**Likert Scales and Multinomial Random Variables Updated 4/23/14 (See p.7)**

In this lecture we consider a generic random variable, *X*, with sample space . Denote its *pdf* as:

. (1)

Then for *iid* associated data collection variables , we have

. (2)

We now define the random variables , where , and where, here, the generic random variable is the *indicator function* associated with the value *x.* In other words, it entails two possible events: (i) , and (ii) . Clearly then, . From this observation it follows that .

We now address the joint *pdf* of . To begin, notice that ”*The act of recording how many of the elements of*  *take on the value x*”. Clearly, we have the condition . It should be clear from this condition that the elements of  are *not* mutually independent. With a nontrivial amount of work [c.f. <http://en.wikipedia.org/wiki/Multinomial_theorem> ], it can be shown that the *pdf* of  is:

. (3).

The sample space for  is 

The distribution (3) is called the ***multinomial distribution***. [c.f. <http://en.wikipedia.org/wiki/Multinomial_distribution> ]

***The mean for* Y:**

Since each , we immediately have

 (4a)

***The covariance for*** **Y:**

Again, since each , we immediately have

. (4b)

The quantity is a bit ‘trickier’, since, as noted above, these random variables are *correlated.* With some work, it can be shown that

. (4c)

To obtain the expression for , note that the off-diagonal elements of this  matrix are exactly the off-diagonal elements of the matrix . The *x*th diagonal of this latter matrix is ; whereas the *x*th diagonal of the matrix  is (4b). We can write this element as: .

Hence, if we define the  matrix, we arrive at:

. (5)

Equation (5) is a concise expression. Furthermore, it highlights the fact that if any of the probabilities  are equal to zero, then the inverse of (5) does not exist. This is important in the contest of the central limit theorem (*CLT*), which states that for sufficiently large sample size, *n*, . Since the *pdf* of this estimator involves the inverse of (5), one must then acknowledge the need to take a pseudo-inverse of (5).

From (4b) and (4c), we can obtain the correlation coefficient

. (6)

***Connection to the Central Limit Theorem* (*CLT*):**

Once, again, recall that, marginally, for each , we have . Hence,

. (7)

Hence, from the *CLT,* we know that for sufficiently large *n*:

 where  (8)

**Example 1**. Often times a survey question answer will include the three options: {*disagree, no opinion, agree*} . In a survey of *n* persons, we will let *Y*1 = “The act of recording the number of persons who *disagree*, *Y*2 = “The act of recording the number of persons who *agree,* and *Y*3 = “The act of recording the number of persons who *have no opinion.*

***The special case where p*1 = *p*2 :**

We chose this case to address a situation that often occurs; namely where it appears that the population is evenly split on a given issue. For notational convenience, let . This will be our *null hypothesis*, *H*0. It follows that:

 (9a)

, or

. (9b)

Now recall that the 3-D random variable  can be written as . And so for convenience in the following, we will *redefine* . We then have

 and . (10)

Letting , it follows that, under :

 and . (11)

Now, consider the test statistic . Clearly, , which, under is zero. To compute the variance of , recall that

.

Specifically, under  we have:

. (12)

Finally, for sufficiently large *n*, we obtain, under :

. (13)

Hence, for this situation hypothesis testing becomes *very* straightforward. ☺

***Numerical Values***: Suppose now that our null hypothesis is , and that our alternative hypothesis is

. Let the *false alarm* probability be , and the sample size be . Then under :

. Equivalently, . For  our threshold is . Then we will announce if .

**Remark** Recall that when conducting the above test in the situation where  and  are *independent,* the variance of  is: . In the present case, however, we have

. (14)

This relatively high level of correlation will increase the standard deviation of  by a factor of .

***Simulations*:** For related ways (e.g. using *Excel*) to simulate a multinomial random variable, see <http://en.wikipedia.org/wiki/Multinomial_distribution> . Here, we will use the *Matlab* code ‘Likert1D\_3.m’ given in the Appendix.

For 103 simulations of the code, we obtain:

**mu\_phat** = 0.3980 0.2013 0.4008

**cov\_phat** =

0.0048 -0.0015 -0.0034

-0.0015 0.0032 -0.0017

-0.0034 -0.0017 0.0051

and

**corrcoef(phat)**=

1.0000 -0.3721 -0.6810

-0.3721 1.0000 -0.4263

-0.6810 -0.4263 1.0000

Notice that the estimated correlation coefficient value -0.6810 is close to the theoretical value -0.67 in (13). □

**Two-Dimensional Multinomial Random Variables**

**Example 2.** When negotiating with potential clients, negotiations often focus on cost. It is relatively easy to measure a potential client’s view of the cost of a project. In the case of building architecture, a major factor in cost is the longevity of the design.

