Lab session 7: Maximum likelihood, redux

Adam Theising[*]

March 14th, 2018

Contents

Getting started...

- As always, questions/comments?...
- Sketched solutions and our code for assignment 3 to come by this weekend

1 Maximum likelihood estimation

This week, Brian has asked me to drive home the notion that the distribution of a dependent variable determines the functional form of a sample's likelihood function. So to illustrate this, let's begin by deriving log-likelihood functions for a handful of common distributional assumptions we make on the dependent variable.

1.1 *Normal distribution*

First, we'll revisit some material that we covered at the end of Lab 5: consider the classical linear regression model with a normally distributed error term. That is to say, conditioned on covariates $\mathbf{X_i}$, y_i is normally distributed with $\mu_i = \mathbb{E}(y_i|X_i) = \mathbf{X_i}'\beta$ and variance σ^2. As we showed in lab 5, this yields log-likelihood function (LLF):

$$\ell(\beta, \sigma^2|X, y) = \sum_{i=1}^{N} \ln\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(y_i - X_i\beta)^2}{2\sigma^2}\right)\right] = -\frac{N}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - X_i\beta)^2$$

To provide some additional intuition on this result, recall that the (log)-likelihood function, informally-speaking, provides us with a sense of how likely we were to have observed our data, given a parameter value. So in the LLF above, note that the 2nd term on the right hand side is increasing as a proxy for model error ($\sum_{i=1}^{N}(y_i - X_i\beta)^2$) is decreasing. Thus, selecting β in a fashion that lowers model error will also increase the value of our likelihood function. There's some deep and beautiful symmetry going on here due to our normality assumption.

[*] theising@wisc.edu

1.2 *Bernoulli distribution*

All right, let's move on to another distributional assumption on our dependent variable. Here, let's assume our outcome variable, y_i, is Bernoulli distributed, with probability mass function:

$$f(y_i|\mathbf{X_i}) = p_i^{y_i}(1-p_i)^{1-y_i}$$

where $y_i = 0, 1$ and $p_i = F(\mathbf{X_i}'\beta)$. For those who are less familiar with the Bernoulli distribution, this is the "coin flip" distribution - it's suitable for a binary dependent outcome where respective outcome probabilities are given by p_i and $1 - p_i$. Note, however, that we've defined the probability parameter, p_i, as a function of covariates $\mathbf{X_i}$ and β; thus, we can back out our LLF as:

$$\ell(\beta|x_i, y_i) = \sum_{i=1}^{N}\left[y_i \ln\left(F(\mathbf{X_i}'\beta)\right) + (1-y_i)\ln\left(1 - F(\mathbf{X_i}'\beta)\right)\right]$$

One initial takeaway here is that the LLF varies quite a bit from the one generated with a normally distributed dependent variable. This should be fairly intuitive - *the likelihood function is derived directly from distributional assumptions we place on our dependent variable*. Moreover, this distributional assumption bounds us in the $[0, 1]$ space for estimation of our outcome variable - this is attractive when a least-squares approach may result in nonsensical estimates outside of that bounded space. A second takeaway - and one that we'll revisit in a few weeks once Brian's lectured on binary choice models - is that there remains some flexibility despite our structural assumption on the dependent variable's distribution. If we specify $F(\mathbf{X_i}'\beta) = \Lambda(\mathbf{X_i}'\beta) = \frac{e^{X_i'\beta}}{1+e^{X_i'\beta}}$, we are estimating a *logit* model, while if we specify $F(\mathbf{X_i}'\beta) = \Phi(\mathbf{X_i}'\beta) = \int_{-\infty}^{\mathbf{X_i}'\beta} \phi(z)dz$ we are estimating a *probit* model. Logit and probit models are the "ground floor" of the discrete choice literature, and important tools for anyone doing applied work... more on this to come soon.

1.3 *Poisson distribution*

Finally, let's consider the Poisson distribution. Recall that in Lab 5, we derived the maximum likelihood estimator (MLE) for an outcome variable following this distribution. The probability mass function we used was $f(y_i|\theta) = \frac{e^{-\theta}\theta^{y_i}}{y_i!}$. An interesting characteristic of this distribution is that its first and second moments are identical. That is to say, a feature (or flaw, depending on context) of the Poisson distribution is that the parameter θ defines both the *mean* and the *variance* of the distributed data. We will revisit this below in our discussion of count models, but for now, recognize that an obvious difference between this distribution and the normal distribution is its definition by a single parameter.

