@gmail.com.

Joseph Hilbe
7242 W. Heritage Way
Florence, AZ 85132 USA

Edition 1
Printing: 1
Date: 1 March, 2011.

TABLE OF CONTENTS

Part 1: COUNT, R package download from CRAN

A: R data sets and functions

DATA FILES (stored as data frames) type: > data()

affairs	azdrg112	azpro	badhealth
fasttrakg	lbw	lbwgrp	loomis
mdvis	medpar	rwm	rwm5yr
ships	titanic	titgrp	

FUNCTIONS type > ls("package:COUNT") U= latest updated code D/M/Yr

COUNT::nb2.obs.pred	Table of negative binomial counts: observed vs predicted proportions and difference
COUNT::poi.obs.pred	Table of Poisson counts: observed vs predicted proportions and difference
COUNT::ml.pois	Maximum likelihood Poisson
COUNT::ml.nb1	NB1: maximum likelihood linear negative binomial regression U12/07/2011
COUNT::ml.nb2	NB2: maximum likelihood standard negative binomial regression U12/07/2011
COUNT::ml.nbc	NBC: maximum likelihood canonical negative binomial regression
COUNT::myTable	
COUNT::modelfit	Fit Statistics for generalized linear models
COUNT::logit_syn	Synthetic logit data and models
COUNT:: probit_syn	Synthetic probit data and models
COUNT:: poisson_syn	Synthetic Poisson data and models
COUNT:: nb1_syn	Synthetic NB1 data and models
COUNT:: nb2_syn	Synthetic NB2 data and models
COUNT:: nbc_syn	Synthetic NB-C data and models

SCRIPTS

aic	binegbin_new	geo_rng
hilbe.obspred	mysim	nb.reg.ml
nbc.reg.ml	nbr_6_2_2	nbr_6_2_3
nbr2_7_1	nbsim	p.reg.ml
syn.bin.logit	suyn.cgeo	syn.geo
syn.hurdle_lnb2	syn.logit	syn.nb1
syn.nb2	syn.nb2nb2fm	syn.nb2o
syn.nbc	syn.poisson	syn.poissono
synppfm	syn.probit	ZANBI
ZAP	ZIBI	

FIGURES

hilbe.NBR2.F3.1	hilbe.NBR2.F6.1	hilbe.NBR2.F6.2
hilbe.NBR2.F6.2alt	hilbe.NBR2.F6.3	hilbe.NBR2.F6.4
hilbe.NBR2.F8.1	hilbe.NBR2.F8.2	hilbe.NBR2.F8.3
hilbe.NBR2.F8.4	hilbe.NBR2.F8.5	hilbe.NBR2.F8.6

hilbe.NBR2.F8.7	hilbe.NBR2.F8.8	hilbe.NBR2.F8.9
hilbe.NBR2.F8.10	hilbe.NBR2.F8.11	hilbe.NBR2.F8.12
hilbe.NBR2.F8.13	hilbe.NBR2.F9.1	hilbe.NBR2.F9.1alt
hilbe.NBR2.F9.2	hilbe.NBR2.F9.2alt	hilbe.NBR2.F9.3
hilbe.NBR2.F9.4	hilbe.NBR2.F9.5	hilbe.NBR2.F9.6ab
hilbe.NBR2.F10.1	hilbe.NBR2.F11.1	hilbe.NBR2.F14.1

NOTE:
ml.nb2 nd ml.nb1 functions amended 12Jul2011 to include an offset option for rate parameterized models. CRAN updated same day. Postestimation:

modelname$Estimate	vector of coefficients displayed
modelname$SE	vector of standard errors displayed
modelname$Z	vector of Z-statistics displayed
modelname$LCL	vector of Lower Confidence Levels displayed
modelname$UCL	vector of Upper Confidence Levels displayed

Estimates and confidence levels may be exponentiated. Estimates become Incidence rate ratios
exp(*modelname*$Estimate)
exp(*modelname*$LCL)
exp(*modelname*$UCL)

B: Converting Stata files to R data frames

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*. Below we assume that the Stata *heartlrm2* file is stored in the c://ado directory. 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 my *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.

Part 2: Stata / R code for creating Figures in Negative Binomial Regression, 2nd edition