Suppose that you work for an architecture firm, and that you are considering carrying out a survey that would allow you to predict a potential client’s perceived longevity of a design, based on the cost. Prior to conducting the survey, you decide to run some simulations under various scenarios, in order to determine the survey sample size that would be needed to obtain trustworthy predictions.

Define the random variables:

*“X = The act of recording a client’s perceived cost of a design”* with events [*X=*1]=”cheap”, [*X=*2]= “moderate”, and [*X=*3]=”expensive”.

and

*“Y = The act of recording the longevity of a design”* with events [*Y=*1]=”short”, [*X=*2]= “medium”, and [*X=*3]=”long”.

The sample space for  is . This sample space is discrete. It includes 9 elements, with joint probabilities .

Now, in order to construct a given simulation you could specify the numerical values of 8 of these 9 joint probabilities. However, it is probably more natural to specify 8 other probabilities that are related to these. In this example, we will specify 6 conditional probabilities and 2 marginal probabilities.

Consider the matrix of conditional probabilities:

 where . (15)

Even though (15) includes 9 parameters, we only need to specify 6 of them. This is because

. (16)

Recall that

. (17)

Define the joint probability matrix

. (18)

To see the relationship between (16) and (18), define the probability matrix

. (19)

Note that even though (16) includes 3 parameters, there are only two that need to be specified. And so, there are a total of 8 parameters that need to be specified in relation to (15) and (18).

Having specified the 8 parameters, one could then program the simulations to use them directly. Alternatively, one could find the corresponding joint probabilities via the relation

 ;  (20)

and write a program that generates simulations of  using these joint probabilities. In this example we will use this approach. The *Matlab* code written for this purpose is

For 104 simulations this code gave the following estimate of :

mu\_Pygxhat =

0.4993 0.4011 0.0996

0.2008 0.5004 0.2988

0.0994 0.3006 0.6000

It also gave the true value of :

Pygx =

0.5000 0.4000 0.1000

0.2000 0.5000 0.3000

0.1000 0.3000 0.6000

The closeness of the above matrices suggests that the code is correct ☺.

The scatter plot below shows the relation between the estimators  and .



**Converting a Bivariate Likert Random Variable to a Univariate Random Variable**

The manner in which the code in the Appendix computed the conditional probabilities entailed a mapping of the 9 elements in  to the numbers 1 through 9. This suggests defining the random variable, call it  **W**, with a Likert scale having the array of singleton probabilities: , and such that . Marginally, . Clearly, **W** is a univariate Likert random variable. And so we can take advantage of the properties of the multinomial *pdf* for the 1-9 scale variable. In particular, we now have

 and  where . (21)

Hence, we are now in a position to carry out a variety of analyses, such as those addressed in ***Example* 1.** It should be noted, however, that we no longer have the natural 2-D format needed to construct a linear model. But we still can proceed as in the bivariate case by using *equivalent events.* ☺

For example, consider the equivalent events :

1. 
2. 

Define . Then .

Hence, we have:  with  and  (22)

Clearly, the matrix in (22) is unique to . For this reason, it is better denoted as . Because we now have closed form expressions, (22), for the mean and covariance of , simulations are no longer needed.

**Example 2 continued.**

***Theoretical Quantities*:**

Pxy =

0.1000 0.0800 0.0200

0.1000 0.2500 0.1500

0.0300 0.1200 0.1500

fxy' = 0.1000 0.0800 0.0200 0.1000 0.2500 0.1500 0.0300 0.1200 0.1500

muW' = 5.0000 4.0000 1.0000 5.0000 12.5000 7.5000 1.5000 4.5000 9.000

CovW =

4.5000 -0.4000 -0.1000 -0.5000 -1.2500 -0.7500 -0.1500 -0.4500 -0.9000

-0.4000 3.6800 -0.0800 -0.4000 -1.0000 -0.6000 -0.1200 -0.3600 -0.7200

-0.1000 -0.0800 0.9800 -0.1000 -0.2500 -0.1500 -0.0300 -0.0900 -0.1800

-0.5000 -0.4000 -0.1000 4.5000 -1.2500 -0.7500 -0.1500 -0.4500 -0.9000

-1.2500 -1.0000 -0.2500 -1.2500 9.3750 -1.8750 -0.3750 -1.1250 -2.2500

-0.7500 -0.6000 -0.1500 -0.7500 -1.8750 6.3750 -0.2250 -0.6750 -1.3500

-0.1500 -0.1200 -0.0300 -0.1500 -0.3750 -0.2250 1.4550 -0.1350 -0.2700

-0.4500 -0.3600 -0.0900 -0.4500 -1.1250 -0.6750 -0.1350 4.0950 -0.8100

-0.9000 -0.7200 -0.1800 -0.9000 -2.2500 -1.3500 -0.2700 -0.8100 7.3800

muX1Y2'= 10.0000 21.0000

CovX1Y2 =

8.0000 -0.2000

-0.2000 12.1800

Running 30,000 simulations for N=50 sample size gives (‘likertsim.m’):

muX1Y2 = 10 21

muhatX1Y2 = 9.9960 21.0194

CovX1Y2 =

8.0000 -0.2000

-0.2000 12.1800

CovhatX1Y2 =

7.9915 -0.2241

-0.2241 12.1827

Remark. Even for this simulation size the covariance between X1 and Y2 is notably variable!