As with the two distributions above, a common extension of the Poisson MLE is to allow the parameter θ to be a function of exogenous variables of interest, X_i. A fairly standard approach is by exponential mean parameterization: $\theta = e^{X_i'\beta}$. This is attractive because (a) it restricts $\theta > 0$ and (b) yields covariate-specific β values. Furthermore, calculating the LLF for this set of distributional assumptions, we have a globally concave likelihood function:

$$\ell(\beta|X_i, y_i) = \sum_{i=1}^{n}[-e^{X_i'\beta} + y_i X_i\beta - \ln(y_i!)]$$

Since the function is globally concave, we are guaranteed a unique maximum![1] And to reiterate one final time, this LLF was defined implicitly by the distributional assumption we made on our dependent variable. We'll pick up here again in a minute, but first...

[1] This isn't obvious to you on a first pass? Prove it to yourself - take the second-order conditions of the LLF with respect to β.

1.4 *A quick review of estimation methods*

Just as a refresher - and so that you have this information in one concise location - let's review the algorithms that you might consider using for maximum likelihood estimation. As with minimizing least square functions, the key difference between these algorithms is the step length used; a general form for all three algorithms can be written as:

$$\beta_{n+1} = \beta_n - vP_n \frac{\partial \ell(\cdot)}{\partial \beta}\bigg|_{\beta=\beta_n}$$

Where v is a variable step length, $\ell(\cdot)$ is the log-likelihood function to be maximized, and P_n is the algorithm-specific step length. For each algorithm below, we'll start with the *general* formulation of the algorithm, then as an example, show a *specific* formulation of it under the assumption of a normally distributed dependent variable. For these normally-distributed formulations, refer back to the LLF in section 1.1 and note that $\sum_{i=1}^{N}(y_i - X_i\beta)^2 \equiv S(\beta)$.

Newton-Raphson

Here, P_n is the inverse Hessian of the LLF. In general form:

$$\beta_{n+1} = \beta_n - v\left[\frac{\partial^2 \ell(\cdot)}{\partial\beta\partial\beta'}\right]^{-1}\Bigg|_{\beta_n}\left[\frac{\partial \ell(\cdot)}{\partial \beta}\right]\Bigg|_{\beta_n}$$

If we take the normally distributed LLF, we have:

$$\beta_{n+1} = \beta_n - v\left[\frac{-1}{2\sigma^2}\frac{\partial^2 S(\beta)}{\partial\beta\partial\beta'}\right]^{-1}\Bigg|_{\beta_n}\left[\frac{-1}{2\sigma^2}\frac{\partial S(\beta)}{\partial \beta}\right]\Bigg|_{\beta_n}$$

$$\beta_{n+1} = \beta_n - v\left[\frac{\partial^2 S(\beta)}{\partial\beta\partial\beta'}\right]^{-1}\Bigg|_{\beta_n}\left[\frac{\partial S(\beta)}{\partial \beta}\right]\Bigg|_{\beta_n}$$

Method of scoring (Gauss-Newton)

Here, P_n is the inverse of the *expectation* of the LLF's Hessian. In general form:

$$\beta_{n+1} = \beta_n - v\left[\mathbb{E}\left(\frac{\partial^2 \ell(\cdot)}{\partial\beta\partial\beta'}\right)\right]^{-1}\Bigg|_{\beta_n}\left[\frac{\partial \ell(\cdot)}{\partial \beta}\right]\Bigg|_{\beta_n}$$

If we specify a normally distributed LLF, we have:

$$\beta_{n+1} = \beta_n - v\left[\frac{-1}{\sigma^2}Z(\beta_n)'Z(\beta_n)\right]^{-1}\left[\frac{-1}{2\sigma^2}\frac{\partial S(\beta)}{\partial \beta}\right]\Bigg|_{\beta_n}$$

$$\beta_{n+1} = \beta_n - \frac{v}{2}\left[Z(\beta_n)'Z(\beta_n)\right]^{-1}\frac{\partial S(\beta)}{\partial \beta}\bigg|_{\beta_n}$$

If you look carefully at the normally-distributed formulations of these first two MLE algorithms, you'll notice they're **identical** to the algorithms we used for NLS estimation. Only difference here is that we're maximizing a likelihood function instead of minimizing a sum of square errors function. At the end of the day, these are the same thing when assuming a normally distributed error term.