A: Stata code

Note: Stata data sets are assumed to reside in the c:\\source\ directory. Amend as needed. Chapter 15 has two Figures. Both were created using SAS code, which is provided here.

```
* ================================================================
* CHAPTER 3
* FIGURE 3.1
clear
set obs 11
gen byte mu = 2
gen byte y = _n-1
gen yp2 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))
gen alpha = 1.5
gen amu = mu*alpha
gen ynb2 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha)-
  lngamma(y+1) - lngamma(1/alpha))
label var yp2 "Poisson: mean=2"
label var ynb2 "Negative Binomial: mean=2, a=1.5"
graph twoway connected ynb2 yp2  y, ms(T d) title(Poisson vs Negative Binomial
   PDFs)

* ================================================================
* CHAPTER 6

* FIGURE 6.1
drop _all
set obs 50000
gen x1 = rnormal()
gen x2 = rnormal()
gen py = rpoisson(exp(2 + 0.75*x1 - 1.25*x2))
sum py
gen double ypoi = (exp(-r(mean))*r(mean)^py)/exp(lngamma(py+1))
graph twoway scatter ypoi py if py<50, title(Synthetic Model: Mean = 21.35)
  xline(21.35) c(0) sort

* ================================================================

* FIGURE 6.2
clear
use c:\\source\medpar
qui poisson los hmo white type2 type3
qui  {
predict mu
local i 0
local newvar "pr`i'"
while `i' <=25 {
 local newvar "pr`i'"
 qui gen `newvar' =  exp(-mu)*(mu^`i')/exp(lnfactorial(`i'))
 local i = `i' + 1
}
quietly gen cnt = .
quietly gen obpr = .
```

```
quietly gen prpr = .
local i 0
while `i' <=25 {
 local obs = `i' + 1
 replace cnt = `i' in `obs'
 tempvar obser
 gen `obser' = (`e(depvar)'==`i')
 sum `obser'
 replace obpr = r(mean) in `obs'
 sum pr`i'
 replace prpr = r(mean) in `obs'
 local i = `i' + 1
}
gen obsprop = obpr*100
gen preprop = prpr*100
gen byte count = cnt
gen diffprop = obsprop - preprop
format obsprop preprop diffprop %8.3f
label var prpr "Predicted days"
label var obpr "Observed days"
label var count "Days in Hospital"
}
twoway scatter prpr obpr count, c(l l) ms(T d) title(Observed vs Predicted days)
ytitle(Probability of LOS)

* ================================================================

* FIGURE 6.3
clear
set obs 20
gen y = _n-1
gen mu = .
gen mu0_5 = (exp(-.5)* .5^y)/exp(lngamma(y+1))
forvalues i = 1(2)9 {
  gen mu`i' = (exp(-`i')*`i'^y)/exp(lngamma(y+1))
}
graph twoway connected mu0_5 mu1 mu3 mu5  mu7 mu9 y, ms(T s o D X O) title("Poisson
Distributions")

* ================================================================

* FIGURE 6.4
clear
use rwm5yr, clear
qui poisson docvis outwork age female married edlevel2-edlevel4
gen rest = _b[_cons] + _b[female]*.4804937 + _b[married]*.7736244 +
    _b[edlevel2]*.0587995 + _b[edlevel3]*.0883778 + _b[edlevel4]*.0657861
gen L1 = _b[outwork]*1 + _b[age]*age + rest
gen L0 = _b[outwork]*0 + _b[age]*age + rest
gen eout1 = exp(L1)
gen eout0 = exp(L0)
lab var eout1 "Not Working"
lab var eout0 "Working"
  graph twoway connected eout1 eout0 age, c(l l) ms(T d)
      title("Predicted visits to doctor: age on outwork levels") sort
* ================================================================

* ================================================================
* Conditional effects plot - probabilities; following Fig 6.4
use c:\\source\rwm5yr
qui poisson docvis outwork age female married edlevel2-edlevel4
prgen age, x(outwork=0) rest(mean) from(25) to(64) gen(pr1) n(40)
prgen age, x(outwork=1) rest(mean) from(25) to(64) gen(pr2) n(40)
```

```
lab var pr1p0 "Not Working"
lab var pr2p0 "Working"
lab var pr2x "Age"
graph twoway connected pr1p0 pr2p0 pr2x, ms(T d) title(Probability of doctor visits:
work vs not working)

* ========================================================================
* CHAPTER 8

* FIGURE 8.1

* NB2 MEANS: 0.5, 1,2,5,10; Alpha=0
set obs 11
gen mu = .5
gen byte y = _n-1
gen yp05 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))
* graph twoway connected yp05 y,

gen byte mu1 = 1
gen yp1 =  (exp(-mu1)*mu1^y)/exp(lngamma(y+1))
* graph twoway connected yp1 y,

gen byte mu2 = 2
gen yp2 =  (exp(-mu2)*mu2^y)/exp(lngamma(y+1))
* graph twoway connected yp2 y,

gen byte mu3 = 5
gen yp5 =  (exp(-mu3)*mu3^y)/exp(lngamma(y+1))
* graph twoway connected yp5 y,

gen byte mu4 = 10
gen yp10 =  (exp(-mu4)*mu4^y)/exp(lngamma(y+1))
* graph twoway connected yp10 y,

lab var yp05 "P: mean=.5"
lab var yp1  "P: mean=1"
lab var yp2  "P: mean=2"
lab var yp5  "P: mean=5"
lab var yp10 "P: mean=10"
graph twoway connected yp05 yp1 yp2 yp5 yp10  y, ms(T s o D X) title("NEGATIVE
BINOMIAL DISTRIBUTIONS: ALPHA = 0")


* ========================================================

* FIGURE 8.2
* NB2 MEANS: 0.5, 1,2,5,10; Alpha=0.33
set obs 11
gen byte y = _n-1
gen alpha = .333

* MEAN = .5
gen mu = .5
gen amu = mu*alpha
gen ynb05 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 1
gen byte mu1 = 1
gen amu1 = mu1*alpha
gen ynb1 = exp(y*ln(amu1/(1+amu1)) - (1/alpha)*ln(1+amu1) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))
```

```
* MEAN = 2
gen byte mu2 = 2
gen amu2 = mu2*alpha
gen ynb2 = exp(y*ln(amu2/(1+amu2)) - (1/alpha)*ln(1+amu2) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 5
gen byte mu3 = 5
gen amu3 = mu3*alpha
gen ynb5 = exp(y*ln(amu3/(1+amu3)) - (1/alpha)*ln(1+amu3) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 10
gen byte mu4 = 10
gen amu4 = mu4*alpha
gen ynb10 = exp(y*ln(amu4/(1+amu4)) - (1/alpha)*ln(1+amu4) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

lab var ynb05 "NB: mean=.5"
lab var ynb1  "NB: mean=1"
lab var ynb2  "NB: mean=2"
lab var ynb5  "NB: mean=5"
lab var ynb10 "NB: mean=10"

graph twoway connected ynb05 ynb1 ynb2 ynb5 ynb10  y, ms(T s o D X) title("NEGATIVE
BINOMIAL DISTRIBUTIONS: ALPHA = .33")

* ================================================================

* FIGURE 8.3
* NB2 MEANS: 0.5, 1,2,5,10; Alpha=0.67
set obs 11
gen byte y = _n-1
gen alpha = .667

* MEAN = .5
gen mu = .5
gen amu = mu*alpha
gen ynb05 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 1
gen byte mu1 = 1
gen amu1 = mu1*alpha
gen ynb1 = exp(y*ln(amu1/(1+amu1)) - (1/alpha)*ln(1+amu1) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 2
gen byte mu2 = 2
gen amu2 = mu2*alpha
gen ynb2 = exp(y*ln(amu2/(1+amu2)) - (1/alpha)*ln(1+amu2) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 5
gen byte mu3 = 5
gen amu3 = mu3*alpha
gen ynb5 = exp(y*ln(amu3/(1+amu3)) - (1/alpha)*ln(1+amu3) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 10
gen byte mu4 = 10
gen amu4 = mu4*alpha
```

```
gen ynb10 = exp(y*ln(amu4/(1+amu4)) - (1/alpha)*ln(1+amu4) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

lab var ynb05 "NB: mean=.5"
lab var ynb1  "NB: mean=1"
lab var ynb2  "NB: mean=2"
lab var ynb5  "NB: mean=5"
lab var ynb10 "NB: mean=10"

graph twoway connected ynb05 ynb1 ynb2 ynb5 ynb10  y, ms(T s o D X) title("NEGATIVE
BINOMIAL DISTRIBUTIONS: ALPHA = .67")


* ================================================================


* FIGURE 8.4
* NB2 MEANS: 0.5, 1,2,5,10; Alpha=1
clear
set obs 11
gen byte y = _n-1
gen alpha = 1

* MEAN = .5
gen mu = .5
gen amu = mu*alpha
gen ynb05 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 1
gen byte mu1 = 1
gen amu1 = mu1*alpha
gen ynb1 = exp(y*ln(amu1/(1+amu1)) - (1/alpha)*ln(1+amu1) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 2
gen byte mu2 = 2
gen amu2 = mu2*alpha
gen ynb2 = exp(y*ln(amu2/(1+amu2)) - (1/alpha)*ln(1+amu2) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 5
gen byte mu3 = 5
gen amu3 = mu3*alpha
gen ynb5 = exp(y*ln(amu3/(1+amu3)) - (1/alpha)*ln(1+amu3) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 10
gen byte mu4 = 10
gen amu4 = mu4*alpha
gen ynb10 = exp(y*ln(amu4/(1+amu4)) - (1/alpha)*ln(1+amu4) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

lab var ynb05 "NB: mean=.5"
lab var ynb1  "NB: mean=1"
lab var ynb2  "NB: mean=2"
lab var ynb5  "NB: mean=5"
lab var ynb10 "NB: mean=10"

graph twoway connected ynb05 ynb1 ynb2 ynb5 ynb10  y, ms(T s o D X) title("NEGATIVE
BINOMIAL DISTRIBUTIONS: ALPHA = 1")

* ================================================================
```

```
* FIGURE 8.5
* NB2 MEANS: 0.5, 1,2,5,10; Alpha 1.5
clear
set obs 11
gen byte y = _n-1
gen alpha = 1.5

* MEAN = .5
gen mu = .5
gen amu = mu*alpha
gen ynb05 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 1
gen byte mu1 = 1
gen amu1 = mu1*alpha
gen ynb1 = exp(y*ln(amu1/(1+amu1)) - (1/alpha)*ln(1+amu1) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 2
gen byte mu2 = 2
gen amu2 = mu2*alpha
gen ynb2 = exp(y*ln(amu2/(1+amu2)) - (1/alpha)*ln(1+amu2) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 5
gen byte mu3 = 5
gen amu3 = mu3*alpha
gen ynb5 = exp(y*ln(amu3/(1+amu3)) - (1/alpha)*ln(1+amu3) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 10
gen byte mu4 = 10
gen amu4 = mu4*alpha
gen ynb10 = exp(y*ln(amu4/(1+amu4)) - (1/alpha)*ln(1+amu4) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

lab var ynb05 "NB: mean=.5"
lab var ynb1  "NB: mean=1"
lab var ynb2  "NB: mean=2"
lab var ynb5  "NB: mean=5"
lab var ynb10 "NB: mean=10"

graph twoway connected ynb05 ynb1 ynb2 ynb5 ynb10  y, ms(T s o D X) title("NEGATIVE
BINOMIAL DISTRIBUTIONS: ALPHA = 1.5")


* ================================================================================

* FIGURE 8.6
* NB2 MEANS: 0.5, 1,2,5,10; Alpha=3
clear
set obs 11
gen byte y = _n-1
gen alpha = 3

* MEAN = .5
gen mu = .5
gen amu = mu*alpha
gen ynb05 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))
```

```
* MEAN = 1
gen byte mu1 = 1
gen amu1 = mu1*alpha
gen ynb1 = exp(y*ln(amu1/(1+amu1)) - (1/alpha)*ln(1+amu1) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 2
gen byte mu2 = 2
gen amu2 = mu2*alpha
gen ynb2 = exp(y*ln(amu2/(1+amu2)) - (1/alpha)*ln(1+amu2) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 5
gen byte mu3 = 5
gen amu3 = mu3*alpha
gen ynb5 = exp(y*ln(amu3/(1+amu3)) - (1/alpha)*ln(1+amu3) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* MEAN = 10
gen byte mu4 = 10
gen amu4 = mu4*alpha
gen ynb10 = exp(y*ln(amu4/(1+amu4)) - (1/alpha)*ln(1+amu4) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

lab var ynb05 "NB: mean=.5"
lab var ynb1  "NB: mean=1"
lab var ynb2  "NB: mean=2"
lab var ynb5  "NB: mean=5"
lab var ynb10 "NB: mean=10"

graph twoway connected ynb05 ynb1 ynb2 ynb5 ynb10  y, ms(T s o D X) title("NEGATIVE
BINOMIAL DISTRIBUTIONS: ALPHA = 3")

* ================================================================================

* FIGURE 8.7
* NB2 ALPHA: 0, .33, .67, 1, 1.5, 3; Mean=0.5
clear
set obs 11
gen mu = .5
gen byte y = _n-1
gen ynb0 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))

* NB ALPHA .33
gen alpha = .333
gen amu = mu*alpha
gen ynb33 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* NB ALPHA .67
gen alpha2 = .667
gen amu2 = mu*alpha2
gen ynb67 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) -
lngamma(y+1) - lngamma(1/alpha2))

* NB ALPHA 1
gen alpha3 = 1
gen amu3 = mu*alpha3
gen ynb10 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) -
lngamma(y+1) - lngamma(1/alpha3))

* NB ALPHA 1.5
gen alpha4 = 1.5
```

```
gen amu4 = mu*alpha4
gen ynb15 = exp(y*ln(amu4/(1+amu4)) - (1/alpha4)*ln(1+amu4) + lngamma(y +1/alpha4) -
lngamma(y+1) - lngamma(1/alpha4))


* NB ALPHA 3
gen alpha5 = 3
gen amu5 = mu*alpha5
gen ynb30 = exp(y*ln(amu5/(1+amu5)) - (1/alpha5)*ln(1+amu5) + lngamma(y +1/alpha5) -
lngamma(y+1) - lngamma(1/alpha5))

lab var ynb0  "NB: alpha=0"
lab var ynb33 "NB: alpha=.33"
lab var ynb67 "NB: alpha=.67"
lab var ynb10 "NB: alpha= 1"
lab var ynb15 "NB: alpha= 1.5"
lab var ynb30 "NB: alpha= 3"

graph twoway connected ynb0 ynb33 ynb67 ynb10 ynb15 ynb30  y, ms(T s o D X)
title("NEGATIVE BINOMIAL DISTRIBUTIONS: MEAN = 0.5")


* ================================================================================


* FIGURE 8.8
* NB2 ALPHA: 0, .33, .67, 1, 1.5, 3; Mean=1
clear
set obs 11
gen mu = 1
gen byte y = _n-1
gen ynb0 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))

* NB ALPHA .33
gen alpha = .333
gen amu = mu*alpha
gen ynb33 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* NB ALPHA .67
gen alpha2 = .667
gen amu2 = mu*alpha2
gen ynb67 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) -
lngamma(y+1) - lngamma(1/alpha2))

* NB ALPHA 1
gen alpha3 = 1
gen amu3 = mu*alpha3
gen ynb10 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) -
lngamma(y+1) - lngamma(1/alpha3))

* NB ALPHA 1.5
gen alpha4 = 1.5
gen amu4 = mu*alpha4
gen ynb15 = exp(y*ln(amu4/(1+amu4)) - (1/alpha4)*ln(1+amu4) + lngamma(y +1/alpha4) -
lngamma(y+1) - lngamma(1/alpha4))

* NB ALPHA 3
gen alpha5 = 3
gen amu5 = mu*alpha5
gen ynb30 = exp(y*ln(amu5/(1+amu5)) - (1/alpha5)*ln(1+amu5) + lngamma(y +1/alpha5) -
lngamma(y+1) - lngamma(1/alpha5))
```

```
lab var ynb0  "NB: alpha=0"
lab var ynb33 "NB: alpha=.33"
lab var ynb67 "NB: alpha=.67"
lab var ynb10 "NB: alpha= 1"
lab var ynb15 "NB: alpha= 1.5"
lab var ynb30 "NB: alpha= 3"

graph twoway connected ynb0 ynb33 ynb67 ynb10 ynb15 ynb30  y, ms(T s o D X)
title("NEGATIVE BINOMIAL DISTRIBUTIONS: MEAN = 1")

*================================================================================

* FIGURE 8.9
* NB2 ALPHA: 0, .33, .67, 1, 1.5, 3; Mean=2
clear
set obs 11
gen mu = 2
gen byte y = _n-1
gen ynb0 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))

* NB ALPHA .33
gen alpha = .333
gen amu = mu*alpha
gen ynb33 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* NB ALPHA .67
gen alpha2 = .667
gen amu2 = mu*alpha2
gen ynb67 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) -
lngamma(y+1) - lngamma(1/alpha2))

* NB ALPHA 1
gen alpha3 = 1
gen amu3 = mu*alpha3
gen ynb10 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) -
lngamma(y+1) - lngamma(1/alpha3))

* NB ALPHA 1.5
gen alpha4 = 1.5
gen amu4 = mu*alpha4
gen ynb15 = exp(y*ln(amu4/(1+amu4)) - (1/alpha4)*ln(1+amu4) + lngamma(y +1/alpha4) -
lngamma(y+1) - lngamma(1/alpha4))

* NB ALPHA 3
gen alpha5 = 3
gen amu5 = mu*alpha5
gen ynb30 = exp(y*ln(amu5/(1+amu5)) - (1/alpha5)*ln(1+amu5) + lngamma(y +1/alpha5) -
lngamma(y+1) - lngamma(1/alpha5))

lab var ynb0  "NB: alpha=0"
lab var ynb33 "NB: alpha=.33"
lab var ynb67 "NB: alpha=.67"
lab var ynb10 "NB: alpha= 1"
lab var ynb15 "NB: alpha= 1.5"
lab var ynb30 "NB: alpha= 3"

graph twoway connected ynb0 ynb33 ynb67 ynb10 ynb15 ynb30  y, ms(T s o D X)
title("NEGATIVE BINOMIAL DISTRIBUTIONS: MEAN = 2")


*================================================================================
```

```
* FIGURE 8.10
* NB2 ALPHA: 0, .33, .67, 1, 1.5, 3; Mean=5
clear
set obs 11
gen mu = 5
gen byte y = _n-1
gen ynb0 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))

* NB ALPHA .33
gen alpha = .333
gen amu = mu*alpha
gen ynb33 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))

* NB ALPHA .67
gen alpha2 = .667
gen amu2 = mu*alpha2
gen ynb67 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) -
lngamma(y+1) - lngamma(1/alpha2))

* NB ALPHA 1
gen alpha3 = 1
gen amu3 = mu*alpha3
gen ynb10 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) -
lngamma(y+1) - lngamma(1/alpha3))

* NB ALPHA 1.5
gen alpha4 = 1.5
gen amu4 = mu*alpha4
gen ynb15 = exp(y*ln(amu4/(1+amu4)) - (1/alpha4)*ln(1+amu4) + lngamma(y +1/alpha4) -
lngamma(y+1) - lngamma(1/alpha4))

* NB ALPHA 3
gen alpha5 = 3
gen amu5 = mu*alpha5
gen ynb30 = exp(y*ln(amu5/(1+amu5)) - (1/alpha5)*ln(1+amu5) + lngamma(y +1/alpha5) -
lngamma(y+1) - lngamma(1/alpha5))

lab var ynb0  "NB: alpha=0"
lab var ynb33 "NB: alpha=.33"
lab var ynb67 "NB: alpha=.67"
lab var ynb10 "NB: alpha= 1"
lab var ynb15 "NB: alpha= 1.5"
lab var ynb30 "NB: alpha= 3"

graph twoway connected ynb0 ynb33 ynb67 ynb10 ynb15 ynb30  y, ms(T s o D X)
title("NEGATIVE BINOMIAL DISTRIBUTIONS: MEAN = 5")

* ================================================================================

* FIGURE 8.11
* NB2 ALPHA: 0, .33, .67, 1, 1.5, 3; Mean=10
clear
set obs 11
gen mu = 10
gen byte y = _n-1
gen ynb0 =  (exp(-mu)*mu^y)/exp(lngamma(y+1))

* NB ALPHA .33
gen alpha = .333
gen amu = mu*alpha
gen ynb33 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) -
lngamma(y+1) - lngamma(1/alpha))
```

```
* NB ALPHA .67
gen alpha2 = .667
gen amu2 = mu*alpha2
gen ynb67 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) -
lngamma(y+1) - lngamma(1/alpha2))

* NB ALPHA 1
gen alpha3 = 1
gen amu3 = mu*alpha3
gen ynb10 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) -
lngamma(y+1) - lngamma(1/alpha3))

* NB ALPHA 1.5
gen alpha4 = 1.5
gen amu4 = mu*alpha4
gen ynb15 = exp(y*ln(amu4/(1+amu4)) - (1/alpha4)*ln(1+amu4) + lngamma(y +1/alpha4) -
lngamma(y+1) - lngamma(1/alpha4))

* NB ALPHA 3
gen alpha5 = 3
gen amu5 = mu*alpha5
gen ynb30 = exp(y*ln(amu5/(1+amu5)) - (1/alpha5)*ln(1+amu5) + lngamma(y +1/alpha5) -
lngamma(y+1) - lngamma(1/alpha5))

lab var ynb0  "NB: alpha=0"
lab var ynb33 "NB: alpha=.33"
lab var ynb67 "NB: alpha=.67"
lab var ynb10 "NB: alpha= 1"
lab var ynb15 "NB: alpha= 1.5"
lab var ynb30 "NB: alpha= 3"

graph twoway connected ynb0 ynb33 ynb67 ynb10 ynb15 ynb30  y, ms(T s o D X)
title("NEGATIVE BINOMIAL DISTRIBUTIONS: MEAN = 10")

*=====================================================================

* FIGURE 8.12
* MEAN=10  a= 0, .1, .3, .5
* POISSON
set obs 30
gen byte mu = 10
gen byte y = _n-1
gen yp =  (exp(-mu)*mu^y)/exp(lngamma(y+1))


* NB2 alpha = .5
gen alpha = .1
gen amu = mu*alpha
gen ynb_5 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) /*
  */  - lngamma(y+1) - lngamma(1/alpha))

* NB2 alpha = 1
gen alpha2 = .3
gen amu2 = mu*alpha2
gen ynb1 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) /*
  */  - lngamma(y+1) - lngamma(1/alpha2))

* NB2 alpha=2
gen alpha3 = .5
gen amu3 = mu*alpha3
gen ynb2 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) /*
   */  - lngamma(y+1) - lngamma(1/alpha3))
```

```
lab var yp "Poisson"
lab var ynb_5 "NB-2;alpha=.1"
lab var ynb1  "NB-2;alpha=.3"
lab var ynb2  "NB-2;alpha=.5"
graph twoway connected ynb2 ynb1 ynb_5 yp  y, ms(s x d T)

*=====================================================================

* FIGURE 8.13
* MEAN = 10   alpha=.6, .8. 1. 1.2
* POISSON
set obs 30
gen byte mu = 10
gen byte y = _n-1

* NB2 alpha = .6
gen alpha = .6
gen amu = mu*alpha
gen ynb_6 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) /*
  */  - lngamma(y+1) - lngamma(1/alpha))

* NB2 alpha = .8
gen alpha1 = .8
gen amu1 = mu*alpha1
gen ynb_8 = exp(y*ln(amu1/(1+amu1)) - (1/alpha1)*ln(1+amu1) + lngamma(y +1/alpha1) /*
  */  - lngamma(y+1) - lngamma(1/alpha1))

* NB2 alpha = 1
gen alpha2 = 1
gen amu2 = mu*alpha2
gen ynb_10 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) /*
  */  - lngamma(y+1) - lngamma(1/alpha2))

* NB2 alpha=1.2
gen alpha3 = 1.2
gen amu3 = mu*alpha3
gen ynb_12 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) /*
   */  - lngamma(y+1) - lngamma(1/alpha3))
lab var ynb_6 "NB-2;alpha=.6"
lab var ynb_8 "NB-2;alpha=.8"
lab var ynb_10 "NB-2;alpha=1"
lab var ynb_12 "NB-2;alpha=1.2"
graph twoway connected ynb_12 ynb_10 ynb_8 ynb_6 y, ms(s x d T)

*=====================================================================================

* CHAPTER 9

* FIGURE 9.1
clear
qui {
use c:\\source\affairs
tab yrsmarr, gen(yrsmarr)
qui poisson naffairs kids avgmarr hapavg vryhap notrel slghtrel smerel vryrel
yrsmarr3-yrsmarr6
predict mu
local i 0
local newvar "pr`i'"
while `i' <=15 {
 local newvar "pr`i'"
 qui gen `newvar' =  exp(-mu)*(mu^`i')/exp(lnfactorial(`i'))
 local i = `i' + 1
}
```

```
quietly gen cnt = .
quietly gen obpr = .
quietly gen prpr = .
local i 0
while `i' <=15 {
 local obs = `i' + 1
 replace cnt = `i' in `obs'
 tempvar obser
 gen `obser' = (`e(depvar)'==`i')
 sum `obser'
 replace obpr = r(mean) in `obs'
 sum pr`i'
 replace prpr = r(mean) in `obs'
 local i = `i' + 1
}
gen byte count = cnt
label var prpr "Poisson - Predicted"
label var obpr "Poisson - Observed"
label var count "Count"
}
twoway scatter prpr obpr count, c(l l) ms(T d) title("nffairs Observed vs Predicted
Probabilities") ytitle(Probability of Affairs)

* ================================================================================
* FIGURE 9.2
clear
qui {
use c:\\source\affairs
tab yrsmarr, gen(yrsmarr)
qui nbreg naffairs kids avgmarr hapavg vryhap notrel slghtrel smerel vryrel yrsmarr3-
yrsmarr6
predict mu
local alpha = e(alpha)
gen amu = mu* e(alpha)
local i 0
local newvar "pr`i'"
while `i' <=15 {
local newvar "pr`i'"
qui gen `newvar' = exp(`i'*ln(amu/(1+amu)) - (1/`alpha')*ln(1+ amu) + lngamma(`i'
+1/`alpha') /*
  */  - lngamma(`i'+1) - lngamma(1/`alpha'))
local i = `i' + 1
}
quietly gen cnt = .
quietly gen obpr = .
quietly gen prpr = .
local i 0
while `i' <=15 {
 local obs = `i' + 1
 replace cnt = `i' in `obs'
 tempvar obser
 gen `obser' = (`e(depvar)'==`i')
 sum `obser'
 replace obpr = r(mean) in `obs'
 sum pr`i'
 replace prpr = r(mean) in `obs'
 local i = `i' + 1
}
gen byte count = cnt
label var prpr "NB2 - Predicted"
label var obpr "NB2 - Observed"
label var count "Count"
}
```

```
twoway scatter prpr obpr count, c(l l) ms(T d) title("nffairs Observed vs Predicted
Probabilities") sub("Negative Binomial") ytitle(Probability of Affairs)

*================================================================================

* FIGURE 9.3
clear
use c:\\source\affairs
tab yrsmarr, gen(yrsmarr)
glm naffairs kids avgmarr hapavg vryhap smerel vryrel yrsmarr4-yrsmarr6, eform fam(nb
ml)
predict  sdev, deviance stan
predict mu
scatter sdev mu, title(Affairs: Negative Binomial)

*================================================================================

* FIGURE 9.4
clear
use c:\\source\azpro
qui {
tab los if procedure==0
tab los if procedure==1
label define procedure 0 "PTCA" 1 "CABG"
label values procedure procedure
drop if los>30
}
hist los, by(procedure) xlab(0 10 20 30)

*================================================================================

* FIGURE 9.5
clear
use c:\\source\mdvis
qui {
qui poisson numvisit reform badh age educ loginc
predict mu
local i 0
local newvar "pr`i'"
while `i' <=20 {
 local newvar "pr`i'"
 qui gen `newvar' =  exp(-mu)*(mu^`i')/exp(lnfactorial(`i'))
 local i = `i' + 1
}
quietly gen cnt = .
quietly gen obpr = .
quietly gen prpr = .
local i 0
while `i' <=20 {
 local obs = `i' + 1
 replace cnt = `i' in `obs'
 tempvar obser
 gen `obser' = (`e(depvar)'==`i')
 sum `obser'
 replace obpr = r(mean) in `obs'
 sum pr`i'
 replace prpr = r(mean) in `obs'
 local i = `i' + 1
}
* Note: starred lines produce Table
* gen obsprop = obpr*100
* gen preprop = prpr*100
gen byte count = cnt
```

```
* gen diffprop = obsprop - preprop
* format obsprop preprop diffprop %8.3f
* list count obsprop preprop diffprop
label var prpr "Predicted days"
label var obpr "Observed days"
label var count "# of Visits"
}
twoway scatter prpr obpr count, c(l l) ms(T d) title(Observed vs Predicted
Probabilities) ytitle(Probability of LOS)

*================================================================================

* FIGURE 9.6A
clear
use c:\\source\mdvis
qui glm numvisit reform badh educ2 educ3 age2 age3 loginc, fam(poi)
predict  sdevp, deviance stan
predict mup
replace mup = 2*sqrt(mup)
lab var mup "Corrected mu"
scatter sdevp mup, title(Poisson: Standardized Deviance vs 2*sqrt(mu)) sub(German
Health Reform 1996 1998)

*================================================================================

* FIGURE 9.6B
clear
use c:\\source\mdvis
qui glm numvisit reform badh educ2 educ3 age2 age3 loginc, fam(nb ml)
predict  sdevnb, deviance stan
predict munb
replace munb = 2*sqrt(munb)
lab var munb "Corrected mu"
scatter sdevnb munb, title(NB2: Standardized Deviance vs 2*sqrt(mu)) sub(German Health
Reform 1996 1998)

*================================================================================

* FIGURE 9.7
clear
use c:\\source\mdvis

qui nbreg numvisit reform badh educ age loginc, nolog
 local maxcount = 20 // compute predictions up to count 20

//  compute mean of predictions (not predictions at mean)

*   cases with reform == 1
*   the plot option will compute variables with mean predictions
    prcounts postref_ if reform==1, plot max(`maxcount')
        rename postref_preq mean_post98
        label var mean_post98 "Mean of 1998 post reform"
        rename postref_obeq observed_prob
        label var observed_prob "Observed probability"
        rename postref_val visits
        label var visits "Number vists"

*   cases with reform == 0
    prcounts preref_ if reform==0, plot max(`maxcount')
        rename preref_preq mean_pre96
        label var mean_pre96 "Mean of 1996 pre reform"

//  compute predictions computed at the mean level of other variables
```

```
*   for those with reform==0
    prvalue, x(reform=0) max(`maxcount')
        matrix preref_atmean = r(probs) // grab computed pr(y)
        svmat preref_atmean // put them in a variable
        rename preref_atmean1 preref_atmean
        label var preref_atmean "Pre reform 1996 at mean" // give it a label

*   for those with procedure==1
    prvalue, x(reform=1) max(`maxcount')
        matrix postref_atmean = r(probs)
        svmat postref_atmean
        rename postref_atmean1 postref_atmean
        label var postref_atmean "Post reform 1998 at mean"

//  variables we can plot

    desc postref_atmean mean_post98 preref_atmean mean_pre96 observed_prob

//  plot predictions

    twoway ///
    (connected postref_atmean visits, msymbol(circle)) ///
    (connected mean_post98 visits, msymbol(circle_hollow)) ///
    (connected preref_atmean visits, msymbol(square)) ///
    (connected mean_pre96 visits, msymbol(square_hollow)) ///
    (connected observed_prob visits, msymbol(plus)) ///
    , ytitle("Pr(y=k)") ///
    ylabel(0(.05).2, grid gmax gmin gstyle(dot)) ///
    xlabel(0(5)20, nogrid)


*=====================================================================
* CHAPTER 10

* FIGURE 10.1
clear
set obs 11
gen y = _n-1
gen muh = .5
gen byte mu1 = 1
gen byte mu2 = 2
gen byte mu5 = 5
gen byte mu10 = 10
gen ynbh =exp(y*ln(muh/(1+muh))  -ln(1+muh) +lngamma(y+1)-lngamma(y+1))
gen ynb1 =exp(y*ln(mu1/(1+mu1))  -ln(1+mu1) +lngamma(y+1)-lngamma(y+1))
gen ynb2 =exp(y*ln(mu2/(1+mu2))  -ln(1+mu2) +lngamma(y+1)-lngamma(y+1))
gen ynb5 =exp(y*ln(mu5/(1+mu5))  -ln(1+mu5) +lngamma(y+1)-lngamma(y+1))
gen ynb10=exp(y*ln(mu10/(1+mu10))-ln(1+mu10)+lngamma(y+1)-lngamma(y+1))
lab var y     "count"
lab var ynbh  "mean=.5"
lab var ynb1  "mean= 1"
lab var ynb2  "mean= 2"
lab var ynb5  "mean= 5"
lab var ynb10 "mean=10"
graph twoway connected ynb10 ynb5 ynb2 ynb1 ynbh y, ms(s x d T) ///
  title("Negative Binomial Distributions: alpha=1")

*=====================================================================

* FIGURE 10.2
clear
use c:\\source\mdvis
```

```
glm numvisit reform badh educ2 educ3 age2 age3 loginc, fam(nb 1) link(nb)
predict sdevg, deviance stan
predict mug
scatter sdevg mug, title(Canommnical Geometric Std Deviance vs Fit)

* ================================================================================
* CHAPTER 11

* FIGURE 11.1
clear
use c:\\source\mdvis
histogram numvisit, bin(20) percent

* ================================================================================

* FIGURE 11.2
* Modified from code in Long and Freese (2006), Page 406
use c:\\source\mdvis, clear
* NB2
qui nbreg numvisit reform badh educ3 age3 if numvis>0, nolog
prcounts pnb2, plot max(9)
lab var pnb2preq "Predicted NB2"
lab var pnb2obeq "Observed NB2"

* ZIP
qui zip numvisit reform badh educ3 age3, inflate(reform badh educ3 age3)
prcounts pzip, plot max(9)
lab var pzippreq "Predicted ZIP

* ZINB
qui zinb numvisit reform badh educ3 age3, inflate(reform badh educ3 age3)
prcounts pzinb, plot max(9)
lab var pzinbpreq "Predicted ZINB

*  Differences
gen obs = pnb2obeq
gen dnb2 = obs - pnb2preq
lab var dnb2 "NB2 Diff"

gen dzip = obs - pzippreq
lab var dzip "ZIP Diff"

gen dzinb = obs - pzinbpreq
lab var dzinb "ZINB Diff"

graph twoway connected dnb2 dzip dzinb pnb2val, ///
  ytitle(Observed-Predicted) ylab(-.10(.05).10) ///
  xlab(0(1)9) msymbol(0h Sh 0 S)

*================================================================
* CHAPTER 13

* Fig 13.1
use c:\\source\mdvis, clear
gen numvjit = numvisit + runiform()
qui qplot scatter numvisit if numvisit <=25, recast(line) scale(1.25) lwidth(medthick)
saving(f13num, replace)
qui qplot scatter numvjit if numvisit <=25, recast(line) scale(1.25) lwidth(medthick)
saving(f13jit, replace)
graph combine f13num.gph f13jit.gph, title(numvisit: Continuous vs Jittered)


*================================================================
```

```
* CHAPTER 15

* SAS CODE TO GENERATE BAYESIAN MODEL OUTPUT AND FIGURES 15.1-15.3
libname data 'sasdata';
ods graphics on / imagefmt=wmf;
proc genmod data=data.badhealth;
   class BADH;
   model numvisit=age badh/d=nb;
   bayes sampling=gam
         diagnostics=none;
run;
data NormalPrior;
input _type_ $ Intercept Age Badh0 Badh1;
datalines;
Var 1e6  1e6   .01631 1e6
Mean 0.0 0.0 -1.3540  0.0
;
proc genmod data=data.badhealth;
   class BADH;
   model numvisit=age badh/d=nb;
   bayes sampling=gam
         nmc=100000
         thin=10
         diagnostics=none
         cprior=normal(input=NormalPrior);
run;
ods graphics off;
```

