

## Snoring and Heart Disease (Agresti 4.2.2)

Ronald A Thisted

2 February 1999

# Analysis 1

## Snoring Example

This is an example from Section 4.2.2 of Agresti, ICDA (p. 75), studying the effects of snoring frequency on the occurrence of heart disease.

In this example, we enter the data into Stata directly. We name the columns indicating presence and absence of heart disease `hd0` (for no heart disease) and `hd1` (for heart disease present). The reason for using a common “stub” (`hd`) followed by 0 or 1 to indicate presence or absence is that it will make it easier to convert between long and wide forms of the data set.

the “scores” that are used for the different degrees of snoring are those assigned in the book. See Agresti for a discussion of how and why to assign numerical scores to categories such as this one.

```
[1.1] . input snore hd1 hd0

           snore      hd1      hd0
1. 0 24 1355
2. 2 35 603
3. 4 21 192
4. 5 30 224
5. end
```

Once the data are entered, we will need a denominator column for any sort of binomial regression.

```
[1.2] . generate n = hd0+hd1
```

```
[1.3] . list
```

```
           snore      hd1      hd0      n
1.          0         24      1355     1379
2.          2         35      603     638
```

3.	4	21	192	213
4.	5	30	224	254

Note in the listing above how much wasted space there is. If we want to have Stata list many columns in a data listing, we need to use fewer spaces for each variable. The `format` command in Stata allows us to assign the amount of space on the page that will be used to display each variable. The next command assigns the same amount of space – five columns in all, with no digits following a decimal place – to each of the variables starting with `snore` and ending with `n`. Note the compression in the listing that follows.

```
[1.4] . format snore-n %5.0f
```

```
[1.5] . list
```

	snore	hd1	hd0	n
1.	0	24	1355	1379
2.	2	35	603	638
3.	4	21	192	213
4.	5	30	224	254

The fourth column of Table 4.1 contains the observed proportion. Here is how we would calculate, and then list, these values.

```
[1.6] . generate p = hd1/n
```

```
[1.7] . format p %6.3f
```

```
[1.8] . list
```

	snore	hd1	hd0	n	p
1.	0	24	1355	1379	0.017
2.	2	35	603	638	0.055
3.	4	21	192	213	0.099
4.	5	30	224	254	0.118

The *linear probability model* is a generalized linear model for binomial outcomes (the random part), which links the mean ( $\mu = \pi/n$ ) to the linear predictor ( $\alpha + \beta x$ ) by an identity function, as follows:

$$\pi = \alpha + \beta x$$

Notice that `hd1` is the outcome variable – the number of individuals with heart disease, `snore` is the predictor variable, the level of snoring, and `n` is the denominator (that is, the total number of individuals) at each level of snoring.

```
[1.9]
```

```
. glm hd1 snore, family(binomial n) link(identity)
```

```
Iteration 1 : deviance = 213.6197
Iteration 2 : deviance = 0.0696
Iteration 3 : deviance = 0.0692
Iteration 4 : deviance = 0.0692
```

```
Residual df = 2 No. of obs = 4
Pearson X2 = .0688373 Deviance = .0691912
Dispersion = .0344186 Dispersion = .0345956
```

```
Binomial (N=n) distribution, identity link
```

hd1	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
snore	.0197778	.0028049	7.051	0.000	.0142804	.0252752
_cons	.0172467	.003451	4.998	0.000	.0104828	.0240105

The estimates for the generalized linear model are  $\hat{\alpha} = 0.0172$  and  $\hat{\beta} = 0.0198$ . We can use these to get fitted values of  $\mu$ , and hence, of  $\pi$ . Stata can calculate these for us after we fit a generalized linear model. The `glmpr` command creates a new variable (`linfit` in this case) consisting of the estimated mean ( $\mu$ ) for each case.

```
[1.10] . glmpr linfit
```

```
[1.11] . list
```

```
      snore   hd1   hd0      n      p   linfit
1.       0    24  1355  1379  0.017  23.78317
2.       2    35   603   638  0.055  36.23987
3.       4    21   192   213  0.099  20.52424
4.       5    30   224   254  0.118  29.49848
```

```
[1.12] . gen phatlin = linfit/n
```

```
[1.13] . format phatlin %6.3f
```

```
[1.14] . list snore-p phatlin
```

```
      snore   hd1   hd0      n      p  phatlin
1.       0    24  1355  1379  0.017  0.017
2.       2    35   603   638  0.055  0.057
3.       4    21   192   213  0.099  0.096
4.       5    30   224   254  0.118  0.116
```

The logistic model is similar to the linear probability model, but it uses a logistic link function:

$$\log\left(\frac{\pi}{1-\pi}\right) = \alpha + \beta x$$

[1.15] . glm hd1 snore, family(binomial n) link(logit)

```
Iteration 1 : deviance =    2.8261
Iteration 2 : deviance =    2.8089
Iteration 3 : deviance =    2.8089
```

```
Residual df =          2                No. of obs =          4
Pearson X2  =  2.874074                Deviance   =  2.808912
Dispersion =  1.437037                Dispersion =  1.404456
```

Binomial (N=n) distribution, logit link

hd1	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
snore	.3973366	.0500086	7.945	0.000	.2993215	.4953518
_cons	-3.866248	.1662036	-23.262	0.000	-4.192001	-3.540495

Consequently, we estimate the log odds of heart disease as 0.397 x snore - 3.866. This can be converted to odds of heart disease by exponentiating, and then to the estimated probability of heart disease by the relationship  $\pi = \text{odds}/(1+\text{odds})$ . Once again, Stata can do this for us.