BHHH

Finally, we'll revisit a third algorithm that's a bit different from the others. Here, we need the log-likelihood for each *single* observation - note the *i* subscript below. In general form:

$$\beta_{n+1} = \beta_n + v \left[\sum_{i=1}^{N} \left(\frac{\partial \ell_i(\beta)}{\partial \beta} \right) \left(\frac{\partial \ell_i(\beta)}{\partial \beta} \right) \right]^{-1} \Bigg|_{\beta_n, \sigma_n^2} \frac{\partial \ell}{\partial \beta} \Bigg|_{\beta_n}$$

For the normally distributed LLF, note that[2]

$$\frac{\partial \ell_i(\beta)}{\partial \beta} = \left(\frac{y_i - f(x_i, \beta)}{\sigma^2} \right) \left(\frac{\partial f(x_i, \beta)}{\partial \beta} \right)$$

Then the normal-distribution specific formulation of the BHHH algorithm is given by:

$$\begin{aligned}
\beta_{n+1} &= \beta_n + v \left[\sum_{i=1}^{N} \left(\frac{(y_i - f(x_i, \beta))^2}{\sigma^4} \right) \left(\frac{\partial f(x_i, \beta)}{\partial \beta} \right) \left(\frac{\partial f(x_i, \beta)}{\partial \beta'} \right) \right]^{-1} \Bigg|_{\beta_n, \sigma_n^2} \left(\frac{-1}{2\sigma^2} \frac{S(\beta)}{\partial \beta} \right) \Bigg|_{\beta_n} \\
\beta_{n+1} &= \beta_n - \frac{v}{2\sigma_n^2} \left[\sum_{i=1}^{N} \left(\frac{(y_i - f(x_i, \beta))^2}{\sigma^4} \right) \left(\frac{\partial f(x_i, \beta)}{\partial \beta} \right) \left(\frac{\partial f(x_i, \beta)}{\partial \beta'} \right) \Bigg|_{\beta_n} \right]^{-1} \left(\frac{S(\beta)}{\partial \beta} \right) \Bigg|_{\beta_n}
\end{aligned}$$

Variance-covariance estimation

Finally, a quick word on estimating the parameter variance-covariance matrix. In each case, we can approximate the matrix Σ_β using the step length P_n evaluated at the ML estimator. That is to say:

$$\begin{aligned}
\hat{\Sigma}_{\beta,NR} &= - \left[\frac{\partial^2 \ell(\cdot)}{\partial \beta \partial \beta'} \right]^{-1} \Bigg|_{\beta_{ML}} \\
\hat{\Sigma}_{\beta,GN} &= - \left[\mathbb{E} \left(\frac{\partial^2 \ell(\cdot)}{\partial \beta \partial \beta'} \right) \right]^{-1} \Bigg|_{\beta_{ML}} \\
\hat{\Sigma}_{\beta,BHHH} &= \left[\sum_{i=1}^{N} \left(\frac{\partial \ell_i(\beta)}{\partial \beta} \right) \left(\frac{\partial \ell_i(\beta)}{\partial \beta} \right) \right]^{-1} \Bigg|_{\beta_{ML}}
\end{aligned}$$

Asymptotically, these approximations are equivalent, but do note that in small samples, they are likely to yield different estimates of the variance-covariance matrix.

2 Estimation with count data

The next assignment will involve estimating parameters from a count model using the aforementioned maximum likelihood estimation methods. Before diving into the estimation of a simple count model in MATLAB, let's set the stage with some motivation: what is a count model and why do we use it?

Count models have been developed in the econometric literature for two main reasons: accounting for discrete dependent variables and restricting the support of dependent variable outcomes to only non-negative values. Put more simply, count models are intrinsically useful when our dependent variable is a *non-negative integer*. Common examples include number of visits to the doctor or a recreational site, number of failures in production, number of live births, number of beers enjoyed on a night out, etc. To reiterate the usefulness of this class of models,

[2] Try deriving this yourself to check understanding.

take note that in each of these examples, the outcome *cannot be negative, and always takes on an integer value.*[3]

So the back to the Poisson distribution from above (section 1.3). Assuming a Poisson distribution on our dependent variable is the often "the starting point for count data analysis" (Cameron and Trivedi, 2005). As we've discussed, it's a discrete distribution with a single parameter (θ) that defines both the mean and variance of the distribution, while restricting all outcomes to be non-negative. These restrictions can be aesthetically preferable to other standard estimators. Most notably, the use of OLS estimation on count data assumes the true value is normally distributed around the mean and therefore can take on any real number value - regardless of whether it is negative on non-discrete. Thus, we could end up estimating models yielding negative counts - this is obviously undesirable!