B: R code for Figures

```
# FIGURE 3.1
# Table 3.1
obs <- 11
mu <- 2
y <- 0:10
yp2 <-  (exp(-mu)*mu^y)/exp(log(gamma(y+1)))
alpha <- 1.5
amu <- mu*alpha
ynb2 = exp(
        y*log(amu/(1+amu))
     - (1/alpha)*log(1+amu)
     + log( gamma(y +1/alpha) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha) )
)
plot(  y, ynb2, col="red", pch=5,
  main="Poisson vs Negative Binomial PDFs")
  lines( y, ynb2, col="red")
  points(y, yp2,  col="blue", pch=2)
  lines( y,  yp2, col="blue")
  legend(4.3,.40,
    c("Negative Binomial: mean=2, a=1.5",
      "Poisson: mean=2"),
    col=( c("red","blue") ),
    pch=( c(5,2) ),
    lty=1)
# FOR NICER GRAPHIC
zt <- 0:10           #zt is Zero to Ten
x <- c(zt,zt)        #two zt's stacked for use with ggplot2
newY <- c(yp2, ynb2) #Now stacking these two vars
Distribution <- gl(n=2,k=11,length=22,
  label=c("Poisson","Negative Binomial")
)
NBPlines <- data.frame(x,newY,Distribution)
library("ggplot2")
ggplot( NBPlines, aes(x,newY,shape=Distribution,col=Distribution ) ) +
  geom_line() + geom_point()

# ============================================================================
# CHAPTER

# FIGURE 6.1
# Table 6.4 plus added code
nobs <- 50000
x1 <- qnorm(runif(nobs))
x2 <- qnorm(runif(nobs))
py <-rpois(nobs, exp(2 + .75*x1 -1.25*x2))
mpy <- mean(py)
ypoi <- (exp(-mpy)*mpy^py)/gamma(py+1)
 plot(ypoi~ py, xlim=c(0,50), main="Synthetic Poisson Model: Mean=21")

# ============================================================================

# FIGURE 6.2  Table 6.15
load("c://source/medpar.RData")
mdpar <- glm(los ~ hmo+white+type2+type3,family=poisson, data=medpar)
mu <- fitted.values(mdpar)
p <- NULL
avgp <- NULL
```

```
for (i in 0:25) {
 p[[i+1]] <- exp(-mu)*(mu^i)/factorial(i)
 avgp[i+1] <- mean(p[[i+1]])
}
nCases <- dim(medpar)
n<- NULL
propObs<- NULL
probFit<- NULL
yFitMean<- NULL
for (i in 0:25) {                  #possible values for LOS
bLos<- medpar$los==i               #selector for los=i
n[i+1]<- sum(bLos) #number for los=i
propObs[i+1]<- n[i+1]/nCases[1]    #observed proportion for LOS=i
}
Diff <- propObs*100 - avgp*100
data.frame(LOS=0:25, ObsProp=propObs*100, avgp*100, Diff)
# ---------------------------------------------------------------
plot(0:25, avgp, type="b", xlim=c(0,25),
      main = "Observed vs Predicted Days",
      xlab = "Days in Hospital", ylab = "Probability of LOS")
lines(0:25, propObs, type = "b", pch = 2)
legend("topright", legend = c("Predicted Days","Observed Days"),
        lty = c(1,1), pch = c(1,2))
# ===============================================================

FIGURE 6.2  Table 6.15 - ANOTHER METHOD

rm(list=ls())
load("c://source/medpar.RData")
mdpar <- glm(los ~ hmo+white+type2+type3, family=poisson, data=medpar)
mu <- fitted(mdpar)
avgp <- sapply(0:25, function(i) mean(exp(-mu)*(mu^i)/factorial(i)))
propObsv <- with(subset(medpar, los < 25), table(los) / nrow(medpar))
Diff <- c(0,propObsv)*100 - avgp[1:25]*100
data.frame(LOS=0:24, ObsProp=c(0,propObsv)*100, avgp[1:25]*100, Diff)
# ===================================================================

# FIGURE 6.3
# Table 6.16
m<- c(0.5,1,3,5,7,9) #Poisson means
y<- 0:19             #Observed counts
layout(1)
for (i in 1:length(m)) {
  p<- dpois(y, m[i]) #poisson pmf
  if (i==1) {
  plot(y, p, col=i, type='l', lty=i)
  } else {
  lines(y, p, col=i, lty=i)
  }
}

# ============================================================================

# FIGURE 6.4
# Table 6.17
load("c://source/rwm5yr.RData")
eppoi <- glm(docvis ~ outwork+age+female+married+edlevel2+edlevel3+ edlevel4,
family=poisson, data=rwm5yr)
rest <-  eppoi$coef[4]*mean(rwm5yr$female) + eppoi$coef[5]*mean(rwm5yr$married) +
         eppoi$coef[6]*mean(rwm5yr$edlevel2) + eppoi$coef[7]*mean(rwm5yr$edlevel3) +
         eppoi$coef[8]*mean(rwm5yr$edlevel4)
out0 <- eppoi$coef[1] + eppoi$coef[3]*rwm5yr$age + rest
out1 <- eppoi$coef[1] + eppoi$coef[2]*1 + eppoi$coef[3]*rwm5yr$age + rest
```

```
eout1 <- exp(out1)
eout0 <- exp(out0)
matplot(cbind(rwm5yr$age, rwm5yr$age), cbind(eout0, eout1),
  pch=1:2, col=1:2, xlab='Count', ylab='Frequency?')
matplot(cbind(rwm5yr$age, rwm5yr$age), cbind(eout0, eout1), type='l',
  lty=1:2, col=1:2, xlab='Doctor visits', ylab='Frequency')




# Figure 6.4  Another method of graphing

# ====================================================================
load("c://source/rwm5yr.RData")
eppoi <- glm(docvis ~ outwork + age + female + married + edlevel2 +
    edlevel3+ edlevel4,  family=poisson, data=rwm5yr)
attach(rwm5yr)
rest <- coef(eppoi)[4]*mean(female) + coef(eppoi)[5]*mean(married) +
     coef(eppoi)[6]*mean(edlevel2) + coef(eppoi)[7]*mean(edlevel3) +
     coef(eppoi)[8]*mean(edlevel4)
L1 <- coef(eppoi)[1] + coef(eppoi)[2]*1 + coef(eppoi)[3]*age + rest
L0 <- coef(eppoi)[1] + coef(eppoi)[2]*0 + coef(eppoi)[3]*age + rest
eout1 <- exp(L1)
eout0 <- exp(L0)
layout(1)
plot(age, eout1, col=1, type='p')
lines(age, eout0, col=2, type='p')
#  ====================================================================




# Figure 6.4b: Probability of doctor visits: work vs not working
#  ====================================================================

use rwm5yr, clear
qui poisson docvis outwork age female married edlevel2-edlevel4
gen L0 = _b[_cons] + _b[age]*43.78566  + _b[female]*.4804937 +
     _b[married]*.7736244 + _b[edlevel2]*.0587995 +
     _b[edlevel3]*.0883778 + _b[edlevel4]*.0657861
gen L1 = _b[outwork]*1 + L0
gen pr1 = exp(L1)
gen pr0 = exp(L0)
egen pro1 = mean(pr1), by(docvis)
egen pro0 = mean(pr0), by(docvis)
gen prob1 = exp(-pro1)*pro1^docvis/exp(lnfactorial(docvis))
gen prob0 = exp(-pro0)*pro0^docvis/exp(lnfactorial(docvis))
lab var prob1 "Not Working"
lab var prob0 "Working"
graph twoway connected prob1 prob0 docvis if docvis<16, sort ms(t l)
     xlabel(1(2)16) title(Probability of doctor visits: work vs not working)
#  ====================================================================
```

```
# ============================================================================
# CHAPTER 8

# FIGURE 8.1
# NB2: MEANS=0.5,1,2,5,10;  ALPHA=0
m<- c(0.5,1,2,5,10)  #mean values
y<- 0:10             #Observed counts
layout(1)
for (i in 1:length(m)) {
  p<- dpois(y, m[i]) #poisson pmf
  if (i==1) {
  plot(y, p, col=i, type='l', lty=i)
  } else {
  lines(y, p, col=i, lty=i)
  }
}

# ============================================================================

# FIGURE 8.2
# NB2: MEANS=0.5,1,2,5,10;  ALPHA=0.33
obs <- 11
mu <- c(0.5,1,2,5,10)
y <- 0:10
alpha <- .33
amu <- mu*alpha
layout(1)
for (i in 1:length(mu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha)*log(1+amu[i])
     + log( gamma(y +1/alpha) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ============================================================================

# FIGURE 8.3
# NB2: MEANS=0.5,1,2,5,10;  ALPHA=0.67
obs <- 11
mu <- c(0.5,1,2,5,10)
y <- 0:10
alpha <- .67
amu <- mu*alpha
layout(1)
for (i in 1:length(mu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha)*log(1+amu[i])
     + log( gamma(y +1/alpha) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
```

```
  lines(y, ynb2, col=i, lty=i)
  }
}
# ======================================================================

# FIGURE 8.4
# NB2: MEANS=0.5,1,2,5,10;  ALPHA=1
obs <- 11
mu <- c(0.5,1,2,5,10)
y <- 0:10
alpha <- 1
amu <- mu*alpha
layout(1)
for (i in 1:length(mu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha)*log(1+amu[i])
     + log( gamma(y +1/alpha) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ======================================================================

# FIGURE 8.5
# NB2: MEANS=0.5,1,2,5,10;  ALPHA=1.5
obs <- 11
mu <- c(0.5,1,2,5,10)
y <- 0:10
alpha <- 1.5
amu <- mu*alpha
layout(1)
for (i in 1:length(mu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha)*log(1+amu[i])
     + log( gamma(y +1/alpha) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ======================================================================

# FIGURE 8.6
# NB2: MEANS=0.5,1,2,5,10;  ALPHA=3
obs <- 11
mu <- c(0.5,1,2,5,10)
y <- 0:10
alpha <- 3
amu <- mu*alpha
layout(1)
```

```
for (i in 1:length(mu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha)*log(1+amu[i])
     + log( gamma(y +1/alpha) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ========================================================================

# FIGURE 8.7
# NB2: ALPHA=0, 0.33, 0.67, 1, 1.5, 3;  MEAN=0.5
obs <- 11
alpha <- c(.009, .33, .67, 1, 1.5, 3)
y <- 0:10
mu <- .5
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ========================================================================

# FIGURE 8.8
# NB2: ALPHA=0, 0.33, 0.67, 1, 1.5, 3;  MEAN=1
obs <- 11
alpha <- c(.009, .33, .67, 1, 1.5, 3)
y <- 0:10
mu <- 1
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
```

```
}

# ================================================================

# FIGURE 8.9
# NB2: ALPHA=0, 0.33, 0.67, 1, 1.5, 3;  MEAN=2
obs <- 11
alpha <- c(.009, .33, .67, 1, 1.5, 3)
y <- 0:10
mu <- 2
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ================================================================

# FIGURE 8.10
# NB2: ALPHA=0, 0.33, 0.67, 1, 1.5, 3;  MEAN=5
obs <- 11
alpha <- c(.009, .33, .67, 1, 1.5, 3)
y <- 0:10
mu <- 5
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ================================================================

# FIGURE 8.11
# NB2: ALPHA=0, 0.33, 0.67, 1, 1.5, 3;  MEAN=10
obs <- 11
alpha <- c(.009, .33, .67, 1, 1.5, 3)
y <- 0:10
mu <- 10
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
```

```
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ================================================================

# FIGURE 8.12
# NB2: ALPHA=0, 0.33, 0.67, 1, 1.5, 3;  MEAN=10
obs <- 11
alpha <- c(.009, .1, .3, .5)
y <- 0:30
mu <- 10
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}

# ================================================================

# FIGURE 8.13
# NB2: ALPHA=.6, .8, 1, 1.2;  MEAN=10
obs <- 11
alpha <- c(.6, .8, 1, 1.2)
y <- 0:30
mu <- 10
amu <- mu*alpha
layout(1)
for (i in 1:length(amu)) {
ynb2 = exp(
        y*log(amu[i]/(1+amu[i]))
     - (1/alpha[i])*log(1+amu[i])
     + log( gamma(y +1/alpha[i]) )
     - log( gamma(y+1) )
     - log( gamma(1/alpha[i]) )
)
if (i==1) {
plot(y, ynb2, col=i, type='l', lty=i)
  } else {
  lines(y, ynb2, col=i, lty=i)
  }
}
```

```
#=================================================================
# CHAPTER 9

# FIGURE 9.1

# =================================================================

# FIGURE 9.2

# =================================================================

# FIGURE 9.3
load("c://source/affairs.RData")
affnb2r <-  glm.nb(naffairs~ avgmarr + hapavg + vryhap + smerel
     + vryrel + yrsmarr4 + yrsmarr5 + yrsmarr6, data=affairs)
summary(affnb2r)
confint(affnbr2)
exp(coef(affnb2r))
exp(confint(affnb2r))
deviance <- residuals(affnb2, type="deviance")
dev <- sum(deviance*deviance)
pred <- predict(affnb2, se.fit=TRUE, type="response")
mu <- pred$fit
stdp <- pred$se.fit                 # Std error of prediction
variance <- mu
h <- stdp * stdp*variance           # hat matrix diagonal
sdeviance <- rstandard(affnb2)      # Std deviance
plot(mu, sdeviance)

#=================================================================

# FIGURE 14.1
library(gamlss.mx)
load("c://source/medpar.RData")
rinb <- gamlssNP(los~ hmo +white+ type2 +type3, random=~1|provnum,
      data=medpar, family=NBI, mixture="gq", K=20)
summary(rinb)

summary(rinb$sigma.fv)
m<-rinb$mu.fv       # fitted values for extended model
s<-rinb$sigma.fv    # sigma
presid <- (medpar$los-m)/sqrt(m+s*m*m)
summary(presid)
hist(presid)
```