**Appendix *Matlab* Codes**

% PROGRAM NAME: Likert1D\_3.m

% Run m simulations of a 1-D multinomial random variable

% with S = {1,2,3} for sample size n.

%--------------------------------------------

k = 3; %This is so the code can be more easily modified for k>3

% SPECIFY PROBABILITIES:

p(1) = .4; p(2) = .2;

% Compute CDF:

F = zeros(1,k-1);

F(1) = p(1);

for j = 2:k-1

F(j) = F(j-1) + p(j);

end

% SPECIFY SAMLE SIZE:

N = 50;

%--------------------------------------------

% GENERATE m SIMULATIONS, each of size n:

NSIM = 10^3;

phat = zeros(NSIM,k); %Initialize estimated proportions

Y = k\*ones(NSIM,k); %Initialize all counts to the value k

%NOTE: The array Y is only needed to illustrate the raw survey results

% If only phat is of concern, then better to not include Y, as it

% can take a significant amount of memory.

for m = 1:NSIM

bc = zeros(1,k-1); %Initialze the three bin counts

for n = 1:N

u = rand(1,1);

if u < F(1)

Y(n,m) = 1; bc(1) = bc(1) + 1;

else if u < F(2)

Y(n,m) = 2; bc(2) = bc(2) + 1;

end

end

end

phat(m,:) = [bc(1)/N , bc(2)/N , 1-(bc(1)+bc(2))/N ];

end

%============================================

% COMPUTE DISTRIBUTIONAL INFORMATION FOR phat:

mu\_phat = mean(phat)

cov\_phat = cov(phat)

% PROGRAM NAME: xyLikert.m

% This code simulates n measurements of (X,Y)

% where Sx = Sy = {1,2,3} = {low, medium, high}

% X = cost of a design

% Y = longevity of the design

%============================

clear all

% Specify matrix of conditional probabilities of Y=y|X=x:

Pygx = [ .5 .4 .1; .2 .5 .3; .1 .3 .6]; %Rows must sum to 1.0

% Specify marginal probabilities for X:

px = [.2 .5 .3];

Px = [px(1)\*ones(1,3);px(2)\*ones(1,3);px(3)\*ones(1,3)];

% Compute joint probabiility matrix for (X,Y):

Pxy = Pygx.\*Px; % This is the 3x3 matrix of joint probabilities

%================================================

% SIMULATIONS:

% In order to simulate, the Uniform random number generator

% will be used.The followng segment of code generates

% the breakpoints of the 9 intervals that the interval [0 , 1]

% will use:

fxy = reshape(Pxy',9,1); %concantonate the rows of Pxy

Fxy = zeros(9,1);

Fxy(1) = fxy(1);

for k = 2:9

Fxy(k) = Fxy(k-1) + fxy(k); % F(k) is the RIGHT end of the kth interval

end

%----------------------

%----------------------

nsim = 10^4; %Number of simulations

N = 10^2; %Sample size for each simulation

pygxhat\_array = zeros(nsim,9); %Array of pygx estimates

for nn = 1:nsim

Bc = zeros(1,9); %Initialise the counts of the 9 bins

for n = 1:N

[q,kbin] = max(ceil(Fxy - rand(1,1)));

Bc(kbin) = Bc(kbin) +1 ;

end

fxyhat = Bc/N;

pxhat = [sum(fxyhat(1:3)) , sum(fxyhat(4:6)) , sum(fxyhat(7:9)) ];

pxhatvec = [pxhat(1)\*ones(1,3),pxhat(2)\*ones(1,3),pxhat(3)\*ones(1,3)];

pygxhat\_array(nn,:) = fxyhat./pxhatvec;

end

mu\_pygxhat = mean(pygxhat\_array);

mu\_Pygxhat = reshape(mu\_pygxhat,3,3)'

Pygx

pause

%======================================

% Investigation of the joint properties of 2 estimators

p1g1 = pygxhat\_array(:,1); % estimates of p1|1

p2g1 = pygxhat\_array(:,2); % estimates of p2|1

figure(1)

plot(p1g1,p2g1,'\*')

grid

xlabel('p1|1\_hat')

ylabel('p2|1\_hat')

title('Scatter Plot of p1|1\_hat (x) vs. p2|1\_hat (y)')