With that said, OLS can often get us close and is much easier to estimate. So when are Poisson models especially preferable to OLS? If you run the code below, you'll see that OLS estimation (and its normality assumptions) starts to reasonably approximate the Poisson distribution at around $\theta = 10$, and does exceptionally well as the parameter value increases. So for lower values of θ, count models using distributional assumptions such as the Poisson are the wiser choice.

```
%% Compare poisson and normal distributions at different theta values
clear;
clc;

%% Theta = 1
n = -3:7;
theta = 1;
y = poisspdf(n,theta);
y2 = normpdf(n,theta,sqrt(theta));
plot(n,y,'o',n,y2,'x')
%%  Theta = 3
n = 0:10;
theta = 3;
y = poisspdf(n,theta);
y2 = normpdf(n,theta,sqrt(theta));
plot(n,y,'o',n,y2,'x')
%% Theta = 10
n = 0:20;
theta = 10;
y = poisspdf(n,theta);
y2 = normpdf(n,theta,sqrt(theta));
plot(n,y,'o',n,y2,'x')
%% Theta = 100
n = 0:200;
theta = 100;
y = poisspdf(n,theta);
y2 = normpdf(n,theta,sqrt(theta));
plot(n,y,'o',n,y2,'x')
```

2.1 *Replication in MATLAB*

To get better acquainted with the estimation of count models, now we're going to replicate some simple results from Cameron and Trivedi (2005). This example comes from chapter 20.3; their replication code is written for Stata[4], but we're going to execute the same procedure in MATLAB to get a better feel for the BHHH algorithm.

The data we're using comes from the RAND Health Insurance Experiment. We're interested in better understanding the *determinants* of the number of doctor visits used by patients studied in this sample. The data comes from around 8000 insurance enrollees in six sites across the country who were randomly enrolled in 1 of 14 insurance plans ranging from free health care

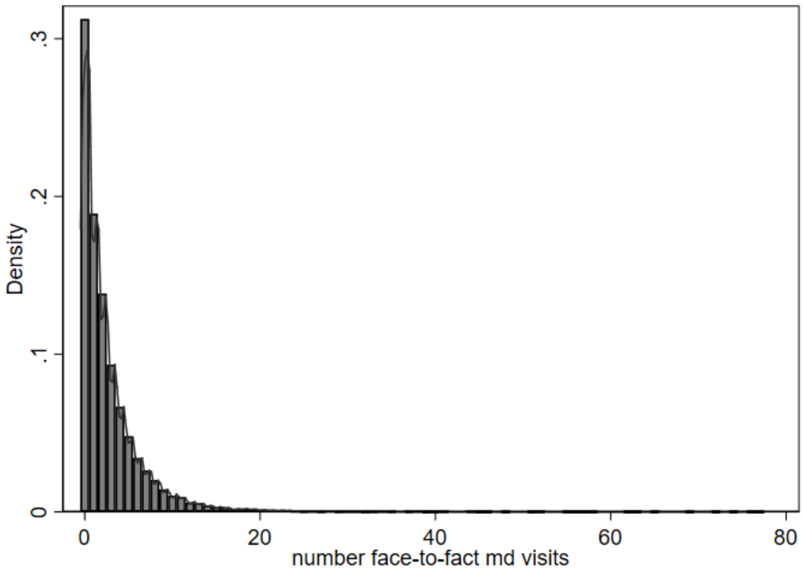
[3] Yes, I know what you're thinking. You're mistaken. One cannot consume half a beer - where's your Wisconsin pride?!

[4] You can acquire the .do file and the data file here.

to very high coinsurance. In addition to information on a patient's number of doctor visits and insurance plan, we also have information on some demographic and health characteristics.