C: DISCRETE RESPONSE GLM-IRLS MODELS

I am providing simple R functions for Poisson, negative binomial, and logistic regression with offset capabilities. These have since been enhanced and made part of an *irls.r* function that provides estimates, SE, z, p, and confidence intervals for all but gamma and inverse Gaussian GLM families. Included are Gaussian, binary logit, probit, and cloglog, binomial or grouped logit, probit, and cloglog, Poisson, and negative binomial. Poisson and negative binomial families may have offsets. A host of ancillary statistics are produced in estimation and are made available for the user, including a *summary(model)* function that provides output identical to *glm()*. Postestimation statistics that can be used include:

[Note: *model* is an example name of the function declared by the user]

```
model$coefficients     # vector of coefficients
mode$se.beta.hat       # standard errors
model$df.null          # null degrees of freedom
model$nobs             # number of observations in model
model$df.residual      # residual degrees of freedom
model$deviance         # model deviance
model$null.deviance    # model null deviance
model$pearson.chi2     # Pearson Chi2 statistic
model$loglik           # model log-likelihood
model$residuals        # list of raw residuals
model$aic              # model AIC statistic
model$X                # matrix of data elements in model
model$i                # list of observations
model$call             # repeat of model definition
model$family           # family
residuals(model, type="standard")   # standardized deviance residuals
hatvalues(model)       # hat statistics
```

residuals and hat values can be made into individual values by:

```
stdres <- format(residuals(irls.logit, type="standard"), digits=6)
hat <- format(hatvalues(irls.logit), digits=6)
```

This *irls.r* function, which we may enhance to include gamma aned inverse Gaussian, is part of a book called:

> Hilbe, Joseph M. and Andrew Robinson, *Methods of Statistical Model Estimation*, Chapman & Hall/CRC

that we are currently working on. *irls.r* is from the chapter on IRLS methods of estimation, in which we describe in full how to construct modular functions in R like *irls* and *glm*. I believe, however, that the function we have constructed is superior to that of the *glm()*. The book is due to be completed before the end of 2011. The three functions below are simple in their design, producing only estimates. But they do give an idea of how IRLS models are structured. Chapter 4 of the text provides a background and details on maximum likelihood estimation, including IRLS estimation.