[1.16] . glmprcd logitfit

[1.17] . gen phatltgt = logitfit/n

Finally, the probit model is similar to the previous two models, but it uses a probit link function:

$$\Phi^{-1}(\pi) = \alpha + \beta x$$

Here,  $\Phi^{-1}$  is the inverse of the standard normal cumulative probability function.

[1.18] . glm hd1 snore, family(binomial n) link(probit)

```
Iteration 1 : deviance =    1.8735
Iteration 2 : deviance =    1.8716
Iteration 3 : deviance =    1.8716
```

```
Residual df =          2                No. of obs =          4
Pearson X2  =  1.910081                Deviance   =  1.871561
Dispersion =  .9550406                Dispersion =  .9357803
```

Binomial (N=n) distribution, probit link

hd1	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
snore	.1877704	.0234804	7.997	0.000	.1417496	.2337912
_cons	-2.060552	.0701661	-29.367	0.000	-2.198075	-1.923028

[1.19] . glmprcd prbitfit

[1.20] . gen phatpbt = prbitfit/n

Here is the display of fitted values corresponding to Table 4.1. The option `nodisplay` in the list command keeps as many columns as possible on a single line.

[1.21] . format phatlgt phatpbt %6.3f

[1.22] . list snore-p phatlin phatlgt phatpbt, nodisplay

	snore	hd1	hd0	n	p	phatlin	phatlgt	phatpbt
1.	0	24	1355	1379	0.017	0.017	0.021	0.020
2.	2	35	603	638	0.055	0.057	0.044	0.046
3.	4	21	192	213	0.099	0.096	0.093	0.095
4.	5	30	224	254	0.118	0.116	0.132	0.131

Agresti distinguishes fitting the linear probability model by maximum likelihood (which is what `glm` does in Stata) from fitting the model using ordinary least squares (OLS). To fit the model using OLS, we would use an ordinary `regress` command in Stata with  $Y = 0$  or  $1$ , depending on whether each individual had heart disease or not. Thus, we would need to have a data set with 2484 rows. Now there are only eight combinations of  $Y$  (heart disease) and  $X$  (snoring status), so we can combine them into just eight rows, with a variable indicating the frequency. This corresponds exactly to the *long form* of our data set, where we need to convert the `hd` variables from wide to long form.

Here is how it is done in Stata. There are a few things to note. The variable being reshaped is heart disease, represented in the current data set by `hd0` and `hd1`. In the reshape command, we tell Stata how to recognize the common “stub” that defines these variables (`hd`). The information contained in the rest of the variable name (0 or 1 in this case, and represented by the `@` sign in the `reshape` command) will be put into the `j()` variable you specify.

[1.23] . reshape long hd@, i(snore) j(Y)  
(note: j = 0 1)

```

Data
-----
Number of obs.          4  ->  8
Number of variables     11 ->  11
j variable (2 values)   ->  Y
xij variables:
                        hd0 hd1 ->  hd
-----

```

```
[1.24] . list snore Y n hd
```

```

      snore      Y      n      hd
1.      0      0  1379  1355
2.      0      1  1379   24
3.      2      0   638   603
4.      2      1   638   35
5.      4      0   213   192
6.      4      1   213   21
7.      5      0   254   224
8.      5      1   254   30

```

```
[1.25] . regress Y snore [fw=hd]
```

```

Source |      SS      df      MS
-----+-----
Model | 3.07633377      1 3.07633377
Residual | 102.052491 2482 .041117039
-----+-----
Total | 105.128824 2483 .042339438

Number of obs =    2484
F( 1, 2482) =    74.82
Prob > F      =    0.0000
R-squared     =    0.0293
Adj R-squared =    0.0289
Root MSE     =    .20277

-----
      Y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-----+-----
snore |   .020038   .0023166     8.650   0.000   .0154954   .0245806
_cons |   .0168723  .0051571     3.272   0.001   .0067598   .0269849
-----

```

Notice that the OLS estimates for  $\alpha$  and  $\beta$  are slightly different from the ML estimates that we obtained above. This is a consequence of the fitting process, and also of the fact that the OLS procedure assumes that the variance of the outcome is constant, that is, doesn't depend on the predictor variable `snore`. Actually, since these are binomial outcomes, the variance **does** change as the

mean changes. Maximum likelihood takes this into account, whereas OLS does not.

As a final note, suppose that we had fit a logistic model to the 2484 individual outcomes, rather than to the collapsed outcomes we used originally. This can easily be done with the data in long form, as they are (now) in this data set. Notice that the estimates and their standard errors are exactly the same as before. The only things that have changed are the degrees of freedom, the deviance, and the other statistics reported at the top. We shall discuss these difference later in class.

```
[1.26] . glm Y snore [fw=hd], fam(binomial 1)
```

```
Iteration 1 : deviance = 978.3730
Iteration 2 : deviance = 856.4720
Iteration 3 : deviance = 838.5820
Iteration 4 : deviance = 837.7346
Iteration 5 : deviance = 837.7316
Iteration 6 : deviance = 837.7316
```

```
Residual df =      2482                No. of obs =      2484
Pearson X2  =  2412.818                Deviance   =  837.7316
Dispersion =  .9721267                Dispersion =  .3375228
```

```
Bernoulli distribution, logit link
```

```
-----
      Y |      Coef.  Std. Err.      z    P>|z|    [95% Conf. Interval]
-----+-----
  snore |   .3973366   .0500103    7.945   0.000    .2993183   .4953549
  _cons |  -3.866248   .166212   -23.261   0.000   -4.192018  -3.540479
-----
```