Let's get a sense for what our dependent variable - a patient's count of doctor visits in the past year - looks like. After loading in the data, dropping missing observations, etc., we can create a frequency table for our count variable and plot it out as a histogram:

```
%%  Reading in and cleaning the data
[base_data,varnames,raw] = xlsread('newranddata');

var_need = {'mdvis','logc','idp','lpi','fmde','linc','lfam','xage','female',...
            'child','fchild','black','educdec','physlm','disea',...
            'hlthg','hlthf','hlthp'};

parnames = var_need(2:end);
parnames = ['constant' parnames]; % If we need a constant

z = pull_data(varnames,var_need,base_data);
z(any(isnan(z),2),:) = []; % drop rows with Na

depend = z(:,1);
rhsvar = z(:,2:end);
rhsvar = [ones(length(depend),1) rhsvar]; % add constant column

[numobs,numc] = size(rhsvar);

%% Frequency table and histogram:
table20_3 = tabulate(depend);
hist20_3 = histogram(depend);
```



We see clearly that there is a large mass at 0 - around 30% of observations - and a very long right tail, with some folks visiting the doctor up to around 80 times. The average number of visits is just below 3 - this falls well into the range of θ values where count models are preferable to OLS. Additionally, the high number of zero values and LONG tail to the right is an initial indicator of overdispersion - a potential problem that comes up with fair frequency when estimating count models with a Poisson MLE. We'll revisit this briefly, post-estimation.

In our model, we'll consider 18 covariates as explanatory variables for our visit count. Each of the covariates could plausibly affect demand for doctor's visits - our goal is measure the extent of their demand effects. These covariates are given in Table 20.4 of Cameron and Trivedi:

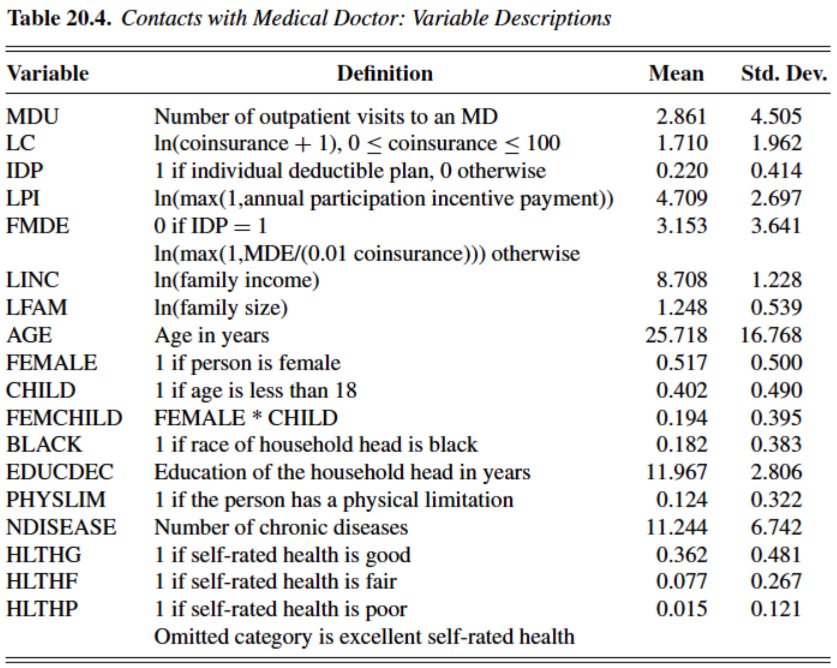


Table 20.4. *Contacts with Medical Doctor: Variable Descriptions*

Variable	Definition	Mean	Std. Dev.
MDU	Number of outpatient visits to an MD	2.861	4.505
LC	ln(coinsurance + 1), 0 ≤ coinsurance ≤ 100	1.710	1.962
IDP	1 if individual deductible plan, 0 otherwise	0.220	0.414
LPI	ln(max(1,annual participation incentive payment))	4.709	2.697
FMDE	0 if IDP = 1 ln(max(1,MDE/(0.01 coinsurance))) otherwise	3.153	3.641
LINC	ln(family income)	8.708	1.228
LFAM	ln(family size)	1.248	0.539
AGE	Age in years	25.718	16.768
FEMALE	1 if person is female	0.517	0.500
CHILD	1 if age is less than 18	0.402	0.490
FEMCHILD	FEMALE * CHILD	0.194	0.395
BLACK	1 if race of household head is black	0.182	0.383
EDUCDEC	Education of the household head in years	11.967	2.806
PHYSLIM	1 if the person has a physical limitation	0.124	0.322
NDISEASE	Number of chronic diseases	11.244	6.742
HLTHG	1 if self-rated health is good	0.362	0.481
HLTHF	1 if self-rated health is fair	0.077	0.267
HLTHP	1 if self-rated health is poor	0.015	0.121
	Omitted category is excellent self-rated health		