I will post some updates to the 3 functions below in the near future, so check back if interested. The full *irls.r* function will be available with the MSME book, but perhaps we can make it available before if there is a good reason.

POISSON REGRESSION

```
# Poisson regression GLM-IRLS  with log-likelihood covergence
irls_pois <- function(formula, data, tol=.0000001, offset=0) {
  mf <- model.frame(formula, data)
  y <- model.response(mf, "numeric")
  X <- model.matrix(formula, data = data)
  if (any(is.na(cbind(y, X)))) stop("Some data are missing.")
  mu <- (y + mean(y))/2
  eta <- log(mu)
  loglike <- sum(y*log(mu) - mu - log(factorial(y)))
  deltall <- 1
  i <- 1
  while (abs(deltall) > tol ) {
    w <-  mu
    z <- eta + (y - mu)/w - offset
    mod <- lm(z ~ X-1, weights=w)
    eta <- mod$fit
    eta <- eta + offset
    mu <- exp(eta)
    ll.old <- loglike
    loglike <- sum(y*log(mu) - mu - log(factorial(y)))
    deltall <- loglike - ll.old
    cat(i, coef(mod), deltall, "\n")
    i <- i + 1
  }
return(coef(mod))
}

library(COUNT)
data(medpar)
attach(medpar)
off <- rep(1:5, each=299, times=1)*100          # offset
loff <- log(off)                                # log offset
mymodel <- irls_pois(los ~ hmo + white , data=medpar, offset=loff)
mymodel
```

DISPLAY OF OUTPUT

```
> mymodel <- irls_pois(los ~ hmo + white , data=medpar, offset=loff)
1 -2.949004 0.02794588 -0.2984571 -5113.528
1 -3.132680 0.004244444 -0.2912925 311.0955
1 -3.151783 -0.001162077 -0.2897268 2.601171
1 -3.151968 -0.001274301 -0.2896969 0.0002354469
1 -3.151968 -0.001274325 -0.2896969 0

> mymodel
X(Intercept)         Xhmo       Xwhite
-3.151967982 -0.001274325 -0.289696863
```

OUTPUT WTHOUT OFFSET

```
> mymodel <- irls_pois(los ~ hmo + white , data=medpar)
   It\geration log not shown
> mymodel
X(Intercept)         Xhmo       Xwhite
   2.4822518   -0.1415782   -0.1908900
```

NEGATIVE BINOMIAL REGRESSION – LOG LINK

```
# IRLS-GLM negative binomial log-link
irls_nb <- function(formula, data, tol=.0000001, offset=0, a=1) {
  mf <- model.frame(formula, data)
  y <- model.response(mf, "numeric")
  X <- model.matrix(formula, data = data)
  if (any(is.na(cbind(y, X)))) stop("Some data are missing.")
  mu <- (y + mean(y))/2
  eta <- log(mu)
  dev <- 2* sum(y*log(y/mu) - (1/a)*log((1+a*y)/(1+a*mu)))
  deltad <- 1
  i <- 1
  while (abs(deltad) > tol ) {
    w <-  mu + a*mu*mu
    z <- eta + (y - mu)/w - offset
    mod <- lm(z ~ X-1, weights=w)
    eta <- mod$fit
    eta <- eta + offset
    mu <- exp(eta)
    dev.old <- dev
    dev <- 2* sum(y*log(y/mu) - (1/a)*log((1+a*y)/(1+a*mu)))
    deltad <- dev - dev.old
    cat(i, coef(mod), deltad, "\n")
    i <- i + 1
  }
return(coef(mod))
}
library(COUNT)
data(medpar)
attach(medpar)
off <- rep(1:5, each=299, times=1)*100
loff <- log(off)
# MODEL- ALPHA SPED\CIFIED AS 0.5
mymodel <- irls_nb(los ~ hmo + white , data=medpar, offset=loff, a=.5)
mymodel
```

DISPLAY OF OUTPUT: WITH ALPHA=.5, WITH OFFSET

```
> mymodel
X(Intercept)         Xhmo       Xwhite
-3.151967980 -0.001274325 -0.289696865
```

DISPLAY OF OUTPUT: WITH ALPHA= 1.5, NO OFFSET

```
mymodel <- irls_nb(los ~ hmo + white , data=medpaR, a=1.5)
mymodel
```

```
> mymodel
X(Intercept)         Xhmo       Xwhite
   2.4822518   -0.1415782   -0.1908900
```

LOGISTIC REGRESSION – BINARY RESPONSE

```
irls_logit <- function(formula, data, tol = .0000001, offset=0) {
  mf <- model.frame(formula, data)
  y <- model.response(mf, "numeric")
  X <- model.matrix(formula, data = data)
  if (any(is.na(cbind(y, X)))) stop("Some data are missing.")
  mu <- (y + .5)/2
  eta <- log(mu/(1-mu))
  loglike <- sum(y*log(mu/(1-mu)) + log(1-mu))
  deltall <- 1
  i <- 1
  while (abs(deltall) > tol ) {
    w <-  mu*(1-mu)
    z <- eta + (y - mu)/(mu*(1-mu)) - offset
    mod <- lm(z ~ X-1, weights=w)
    eta <- mod$fit
    eta <- eta + offset
    mu <- 1/(1+exp(-eta))
    ll.old <- loglike
    loglike <- sum(y*log(mu/(1-mu)) + log(1-mu))
    deltall <- loglike - ll.old
    cat(i, coef(mod), deltall, "\n")
    i <- i + 1
  }
return(coef(mod))
}

library(COUNT)
data(medpar)
attach(medpar)
mymodel <- irls_logito(died ~ hmo + white , data=medpaR)
mymodel
```

DISPLAY OF OUTPUT

```
> mymodel <- irls_logito(died ~ hmo + white , data=medpar)
1 -1.051936 -0.01343265 0.318181 -532.3138
2 -0.9224268 -0.01216259 0.3017145 2.096652
3 -0.9261831 -0.01224641 0.3033848 0.0008688413
4 -0.9261862 -0.01224648 0.3033872 1.904255e-10

> mymodel
X(Intercept)         Xhmo       Xwhite
 -0.92618620  -0.01224648   0.30338724
```

SYNTHETIC ZERO-INFLATED POISSON

```
# ZERO-INFLATED POISSON  syn.zip.r
# J Hilbe  6Jun2011
library(MASS)
library(pscl)
nobs <- 50000
x1 <- runif(nobs)
x2 <- runif(nobs)
# POISSON
xb <- 2 + .75*x1 - 1.25*x2
exb <- exp(xb)
poy <- rpois(nobs, exb)
pdata <- data.frame(poy, x1, x2)
pi <- 1/(1+exp(-(.9*x1 + .1*x2 + .2)))
pdata$bern <- runif(nobs)>pi
zy <- pdata$bern * poy
zip <- zeroinfl(zy ~ x1 + x2 | x1 + x2, dist="poisson", data=pdata)
summary(zip)
```

OUTPUT

```
Call:
zeroinfl(formula = zy ~ x1 + x2 | x1 + x2, data = nbdata, dist = "poisson")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-0.8020 -0.6383 -0.5680  0.6497  5.7667

Count model coefficients (poisson with log link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.001297   0.007878  254.02   <2e-16 ***
x1           0.765092   0.011039   69.31   <2e-16 ***
x2          -1.276300   0.011658 -109.48   <2e-16 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.21977    0.02497   8.800   <2e-16 ***
x1           0.86788    0.03366  25.787   <2e-16 ***
x2           0.09053    0.03351   2.701   0.0069 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 11
Log-likelihood: -6.808e+04 on 6 Df
```

SYNTHETIC ZERO-INFLATED NEGATIVE BINOMIAL

```
# ZERO-INFLATED NEGATIVE BINOMIAL  syn.zinb.r
# J Hilbe  6Jun2011
library(MASS)
library(pscl)
nobs <- 50000
x1 <- runif(nobs)
x2 <- runif(nobs)
xb <- 2 + .75*x1 - 1.25*x2
a <- .5
ia <- 1/.5
exb <- exp(xb)
xg <- rgamma(nobs, a, a, ia)
xbg <-exb*xg
nby <- rpois(nobs, xbg)
nbdata <- data.frame(nby, x1, x2)
pi <- 1/(1+exp(-(.9*x1 + .1*x2 + .2)))
nbdata$bern <- runif(nobs)>pi
zy <- nbdata$bern * nby
zinb <- zeroinfl(zy ~ x1 + x2 | x1 + x2, dist="negbin", data=nbdata)
summary(zinb)



Call:
zeroinfl(formula = zy ~ x1 + x2 | x1 + x2, data = nbdata, dist = "negbin")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-0.4067 -0.3591 -0.3243 -0.2905 15.1599

Count model coefficients (negbin with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.97341    0.03361   58.72   <2e-16 ***
x1           0.78066    0.04398   17.75   <2e-16 ***
x2          -1.18526    0.04366  -27.15   <2e-16 ***
Log(theta)  -0.65375    0.03245  -20.15   <2e-16 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26235    0.03805   6.894 5.42e-12 ***
x1           0.89756    0.04473  20.066  < 2e-16 ***
x2           0.03071    0.04406   0.697    0.486
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta = 0.5201
Number of iterations in BFGS optimization: 25
Log-likelihood: -6.261e+04 on 7 Df
```

SYNTHETIC POISSON-LOGIT HURDLE

```
# syn.logitpoisson.hurdle.r
# R - SYNTHETIC LOGIT-POISSON HURDLE SCRIPT
# J. Hilbe 6 June 2011
library(MASS)
library(pscl)

# Generate sample size
nobs <- 50000

# Generate predictors, design matrix
x1 <- runif(nobs)
x2 <- runif(nobs)
xb <- 2 + .75*x1 - 1.25*x2

# Construct Poisson responses
exb <- exp(xb)
poy <- rpois(nobs, exb)
pdata <- data.frame(poy, x1, x2)

# Construct filter
pi <- 1/(1+exp(-(.9*x1 + .1*x2 + .2)))
pdata$bern <- runif(nobs) > pi

# Remove all response zeros.
pdata <- subset(pdata, poy > 0)

# Add structural zeros
pdata$poy[pdata$bern] <- 0

# Model Synthetic Logit-Poisson Hurdle data
hlpoi <- hurdle(poy ~ x1 + x2,
                dist = "poisson",
                zero.dist = "binomial",
                link = "logit",
                data = pdata)
summary(hlpoi)
```

SYNTHETIC NEGATIVE BINOMIAL-LOGIT HURDLE

```
# syn.logitnb2.hurdle.r
# Table 11.4:  R - SYNTHETIC LOGIT-NB2 HURDLE SCRIPT
# in Hilbe, J.M. (2011), Negative Binomial Regression, 2nd ed, Cambridge Univ Press
library(MASS)
library(pscl)

# Generate sample size
nobs <- 50000

# Generate predictors, design matrix
x1 <- runif(nobs)
x2 <- runif(nobs)
xb <- 2 + .75*x1 - 1.25*x2
a <- .5

# Construct negative binomial responses
ia <- 1/.5
exb <- exp(xb)
xg <- rgamma(nobs, a, a, ia)
xbg <- exb*xg
nby <- rpois(nobs, xbg)
nbdata <- data.frame(nby, x1, x2)

# Construct filter
pi <- 1/(1+exp(-(.9*x1 + .1*x2 + .2)))
nbdata$bern <- runif(nobs) > pi

# Remove all response zeros.
nbdata <- subset(nbdata, nby > 0)

# Add structural zeros
nbdata$nby[nbdata$bern] <- 0

# Model Synthetic Logit-NB2 Hurdle data
hlnb2 <- hurdle(nby ~ x1 + x2,
                dist = "negbin",
                zero.dist = "binomial",
                link = "logit",
                data = nbdata)
summary(hlnb2)
```

BAYESIAN POISSON AND NEGATIVE BINOMIAL

WinBUGS code

```
POISSON
model
{  for (i in 1:n)  {y[i] - dpois(lambda)}
    lambda ~ dgamma(0.001, 0.001)
}

NEGATIVE BINOMIAL
model
{ for (i in 1:n)
{
y[i] ~ dpois(lambda[i])
lambda[i] ~ gamma(alpha, beta)
}
alpha ~ dgamma(0.001, 0.001)
beta  ~ dgamma(0.001, 0.001)
mu <- alpha/beta
variance <- alpha/beta + alpha/(beta*beta)
}

NEGATIVE BINOMIAL USING RANDOM EFFECTS
model
{ for (i in 1:n)
{
y[i] ~ dpois(lambda[i])
lambda[i] <- max(0.000001, mu+e[i])
e[i] ~ dnorm(0,tau)
}
mu ~ dgamma(0.001, 0.001)
tau ~ dgamma(0.001, 0.001)
sigma <- 1/sqrt(tau)
}
```

Part 3: Stata Commands

Canonical Negative Binomial MLE

```
*! Version 1.2
* CANONICAL NEGATIVE BINOMIAL REGRESSION:  Joseph Hilbe  : 25Sep2005 23Feb2009
program cnbreg, eclass properties(svyb svyj  svyr)
    version 9.1
    syntax [varlist] [if] [in] [fweight pweight aweight  iweight] [, ///
        CENsor(string)  Level(cilevel)                ///
        OFFset(passthru)  EXposure(passthru)            ///
        CLuster(passthru) IRr Robust noLOG FROM(string asis) *  ]
    gettoken  lhs rhs : varlist
    mlopts mlopts, `options'
    if ("`weight'" != "")   local weight "[`weight'   `exp']"
    if (`"`from'"' != `""') local initopt `"init(`from')"'

    ml model lf cnbreg_ll (xb: `lhs' =  `rhs', `offset' `exposure')    ///
                /lnalpha                    ///
        `if' `in' `weight',                          ///
        `mlopts' `robust' `cluster'                ///
        title("Canonical Negative Binomial Regression")            ///
        maximize `log' `initopt'                ///
        diparm(lnalpha, exp label(alpha))

    ereturn scalar k_aux = 1
    ml display, level(`level') `irr'

qui {
* AIC
    tempvar aic
    local nobs e(N)
    local npred e(df_m)
    local df = e(N) - e(df_m) -1
    local llike  e(ll)
    gen `aic' = ((-2*`llike') +  2*(`npred'+1))/`nobs'
}

* DISPLAY
di in gr _col(1) "AIC Statistic   =  " in ye %11.3f `aic'
end
```

NB-C Log_likelihood function file

```
*! version 1.0.1 25Sep2005 21Jan2009 revision 07Jul2010
* Hilbe Canonical Negative binomial: log likelihood  function :Joseph Hilbe
program define cnbreg_ll
version  9.1
args lnf xb  alpha

tempvar a
qui gen  double `a' = exp(`alpha')
qui replace `lnf' = ($ML_y1* `xb') + (1/`a' * ln(1-exp(`xb')))  +  ///
lngamma($ML_y1 + 1/`a') - lngamma($ML_y1 + 1) - lngamma(1/`a')
end
```

Random Number Generator: Poisson – Rejection Method

```
Example: rndpoi 1000 4  [set obs 1000;  4 is the mean]

program define rndpoi1
   version 3.1
   set type double
   cap drop xp
   qui  {
      local cases `1'
      set obs `cases'
      mac shift
      local xm `1'
      mac shift
      tempvar em t ds sum1 ran1
      local g = exp(-`xm')
      gen `em'= -1
      gen `t' = 1.0
      gen `ran1' = uniform()
      gen `ds' = 1
      count if `ds'>0
      noi di in gr "( Generating " _c
      while _result(1)>0  {
          replace `em' = `em'+ 1 if (`ds'==1)
          replace `t' = `t' * `ran1' if (`ds'==1)
          replace `ds'=0 if (`g' > `t')
          replace `ran1' = uniform()
          noi di in gr "." _c
          count if `ds'>0
      }
      noi di in gr " )"
      gen xp = int(`em'+0.5)
      noi di in bl "Variable " in ye "xp " in bl "created."
      set type float
   }
end
```

Random Number Generator: NB2 – Rejection Method

```
* negative binomial random number generator with mu variable specified
* Example: rndnbl mu, k(0.15)
* k the quadratic variance parameter = 0.15]

program define rndnblx
       version 4.0
        set type double
        local varlist "ex req"
       local options "`options' K(real 1.0)"
       parse "`*'"
       parse "`varlist'",parse(" ")
       qui {
          local mu "`1'"
          mac shift
          tempvar ran1 ran2 ds ts sum1 t em y al c1 c2 c3 sq am
          gen `al'=(`mu'-0.5)*(1-`k')
          gen `c1'=lngamma(`al'+1)-lngamma(`al'+ 1/`k')
          gen `c2'=log(1+1/(`mu'*`k'))
          gen `c3'=lngamma(1/`k')
```

```
        gen `sq'=sqrt(2.0*(`mu'+`mu'*`mu'*`k'))
        if `k' < 1 {
              gen `am'=(`mu'-0.5)*(1-`k')
        }
        else {
              gen `am'=0.0
        }
        gen `ran1'=uniform()
        gen `ran2'=uniform()
        gen `ds'=1
        gen `ts'=1
        gen `em'=-1
        gen `t'=-1
        gen `y'=-1
        egen `sum1'=sum(`ds')
        noi di in gr "( Generating " _c
        while `sum1' > 0 {
*            noi display `sum1'
            replace `y'=sin(_pi*`ran1')/cos(_pi*`ran1')
            replace `em'= `sq'*`y'+`am' if `ds'==1
            replace `ts'=0 if ((0>`em') & (`ds'==1))
            if `k' < 1 {
              #delimit ;
              replace `t'=0.9*(1+`y'*`y')*exp(lngamma(`em'+1/`k') -
                  lngamma(`em'+1) - (`em'-`al')*`c2'+`c1') if
                  ((`ds'==1) & (`ts'==1));
              #delimit cr
            }
            else {
              #delimit ;
              replace `t'=0.9*(1+`y'*`y')*exp(lngamma(`em'+1/`k') -
                  lngamma(`em'+1)-`c3' - `em'*`c2') if
                  ((`ds'==1) & (`ts'==1));
              #delimit cr
            }
              replace `ds'=0 if ((`ran2'<`t') & (`ds'==1) & (`ts'==1))
              replace `ran1'=uniform()
              replace `ran2'=uniform()
              replace `t'=-1
              replace `ts'=1
              drop `sum1'
              egen `sum1'=sum(`ds')
              noi di in gr "." _c
        }
        noi di in gr " )"
        gen xnb =int(`em'+0.5)
        noi di in bl "Variable " in ye "xnb " in bl "created."
         lab var xnb "Constructed negative binomial random variable"
         set type float
   }
end
```

SYNTHETIC ZERO-INFLATED POISSON W LOGIT BINARY COMPONENT

```
* Zero-inflated Poisson with logit binary component
* Joseph Hilbe   zip_syn.do
* LOGIT: x1=-.9, x2=-.1, _c=-.2
* POISSON: x1=.75, n2=-1.25, _c=2, alpha=.5
clear
set obs 50000
set seed 1000
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
* POISSON
gen xb = 2 + 0.75*x1 - 1.25*x2
gen exb = exp(xb)
gen poy = rpoisson(exb)
* BERNOULLI
gen pi =1/(1+exp(-(.9*x1 + .1*x2+.2)))
gen bernoulli = runiform()>pi
gen zy = bernoulli*poy
rename zy y
* ZIP
zip y x1 x2, inf(x1 x2) nolog
```

SYNTHETIC ZERO-INFLATED NB W LOGIT BINARY COMPONENT

```
* Zero inflated Negative binomial with logit as binary component
* Joseph Hilbe  5Jun2011   zinb_syn.do
* LOGIT: x1=-.9, x2=-.1, _c=-.2
* NB2  : x1=.75, n2=-1.25, _c=2, alpha=.5
clear
set obs 50000
set seed 1000
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
* NEGATIVE BINOMIAL- NB2
gen xb = 2 + 0.75*x1 - 1.25*x2
gen a = .5
gen ia = 1/a
gen exb = exp(xb)
gen xg = rgamma(ia, a)
gen xbg = exb * xg
gen nby = rpoisson(xbg)
* BERNOULLI
gen pi =1/(1+exp(-(.9*x1 + .1*x2+.2)))
gen bernoulli = runiform()>pi
gen zy = bernoulli*nby
rename zy y
* ZINB
zinb y x1 x2, inf(x1 x2) nolog
```

PART 4: SAS CODE

A: Censored Poisson macro

SAS macro for estimating a NB2 negative binomial using Genmod. Authored 1995 by Joseph Hilbe and Gordon Johnston.

```
/*************************************************************************/
/*                           SAS SAMPLE LIBRARY                          */
/*                                                                       */
/*                                                                       */
/* NAME   : CENSORED POISSON REGRESSION, Version 1.0                     */
/* TITLE  : CENSORED POISSON REGRESSION                                  */
/* PRODUCT: SAS                                                          */
/* SYSTEM : ALL, VERSIONS 6.08 AND ABOVE                                 */
/* KEYS   :                                                              */
/* PROCS  : SAS/IML                                                      */
/* SUPPORT: SASGJJ                                                       */
/* AUTHORS: Joseph Hilbe, Arizona State University, Tempe, AZ            */
/*              Email: hilbe@asu.edu                                     */
/*          Gordon Johnston, SAS Institute                               */
/*              Email: sasgjj@unx.sas.com                                */
/*                                                                       */
/* MISC   : Date - 7/7/95                                                */
/*                                                                       */
/*************************************************************************/

/*************************************************************************
 |  USAGE:                                                               |
 |  1. Load data to be modeled into memory                               |
 |  2. Load this macro into memory: poicen.sas                           |
 |  3. Call macro with desired options                                   |
 |                                                                       |
 |    %poicen(dsin=<data name>, yvar=<response>, xvars=<predictors>,     |
 |    cenvar=(0=uncensored,1=rgt censored, -1=lft censored),             |
 |    beta=0, noint=(0,1), cov=(0,1), corr=(0,1), dud=(1,0),             |
 |    offvar=<offset>, opvar=(0-6);                                      |
 |                                                                       |
 |    See lines 1 - 15 of macro for additional information.              |
 |                                                                       |
 |  EXAMPLE:                                                             |
 |  Data set named poiex with response xcp, two predictors x1 and x2,    |
 |  and a right censor variable named c.                                 |
 |                                                                       |
 |  %poicen(dsin=poiex, yvar=xcp, xvars=x1 x2, cenvar=c);                |
 |                                                                       |
 *************************************************************************/


%MACRO POICEN ( DSIN =,    /* input data set                          */
             YVAR =,       /* response                                */
             XVAR =,       /* covariates                              */
             CENVAR  =,    /* censor variable: 0=uncensored
                                              1=right censored
                                             -1=left censored       */
             BETA =0,       /* initial parameter estimates           */
             OFFVAR =,      /* offset variable                       */
             NOINT = 0,     /* 1 fits no intercept model             */
             COV   = 0,     /* 1 prints covariance matrix            */
             CORR  = 0,     /* 1 prints correlation matrix           */
```

```
            DUD   = 1,     /* 0 uses analytic 1st derivatives        */
            OPT   = 0);    /* level of iteration history printed for
                              optimization. 0 <= opt <= 6,
                              2 and 6 most useful.                   */

/*---basic syntax error checks---*/
%if %upcase(&dsin) eq %then %do;
   %put You must specify an input data set.;
   %goto out;
   %end;
%if %upcase(&yvar) eq %then %do;
   %put You must specify a response variable.;
   %goto out;
   %end;
%if %upcase(&cenvar) eq %then %do;
   %put You must specify a censor variable.;
   %goto out;
   %end;
%if %upcase(&xvar) eq %then %do;
   %put You must specify a covariate.;
   %goto out;
   %end;

/*--- For optimization ---*/
%if( &DUD ) = 0 %then %let gradstmt = GRD="G_POICEN";
%else %let gradstmt= ;

/*--- Data is first modeled using Poisson regression
  for initial parameter estimates, if none specified ---*/

%if( &BETA = 0 ) %then %do;
   %if %upcase(&offvar) ne %then %let offstmt = OFFSET=&offvar;
   %else %let offstmt= ;

   %if %upcase(&NOINT) ne 0 %then %let intstmt = NOINT;
   %else %let intstmt= ;

   /*--- Turn off printing---*/
   %global _print_;
   %let _print_ = OFF;

   ods output parameterestimates = _A_;
   proc genmod data=&dsin;
      model &yvar = &xvar / dist   = poisson
                            &offstmt
                            &intstmt ;
      run;

   %let _print_ = ON;
   %end;

*OPTION mprint;
OPTION center;
PROC IML ;
RESET noname;

/*---log likelihood--*/
start f_poicen(beta) global(xvar, yvar, cvar, offset, p, nobs);
   sumll = 0;
   n = NROW(offset);
   do i = 1 to nobs;
      xi = xvar[i,];
      etai = xi * beta`;
```

```
      if n > 1 then etai = etai + offset[i];
      mui = exp( etai );
      temp = yvar[i] * etai;
      if cvar[i] = 0 then
         sumll = sumll + yvar[i] * etai - mui - lgamma( yvar[i]+1 );
      if cvar[i] = 1 then
         sumll = sumll + log(probgam( mui, yvar[i] ));
      if cvar[i] = -1 then
         sumll = sumll + log(1 - probgam( mui, yvar[i]+1 ));
   end;
   return( sumll );
finish f_poicen;

/*---gradient---*/
start g_poicen(beta) global(xvar, yvar, cvar, offset, p, nobs);
   g = j(1,p,0);
   n = NROW(offset);
   do i = 1 to nobs;
      xi = xvar[i,];
      etai = xi * beta`;
      if n > 1 then etai = etai + offset[i];
      mui = exp( etai );
      do j = 1 to p;
         if cvar[i] = 0 then
            g[j] = g[j] + xi[j] * ( yvar[i] -  mui );
         if cvar[i] = 1 then do;
            if yvar[i] = 1 then num = exp( -mui );
            else num =
                  poisson( mui, yvar[i]-1 )
                - poisson( mui, yvar[i]-2 );
            g[j] = g[j] + xi[j] * mui * num / probgam( mui, yvar[i]);
            end;
         if cvar[i] = -1 then do;
            num =
               poisson( mui, yvar[i] ) - poisson( mui, yvar[i]-1 );
            g[j] =
              g[j] - xi[j] * mui * num / (1-probgam( mui, yvar[i]+1 ));
            end;
         end;
      end;
   return( g );
finish g_poicen;



/*---Main---*/
start MAIN;
USE &DSIN;
labely={ &YVAR };
labelx={ &XVAR };
labelc={ &CENVAR };
%if %upcase(&offvar) ne %then
   %do;
      labelo={ &OFFVAR };
   %end;
%else
   %do;
      offset={0};
      labelo = {'_0_'};
   %end;
noint ={ &NOINT };

/*  xvar: n X p design matrix           */
```

```
/*  yvar: n X 1 response vector         */
/*  cvar: n X 1 censor indicator vector:
       0 = uncensored
       1 = right censored
      -1 = left censored                */

SETIN &DSIN NOBS nobs;
READ ALL VAR labely  INTO yvar;
READ ALL VAR labelc  INTO cvar;
IF labelo^={'_0_'} THEN DO; READ ALL VAR labelo INTO offset; END;

optn = {1 &opt . . . . . 99};
parm = {1 99};

/*---fit null model for R**2 ---*/
if noint = {0} then do;
   p = 1;
   xvar = j(nobs,1,1);
   x0 = log(yvar[:]);
   call NLPTR(rc, xopt, "f_poicen", x0 ) opt=optn &gradstmt;
   call NLPFDD(liknull, g, h, "f_poicen", xopt )  &gradstmt;
   end;

READ ALL VAR labelx  INTO xvar;

if noint = {0} then do;
   xvar = j(nobs,1,1)||xvar;
   labelx = {"Intercept"}||labelx;
   end;

p=NCOL(xvar);
/*--- Initial values--- */
beta = {&BETA}`;
if  beta = {0} then do;
   USE _A_;
   SETIN _A_;
   READ ALL VAR {ESTIMATE} into beta;
   end;
beta = shape(beta, p,1,0);

/*---fit model---*/
x0 = beta`;
call NLPTR(rc, xopt, "f_poicen", x0 ) opt=optn &gradstmt;
call NLPFDD(likmodel, g, h, "f_poicen", xopt ) par=parm &gradstmt;

/*---print summary of fit table---*/
print " "; print " ";
print "Summary of Fit";
if noint = {0} then do;
   lrt = 2. * ( likmodel - liknull );
   plrt = 1. - probchi( lrt, p - 1 );
   rsq0 = 1. - exp( - 2. * ( likmodel - liknull ) / nobs );
   rsq1 = 1. - likmodel / liknull;
   labelf = {"No. of Obs." "Model LRT Chi**2" "Prob>Chi**2" "R**2"
              "Pseudo R**2" "Log Likelihood"};
   fitsum = nobs//lrt//plrt//rsq0//rsq1//likmodel;
   end;
else do;
   labelf = {"No. of Obs." "Model LRT Chi**2" "Prob>Chi**2"
             "Log Likelihood"};
   fitsum = nobs//lrt//plrt//likmodel;
   end;
print fitsum[rowname=labelf];
```

```
/*---std errs, etc. ---*/
cov = -inv( h );
prob = .05; noqua = probit( 1. - prob / 2. );
stderr = sqrt(abs(vecdiag(cov)));
beta = xopt`;
xlb = beta - noqua * stderr;
xub = beta + noqua * stderr;
z = beta / stderr;
probz = 1. - probnorm( abs(z) );

/*---print parameter estimates table---*/
output = beta||xlb||xub||stderr||z||probz;
labelt = {"Estimate" "Lower" "Upper" "Std. Err." "z" "Prob>|z|"};
print "Censored Poisson Parameter Estimates";
print "Response Variable: %upcase(&YVAR)";
print output[rowname=labelx colname=labelt format=10.4];

if( &COV = 1 ) then do;
   print "Estimated Covariance Matrix";
   print cov[rowname=labelx colname=labelx];
   end;

if( &CORR = 1 ) then do;
   s = 1. / stderr;
   corr = diag(s) * cov * diag(s);
   print "Estimated Correlation Matrix";
   print corr[rowname=labelx colname=labelx];
   end;

finish MAIN;
run MAIN;


QUIT;
%out: %MEND;
```