To estimate this model, we will maximize the log-likelihood function generated by our assumption of a Poisson distribution on the dependent variable and use of exponential mean parameterization. Recall, we defined this LLF above in Section 1.3 as:

$$\ell(\beta|X_i, y_i) = \sum_{i=1}^{n}[-e^{X_i'\beta} + y_i X_i\beta - \ln(y_i!)]$$

We can code this up in MATLAB as a simple function:

```
function  llf = lf_poisson(gamma)
    global rhsvar depend;
    llf = - exp(rhsvar*gamma) + depend.*(rhsvar*gamma) - log(factorial(depend));
    %llf = - exp(rhsvar*gamma) + depend.*(rhsvar*gamma) - gam_ln_bwg(depend + 1);
end
```

We run the BHHH algorithm, using starting values backed out of a simple OLS regression. This algorithm iterates over β values, using a step length defined as the inverse product of the LLF's Jacobian. Compared to our familiar NR and GN algorithms, note that this BHHH variation is reflected in changes of only a few lines of code (lines 19-36):

```
while ((crit > critic_limit) & (iter<= iter_limit));
   z = Grad(b0,func_name,numobs);  % Numerical Gradients
   H = z'*z;                       % Compute approx. to Hessian
   g = sum(z)';                    % Gradient vector (K x 1)
   db = inv(H)*g;                  % Compute full step adjustment
   if do_step == 1;                % Use a variable step length
    s = 1;                         % Reset base step length
    li1 = 0; li2 = 1;              % ** Intialize LLF's under 2 step lengths
    while (li1 < li2) && (s >=.2); % ** Loop to determine step length
       li1 = sum(feval(func_name,(b0 + s*db)));  %Eval. LLF at higher SL
       li2 = sum(feval(func_name,(b0 + s*db/2)));%Eval. LLF at lower SL
       s_star=s;
       s = s/2;
    end
```

```
      else
          s_star = s;
      end
      betas = b0 + s_star*db;                 % Update parameter values
```

We run the algorithm, and our maximum likelihood estimators are returned. Comparing with Table 20.5 in Cameron and Trivedi, we find that are estimates are indeed the same! What can we say about these results, and how do we interpret the coefficients? Let's start with the implied marginal effect of a change in some X variable:

$$y_i = e^{X_i'\beta} + \varepsilon_i \Rightarrow \frac{\partial}{\partial x_j}\mathbb{E}[y_i|X_i] = \beta_j e^{X_i'\beta}$$

As we can see, the marginal effect of a covariate depends on $e^{X_i\beta}$, which is individual-specific. This is less simple to interpret than in the OLS context. Furthermore, while the parameters alone do not say much about the marginal effects of a covariate, they can indicate relative strength of covariates:

$$\frac{\frac{\partial}{\partial x_j}\mathbb{E}[y_i|X_i]}{\frac{\partial}{\partial x_k}\mathbb{E}[y_i|X_i]} = \frac{\beta_j e^{X_i'\beta}}{\beta_k e^{X_i'\beta}} = \frac{\beta_j}{\beta_k}$$

Thus, assuming continuous covariates, if a parameter β_j is twice as large as β_k, the marginal effect of changing x_j will be twice as large as an identical-sized change in x_k. So in the context of our generated estimates, we can see that the marginal effect of $logc$ is about twice as large as the marginal effect of $fmde$.

Okay, let's wrap up with 2 final things: a bit of hypothesis testing, and then a very quick discussion of extensions in the genre of count data models. First let's test the overall fit of the model. As discussed in Greene (Ch. 21.9.1) - there is no direct analog to the R^2 measure when running a Poisson model. But we can construct our own measure built from residuals. Let's estimate the first of the two measures described by Greene[5]:

$$R_p^2 = 1 - \frac{\sum_{i=1}^{N}\left[\frac{y_i - \hat{\lambda}_i}{\sqrt{\hat{\lambda}_i}}\right]^2}{\sum_{i=1}^{n}\left[\frac{y_i - \bar{y}}{\sqrt{\bar{y}}}\right]^2}$$

```
%% Measure of model fit (Greene 21.9.1)
lambdahat = exp(rhsvar * ml_parm);
num = sum(((depend - lambdahat)./(lambdahat.^(0.5))).^2);
den = sum(((depend - mean(depend))./(mean(depend).^(0.5))).^2);
R2_p = 1 - num/den;
```

This measure, built from Pearson residuals, should fall between 0 and 1 - our value of around .16 suggests the model does have some degree of explanatory power compared to a model with only a constant.