B: Censored Negative Binomial Macro

This macro was contributed by Niel Hens (niel.hens@uhasselt.be) on 10 February, 2008. It is based on SAS PROC NLMIXED.

```
 proc nlmixed data=data1;
title 'Weighted censored negative binomial model (unknown y)';
eta = b0 + b1*age2 + b2*age3 + b3*age4 + b4*age5 + b5*age6 + b6*age7 + b7*age8 + b8*age9 +
b9*age10 + b10*ageM
+ c1*sexMa + c2*sexM
+ d1*hh2 + d2*hh3 + d3*hh4 + d4*hh5 + d5*hh6
+ e1*dowMo + e2*dowTu + e3*dowWe + e4*dowTh + e5*dowFr + e6*dowSa +e7*dowM
+ f1*DE + f2*FI + f3*GB + f4*IT + f5*LU + f6*NL + f7*PL;
mean = exp(eta);
select;
when (cens=1) lc=0;
do j=0 to (y-1);
lc=lc+exp(lgamma(j+(1/k)) - lgamma(j+1) - lgamma(1/k) + j*log(k*mean) - (j+(1/k))*log(1+k*mean));
end;
llc= log(1-lc);
ll = weight*llc;
when (cens=0) ll = weight*(lgamma(y+(1/k)) - lgamma(y+1) - lgamma(1/k) + y*log(k*mean) -
(y+(1/k))*log(1+k*mean));
end;
model y ~ general(ll);
run;
```

Part 5: Comments to discussion in text

Chapter 6, following Table 6.22 (just prior to summary)

The following may be added to the discussion in the text directly following Table 6.22. It discusses the relationship of the factor level coefficients when they are inverted.

==========

Factor variables can be reversed by subtracting the factor variable from n+1. For example, given the 4 levels of *killip* as above described, create an inverse relationship by the following and compare with the previous model output.

```
. gen byte dkillip = 5-killip
. tab dkillip, gen(dk)

. glm die anterior hcabg dk1-dk3,fam(poi) eform lnoffset(cases) nolog nohead

------------------------------------------------------------------------------
             |                 OIM
         die |        IRR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    anterior |   1.963766   .3133595     4.23   0.000     1.436359    2.684828
       hcabg |   1.937465   .6329708     2.02   0.043     1.021282    3.675546
         dk1 |   12.33746   3.384215     9.16   0.000     7.206717    21.12096
         dk2 |   3.044349   .7651196     4.43   0.000      1.86023    4.982213
         dk3 |   2.464633   .4247842     5.23   0.000      1.75811    3.455083
       cases |  (exposure)
------------------------------------------------------------------------------
```

Estimating the coefficients shows a more interesting relationship:

```
. glm die anterior hcabg kk2-kk4,fam(poi) lnoffset(cases) nolog nohead

------------------------------------------------------------------------------
             |                 OIM
         die |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    anterior |   .6748639   .1595707     4.23   0.000     .3621111    .9876168
       hcabg |   .6613804   .3267006     2.02   0.043      .021059    1.301702
         kk2 |   .9020431   .1723519     5.23   0.000     .5642396    1.239847
         kk3 |   1.113287   .2513246     4.43   0.000        .6207    1.605874
         kk4 |    2.51264   .2743041     9.16   0.000     1.975014    3.050266
       _cons |   -4.06977   .1459076   -27.89   0.000    -4.355743   -3.783796
       cases |  (exposure)
------------------------------------------------------------------------------

. glm die anterior hcabg  dk2-dk4,fam(poi) lnoffset(cases) nolog nohead

------------------------------------------------------------------------------
             |                 OIM
         die |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    anterior |   .6748639   .1595707     4.23   0.000     .3621111    .9876168
       hcabg |   .6613804   .3267006     2.02   0.043      .021059    1.301702
         dk2 |  -1.399353   .3358858    -4.17   0.000    -2.057677   -.7410287
         dk3 |  -1.610597   .2828356    -5.69   0.000    -2.164944   -1.056249
```

```
         dk4 |   -2.51264   .2743041    -9.16   0.000    -3.050266   -1.975014
       _cons |   -1.55713   .2757328    -5.65   0.000    -2.097556   -1.016703
       cases | (exposure)
------------------------------------------------------------------------------

The value of dk4 is the reverse sign of kk4,

. di _b[dk4] - _b[dk3]  = -.90204308

is the reverse sign of kk2, and

. di _b[dk4] - _b[dk2] = -1.113287

is the reverse sign of kk3
=================
```

Chapter 13: Sources

I recommend reading

Cameron, A.C. & P.K. Trivedi, *Microeconometrics Using Stata*, College Station, TX: Stata Press

for excellent presentations of several models discussed in this chapter. Their presentations of finite mixture and quantile count models were in particular influential to the manner in which I presented the material. For those wanting more material on finite mixture models, I recommend

McLachlan, G.J., and D. Peel (2000), *Finite Mixture Models*, New York: John Wiley.

which was also influential in shaping this section.

In addition to Cameron and Trivedi, an excellent resource for quantile count models is the journal article,

Machado, J., J.M.C. Santos Silva (2005), Quantiles for Counts. *Journal of the American Statistical Association* 100: 1226--1237.

Part 6: Stata references for download of user commands using in text

STATA FILES REFERRED TO IN TEXT

Jann, B. (2007). center: module to center (or standardize) variables,
Statistical Software Components (SSC) archive, http://fmwww.bc.edu/repec/bocode/c/center

Simons, J.S. (2004). mcenter: module to center variables at their means,
Statistical Software Components (SSC) archive, http://fmwww.bc.edu/repec/bocode/m/mcenter

Part 7: Tables and Scripts – Stata and R

Table 6.15: R *Observed v predicted counts*

```
=========================================
load("c://source/medpar.RData")
qui {
medp <- glm(los ~ hmo + white + type2 + type3, family=poisson, data=medpar)
# predicted value at each LOS 0-25
mu <- fitted.values(medp)
for (i in 0:25) {
  p[i] <- exp(- mu[i])*(mu[i]^i)/factorial(i)
}
use c:\source\medpar
qui glm los hmo white type2 type3, fam(poi)
. predict mu
. local i 0
. local newvar "pr`i'"
*: Predicted probability at each los
. while `i' <=25 {
  2. local newvar "pr`i'"
  3. qui gen `newvar' =  exp(-mu)*(mu^`i')/exp(lnfactorial(`i'))
  4. local i = `i' + 1
  5. }
. quietly gen cnt = .
. quietly gen obpr = .
. quietly gen prpr = .
. local i 0
*: Observed and predicted los
. while `i' <=25 {
  2. local obs = `i' + 1
  3. replace cnt = `i' in `obs'
  4. tempvar obser
  5. gen `obser' = (los==`i')  /* generic = `e(depvar)' */
  6. sum `obser'
  7. replace obpr = r(mean) in `obs'
  8. sum pr`i'
  9. replace prpr = r(mean) in `obs'
 10. local i = `i' + 1
 11. }
*: Preparation for table
. gen obsprop = obpr*100   /* outcomes equal to # */
. gen preprop = prpr*100   /* average predicted prob */
. gen byte count = cnt
. gen diffprop = obsprop - preprop
. format obsprop preprop diffprop %8.3f
}
. l count obsprop preprop diffprop in 1/21
* Code for Fig 6.2 in text

# R  Complete code on books web site
obsprop <- obpr*100
preprop <- prpr*100
count <- cnt
diffprop <- obsprop – preprop
tpr <- cbind(count, obsprop, preprop, diffprop)
tpr[1:21, 1:4]
```

```
matplot(cbind(count, count), cbind(obs, pred),
  pch=1:2, col=1:2, xlab='Count', ylab='Frequency?') #points
matplot(cbind(count, count), cbind(obs, pred), type='l',
  lty=1:2, col=1:2, xlab='Count', ylab='Frequency?') #lines
=======================================
```

ADVICE:
Using the R **ggplot2** function. Main trick is to have a set of X,Y values all with the same factor value. If the first 3 are one set of X,Y values and the second 3 are a new set, it looks like:

```
Factor, x, y
1        1,20
1        2,27
1        3,32
2        1,21
2        2,26
2        3,33
```

Then plot X against Y with lines, points, etc. done by Factor

Graph two distributons on count: on same graph (observed vs predicted)

```
count<- 0:4
obs<- c(0.000, 8.428, 4.749, 5.017, 6.957)
pred<- c(0.012, 0.108, 0.471, 1.381, 3.047)

layout(t(1:2)) #Side-by-side plots t(1:1) for one graph
plot(rbind(count,count), rbind(obs, pred), xlab='Count',
  ylab='Frequency?', pch=1:2, col=1:2) #plot as points
plot(count, obs, type='l', lty=1, col=1, ylab='Frequency?') #plot as lines
lines(count, pred, lty=2, col=1)   # col=1 if separate graphs desired
```

Figure 8.12

```
=====================================================
* POISSON
set obs 30
gen byte mu = 10
gen byte y = _n-1
gen yp =  (exp(-mu)*mu^y)/exp(lngamma(y+1))
* graph twoway connected yp y,

* NB-2 alpha = .5
gen alpha = .1
gen amu = mu*alpha
gen ynb_5 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) /*
  */  - lngamma(y+1) - lngamma(1/alpha))
* graph twoway connected ynb_5 yp y

* NB-2 alpha = 1
gen alpha2 = .3
gen amu2 = mu*alpha2
gen ynb1 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) /*
  */  - lngamma(y+1) - lngamma(1/alpha2))

* NB-2 alpha=2
gen alpha3 = .5
gen amu3 = mu*alpha3
```

```
gen ynb2 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) /*
   */  - lngamma(y+1) - lngamma(1/alpha3))
lab var yp "Poisson"
lab var ynb_5 "NB-2;alpha=.1"
lab var ynb1  "NB-2;alpha=.3"
lab var ynb2  "NB-2;alpha=.5"
graph twoway connected ynb2 ynb1 ynb_5 yp  y, ms(s x d T)
=====================================================================
```

Figure 8.13

```
=====================================================================
* MEAN = 10  ; Alpha .6, .8, 1, 1.2
* POISSON
set obs 30
gen byte mu = 10
gen byte y = _n-1

* NB-2 alpha = .6
gen alpha = .6
gen amu = mu*alpha
gen ynb_6 = exp(y*ln(amu/(1+amu)) - (1/alpha)*ln(1+amu) + lngamma(y +1/alpha) /*
  */  - lngamma(y+1) - lngamma(1/alpha))

* NB-2 alpha = .8
gen alpha1 = .8
gen amu1 = mu*alpha1
gen ynb_8 = exp(y*ln(amu1/(1+amu1)) - (1/alpha1)*ln(1+amu1) + lngamma(y +1/alpha1) /*
  */  - lngamma(y+1) - lngamma(1/alpha1))

* NB-2 alpha = 1
gen alpha2 = 1
gen amu2 = mu*alpha2
gen ynb_10 = exp(y*ln(amu2/(1+amu2)) - (1/alpha2)*ln(1+amu2) + lngamma(y +1/alpha2) /*
  */  - lngamma(y+1) - lngamma(1/alpha2))

* NB-2 alpha=1.2
gen alpha3 = 1.2
gen amu3 = mu*alpha3
gen ynb_12 = exp(y*ln(amu3/(1+amu3)) - (1/alpha3)*ln(1+amu3) + lngamma(y +1/alpha3) /*
   */  - lngamma(y+1) - lngamma(1/alpha3))
lab var ynb_6 "NB-2;alpha=.6"
lab var ynb_8 "NB-2;alpha=.8"
lab var ynb_10 "NB-2;alpha=1"
lab var ynb_12 "NB-2;alpha=1.2"
graph twoway connected ynb_12 ynb_10 ynb_8 ynb_6 y, ms(s x d T)
=====================================================================
```

Table 10.11 R *NB-C solved using optimization*

```
===================================
rm(list=ls())
load("c://source/medpar.RData")
medpar$type <- factor(medpar$type)

nbc.reg.ml <- function(b.hat, X, y) {
  a.hat <- b.hat[1]
  xb.hat <- X %*% b.hat[-1]
  mu.hat <- 1 / ((exp(-xb.hat)-1)*a.hat)
  p.hat <- 1 / (1 + a.hat*mu.hat)
  r.hat <- 1 / a.hat
```

```
  sum(dnbinom(y,
              size = r.hat,
              prob = p.hat,
              log = TRUE))
}
# Create the design matrix
nbcX <- model.matrix(~ hmo + white + type, data = medpar)

# Starting points (discovered by trial and error!)
p.0 <- c(alpha = 0.5,
         cons = -1,
         hmo = 0,
         white = 0,
         type2 = 0,
         type3 = 0)

# Maximize the joint conditional LL
nbc.fit <- optim(p.0,
                 nbc.reg.ml,
                 X = nbcX,
                 y = medpar$los,
                 control = list(
                   fnscale = -1,
                   maxit = 10000),
                 hessian = TRUE
                 )

# and obtain the parameter estimates and asymptotic SE's by
(nbc.beta.hat <- nbc.fit$par)
(nbc.se.beta.hat <- sqrt(diag(solve(-nbc.fit$hessian))))
nbc.results <- data.frame(Estimate = nbc.beta.hat,
                          SE = nbc.se.beta.hat,
                          Z = nbc.beta.hat / nbc.se.beta.hat,
                          LCL = nbc.beta.hat - 1.96 * nbc.se.beta.hat,
                          UCL = nbc.beta.hat + 1.96 * nbc.se.beta.hat)
rownames(nbc.results) <- c("alpha", colnames(nbcX))
nbc.results <- nbc.results[c(2:nrow(nbc.results), 1),]
nbc.results
=======================================================
```

R OUTPUT

```
> nbc.results
               Estimate          SE           Z         LCL          UCL
(Intercept) -0.20128564 0.013496886 -14.913488 -0.22773954 -0.174831744
hmo         -0.01399781 0.010729402  -1.304622 -0.03502744  0.007031817
white       -0.02491594 0.010756500  -2.316361 -0.04599868 -0.003833201
type2        0.04101961 0.008932639   4.592105  0.02351164  0.058527585
type3        0.10723427 0.010062008  10.657343  0.08751273  0.126955803
alpha        0.44516214 0.019775949  22.510280  0.40640128  0.483922999
```

Table 11.4 R: *Synthetic NB2 – logit hurdle model*

```
====================================
# syn.hurdle_lnb2.r
library(MASS)
library(pscl)
nobs <- 50000
x1 <- runif(nobs)
x2 <- runif(nobs)
xb <- 2 + .75*x1 - 1.25*x2
a <- .5
```

```
ia <- 1/.5
exb <- exp(xb)
xg <- rgamma(nobs, a, a, ia)
xbg <-exb*xg
nby <- rpois(nobs, xbg)
nbdata <- data.frame(nby, x1, x2)
nby <- nbdata[nbdata$nby!=0, ]
pi <- 1/(1+exp(-(.9*x1 + .1*x2 + .2)))
bern <- runif(nobs)>pi
bern <- as.numeric(bern)
jhObs <- which( nbdata$bern==0 ) #  nbdata$nby <- ifelse(bern==0, 0, nbdata$nby)
nbdata$nby[jhObs] <- 0           #  hy <- nbdata$nby
hy <- nby
hlnb2 <- hurdle(hy ~ x1 + x2, dist="negbin",
      zero.dist= "binomial", link="logit", data=nbdata)
summary(hlnb2)
=====================================================
```

CODE NEEDED FOR ALL TRUNCATED MODELS, LOAD *MEDPAR* DATA FRAME

```
library(gamlss)
library(gamlss.tr)
load("c://source/medpar.RData")

# LEFT TRUNCATED AT 3
tL3 <- subset(medpar, medpar$los>3)
model1 <- gamlss(los~white+hmo, family=trun(3, "NBI", "left"), data=tL3)
summary(model1)

# ZERO-TRUNCATED NB
tL0 <- subset(medpar, medpar$los !=0)
# or  tL0 <- subset(medpar, medpar$los<1)   IS THIS CORRECT AS WELL
model2 <- gamlss(los~gender+age, family=trun(0, "NBI", "left"), data=tL0)
summary(model2)

# RIGHT TRUNCATED AT 20
tR20 <- subset(medpar, medpar$los<20)
model3 <- gamlss(los~gender+age, family=trun(20, "NBI", "right"), data=tL0)
summary(model2)

# GLOBAL  LEFT TRUNCATED AT 3
tL3 <- subset(medpar, medpar$los>3)
gen.trun(3, "NBI", "left")
model4 <- gamlss(los~gender+age, family=NBIleft, data=tL3)
```

left censoring at y gives likelihood F(y), (i,e, Y<=y so exact y is included in the censoring probability) right censoring at y gives likelihood [1-F(y)], (i.e. Y>y+1 so exact y is NOT included in the censoring probability) dPOleft pPOleft qPOleft rPOleft POleft

Censored Poisson

Stata

```
CENSORED POISSON REGRESSION  - econometric parameterization
*! Version 3.0.4    Joseph Hilbe :  27Sep2009
program cpoissone, properties(svyb svyj svyr)
    version 11
    syntax [varlist] [if] [in] [fweight pweight aweight iweight] [,    ///
        CENsor(string) Level(cilevel)  CLEft(real 1) CRIght(real 1)         ///
        OFFset(passthru) EXposure(passthru) noLOG        ///
        CLuster(passthru) IRr Robust FROM(string asis) * ]
    gettoken  lhs rhs : varlist
    marksample touse
    mlopts mlopts, `options'
    if ("`weight'" != "")   local weight "[`weight'   `exp']"
    if (`"`from'"' != `""') local initopt `"init(`from')"'
    global S_cen  "`censor'"
    local clft = `cleft'
    global Sclt   `"`clft'"'
    local crgt = `cright'
    global Scrg   `"`crgt'"'
    qui count if  !inlist(`censor',-1,0,1)
    if  r(N)>0   {
        noi di as txt                        ///
        "Note: `r(N)' values of `censor' are not one of -1, 0,  1"
        qui replace `touse' = 0  if  !inlist(`censor',-1,0,1)
    }
    ml model lf cpoisxll (xb: `lhs' =  `rhs', `offset' `exposure')    ///
        if `touse' `weight',                    ///
        `mlopts' `robust' `cluster'                ///
        title("Censored Poisson Regression")            ///
        maximize `log' `initopt'
    ml display, level(`level') `irr'
end

Censored Poisson (econometric) : Log-likelihood function
program cpoisxll
    version  11
    args lnf xb
    local censor "$S_cen"
    local cleft   $Sclt
    local cright  $Scrg

    qui replace `lnf' = cond(`censor' == 1,                ///
        -exp(`xb') + $ML_y1 * `xb' - lngamma($ML_y1 + 1),    ///
        ln(poisson(exp(`xb'), `cleft')) )
    qui replace `lnf' = ln(poissontail(exp(`xb'), `cright')) if `censor' == -1
end
```

Stata

Synthetic Poisson-logit Hurdle Model: hplogit.do

```
=======================================
* SYNTHETIC POISSON-LOGIT HURDLE MODEL
* Joseph Hilbe  3Jun2011   hplogit.do
* LOGIT: x1=-.9, x2=-.1, _c=-.2
* POISSON  : x1=.75, n2=-1.25, _c=2
clear
set obs 50000
set seed 1000
gen x1 = invnorm(runiform())
gen x2 = invnorm(runiform())
* POISSON
gen xb = 2 + 0.75*x1 - 1.25*x2
gen exb = exp(xb)
gen py = rpoisson(exb)
* BERNOULLI
drop if py==0
gen pi =1/(1+exp(-(.9*x1 + .1*x2+.2)))
gen bernoulli = runiform()>pi
replace py=0 if bernoulli==0
rename py y
* POISSON-LOGIT HURDLE
hplogit y x1 x2, nolog
* TEST - optional
* logit bernoulli x1 x2, nolog /// test
* ztp y x1 x2 if y>0, nolog   /// test
===================================
```

I adapted this code from another source, since forgotten. It was originally written in the mid 1990s, and employs an older style of maximum likelihood estimation using Stata. I would appreciate acknowledging the authorship of the original command if known, and any related details.

CONDITIONAL FIXED EFFECTS NEGATIVE BINOMIAL

```
*! version 1.0.1 042498  1Mar2005 J. Hilbe
program define fenbreg
  version 5.0
  local varlist "req ex"
  local if "opt"
  local in "opt"
  local options "I(string) nolog TRace"
  parse "`*'"
  if "`log'" == "nolog"  {
    local pre1 "qui"
    }
  else local pre1 "noi"
  xt_iss `i'
  local ivar "$S_1"
  parse "`varlist'", parse(" ")
  local depv "`1'"
  mac shift 1
  local indv "`*'"
```

```
  tempvar touse
  mark `touse' `if' `in'
  markout `touse' `depv' `indv'
  global S_mark "`touse'"
  global S_ivar "`ivar'"
  sort $S_mark $S_ivar
  tempname b f0 f V0 V
/* first estimate baseline model (constant only) */
  ml begin
  ml function fenbll
  ml method deriv0
  eq `depv' : `depv'
  ml model `b' = `depv' `pre1'
  ml maximize `f0' `V0'
/* Main model */
  ml begin
  ml function fenbll
  ml method deriv0
  eq `depv' `indv'
  ml model `b' = `depv' `pre1'
  ml maximize `f' `V', `trace'
  ml post fenegbin, title("Conditional Fixed Effects Negative Binomial Regression")
  ml mlout fenegbin
end
================================================================
```

Conditional fixed effects negative binomial: LL function

```
================================================================
program define fenbll
version 5.0
  local b "`1'"
  local f "`2'"
  tempvar prdn rr a1 a2 a3 a4 a5 a6
  quitely {
     matrix score double `prdn' = `b' if $S_mark
     replace `prdn' = exp(`prdn')
     #delimit ;
     by $S_mark $S_ivar: gen double `a1' =
           cond(_n==_N,sum(lngamma(`prdn'+$S_mldepn)),0) if $S_mark;
     by $S_mark $S_ivar: gen double `a2' =
           cond(_n==_N,sum(lngamma(`prdn')),0) if $S_mark;
     by $S_mark $S_ivar: gen double `a3' =
           cond(_n==_N,sum(lngamma($S_mldepn+1)),0) if $S_mark;
     by $S_mark $S_ivar: gen double `a4' =
           cond(_n==_N,lngamma(sum(`prdn')),0) if $S_mark;
     by $S_mark $S_ivar: gen double `a5' =
           cond(_n==_N,lngamma(sum($S_mldepn)+1),0) if $S_mark;
     by $S_mark $S_ivar: gen double `a6' =
           cond(_n==_N,lngamma(sum($S_mldepn)+sum(`prdn')),0) if $S_mark;
  #delimit cr
  gen double `rr' = `a1'- `a2' - `a3' + `a4' + `a5' - `a6' if $S_mark
  replace `rr' = sum(`rr') if $S_mark
  scalar `f' = `rr'[_N]
  }
end
================================================================
```

R Bivariate Poisson model

```
library(bivpois)
library(Hmisc)
# data("rwm") if data stored in R format.
rwm <- stata.get("c://source/rwm.dta")
# ----------------------------
#OR
library(COUNT)
data(rwm)
#-----------------------------
docvis <- as.numeric(rwm$docvis)
hospvis <- as.numeric(rwm$hospvis)
reform <- as.numeric(rwm$reform)
outwork <- as.numeric(rwm$outwork)
hhkids <- as.numeric(rwm$hhkids)
visit <- data.frame(docvis, hospvis, reform, outwork, hhkids)
bvp <- lm.bp(docvis~reform+outwork+hhkids, hospvis~reform+
outwork+hhkids, data=visit)

# use the following to obtain a list of coefficients
bvp$coef
dp <- lm.bp(docvis~reform+outwork+hhkids, hospvis~reform+outwork+
hhkids, data=visit, zeroL3=TRUE)
```

Part 8: Derivations

A: Quasi-likelihood: calculation from QL definition of negative binomial to the negative binomial quasi-likelihood function

$$2\int_{\mu}^{y}\frac{y-\mu}{\mu+k\mu^2}\,d\mu$$

$$\int\frac{a-x}{x+bx^2}\,dx$$

$$=aln(x)+\frac{-1-ab}{b}ln(1+bx)$$

Let $\alpha=y$; $x=\mu$; $b=k$

$$=yln(\mu)+\frac{-1-yk}{k}ln(1+ky)$$

$$=yln(\mu)-\frac{1+yk}{k}ln(1+ky)$$

$$2\left\{yln(\mu)-\frac{1+yk}{k}ln(1+ky)-yln(\mu)+\frac{1+yk}{k}ln(1+ky)\right\}$$

$$2\left\{yln\left(\frac{y}{\mu}\right)+\frac{1+ky}{k}ln\left(\frac{1+k\mu}{1+ky}\right)\right\}$$

$$2\left\{yln\left(\frac{y}{\mu}\right)-\frac{1+ky}{k}ln\left(\frac{1+ky}{1+k\mu}\right)\right\}$$