Next, let's run a simple likelihood-ratio test to examine the null hypothesis that gender has no effect on the expected number of doctor visits. To do so, we run a restricted version of the model without the variables $female$ and $fchild$:

```
%% Likelihood ratio test on gender:
%   Estimate a restricted version of the model:
rhsvar(:,9) =[];
rhsvar(:,10) = [];
parnames(:,9) = [];
```

[5] You will have to code up the other one for Assignment 4.

```
parnames(:,10)  =[];
startvalue = inv(rhsvar'*rhsvar)*rhsvar'*log(depend+0.001);
[ml_rest,covb_rest] = max_bhhh(startvalue,parnames);
res_llf = sum(feval(func_name,ml_rest));

LR = 2*(unres_llf - res_llf);
crit_val = chi_bwg(0.05,2,LR);
```

Our LR test suggests we should reject the null, meaning we have fairly strong evidence that these dummy variables belong in the model.

Finally, let's estimate our model and see how the fitted values compare to the true outcomes:

```
%% Predicted values
pred_poisson = exp(rhsvar*ml_parm); % Predicted expectation
prob_matrix = zeros(numobs,10); %(numobs,11) if we want to include >9

% First element: P[y=0] = f(y = 0| theta)
prob_matrix(:,1) = exp(-pred_poisson);

% We will use Panjer recursion here
% P[y=k] = P[y=k-1]*theta/k for k>=1
for i = 2:10
   prob_matrix(:,i) = prob_matrix(:,i-1).*(pred_poisson/(i-1));
end

% All values > 9
%prob_matrix(:,11) = 1 - sum(prob_matrix, 2);

% Replication of Rows 1&2, Table 20.6 in Cameron and Trivedi
prob_frequency = [table20_3(1:10, 3)'; 100*mean(prob_matrix, 1)]
```

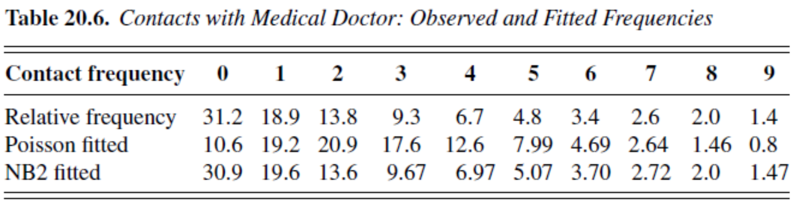


Table 20.6. *Contacts with Medical Doctor: Observed and Fitted Frequencies*

Contact frequency	0	1	2	3	4	5	6	7	8	9
Relative frequency	31.2	18.9	13.8	9.3	6.7	4.8	3.4	2.6	2.0	1.4
Poisson fitted	10.6	19.2	20.9	17.6	12.6	7.99	4.69	2.64	1.46	0.8
NB2 fitted	30.9	19.6	13.6	9.67	6.97	5.07	3.70	2.72	2.0	1.47

We see that our Poisson regression does a fairly poor job of fitting this data. Using the fitted Poisson model, we seriously underpredict the likelihood of zero visits and overpredict the number of visits falling in the 1-7 range. This overdispersion phenomenon is born directly from our initial assumption on the distribution of the the dependent variable. Because we chose a Poisson distribution, we assumed that the variance of our dependent variable is equal to its mean value. In reality, the variance of our doctor visit count is *much, much larger* than its mean - one need only to look at summary statistics in Table 20.4 for proof.

To avoid getting too far into the weeds of count models - and since Brian's cooking up a question on this for the next assignment - we won't code any further on this today. But keep this in mind when working with count data in the future. All of the major econometric reference books do a decent job laying out ways around this problem; see Chapters 20.4 and 20.7 of Cameron and Trivedi, Chapters 21.9.2-3 of Greene, or Chapters 19.3-4 of Wooldridge. The standard improvement is to assume a negative binomial (NB) distribution on the dependent variable, which relaxes the mean-variance equivalence that exists under a Poisson distribution. This model nests the Poisson model, and so when estimating an NB model, one can formally test whether the data actually suggests a Poisson distribution. Row 3 of Table 20.6 above shows the fitted estimates of the same model we estimated under the negative binomial assumption - as can be seen, performance is greatly improved.