Lab Guide for Modeling Categorical Outcomes | 2018

Initial draft: Please excuse typos, fragments, and unclear writing. Suggestions/corrections appreciated.

Scott Long – 2018-06-01

mco18 lab guide 2018-06-01.docx

Contents

Part I. Introduction to the lab guide

The lab guide helps you understand the commands from lecture. The guide focuses on the mechanics of Stata rather than specification and interpretation.

1. You will be receive a handout about setting up Stata on your computer. This will explain how to install Stata packages and download files. The **`mcolab-*.do`** files are simplified versions of the **`mcolec-*.do`** files used in lecture. Datasets are downloaded, including those used in lecture and some larger datasets that have PDF files with documentation.
2. If you are new to Stata, look at the *cda2014 StataGettingStarted 2014-06-04color.pdf* in your working directory and/or google youtube stata and watch videos from StataCorp.
3. Part II talks about do-files. Since all of our work uses do-files, make sure you understand this section.
4. Part III is divided by lecture topics. Tools for each lecture are explained using a **`mcolab`** do-files that is based on one of the lecture do-files. If a subheading includes a #, this refers a locations in the do-file. After you read a section, use either the **`mcolab*.do`** or **`mcolec*.do`** files as templates to explore the commands.
5. Stata commands and output are in this **`font`**. Commands in the output are preceded by "**`.`**" and "**`>`**" which are not part of the command.

Part II: Do-files

Do-files are simply text files with commands. Other programs refer to these as script files, command files, or syntax files. Using do-files is essential for reproducing your work. Commands that are entered from the Command window or with dialog boxes are very hard to reproduce later. I do all of my work using do-files.

Do-files need to be *robust* so they produce the same results when run later. They also need to be *legible* so they are easy to understand. To make do-files robust and legible, I suggest that you follow the template below (see **`mcolab-template-dofile.do`**).

```
01     capture log close
02     log using jslong-mcolab-lrm-2018-05-21, replace text

03     version 14.1
04     clear all
05     macro drop _all
06     set linesize 80
07     set scheme s1manual

08     //  LRM lab exercise
09     //  MCO workshop 2018
10     local pgm jslong-mcolab-lrm
11     local who Scott Long
12     local 2018-05-21
13     local tag `pgm'.do `who' `dte' // do not change

14     //  #1 load data
       {commands here}

15     //  #2 descriptive statistics
       {commands here}

16     //  #3 fit model
       {commands here}

17     log close
18     exit
```

Lines 1-2 open a log file where output from your commands is written. Line 1 ensures no log file is already open. Line 2 creates a new log file whose name matches the name of the do-file. Line 17 closes the log file so additional results are not saved to the file. If you do not add a return after line 17, line 17 does not run. Line 18 makes sure that line 17 runs and tells Stata to ignore any later lines. You can put ideas, notes, etc. after line 19.

Lines 3-7 erase things in memory so that your results are not dependent on something you did before running the do-file. The scheme makes graphs look the same on different computer.

Line 8 documents what the do-file is for, while line 9 indicates the project. Lines 10-12 document what, who and when produced the results. This helps when looking at the output and is very useful when creating variables and saving new datasets. Always update this information in a new do-file. Line 13 creates a tag that is explained in the section on the LRM.

Lines that start with `//` are comments. They do not run Stata commands, but simply let you add notes. You can comment out single lines of text with an asterisk (`*`) or a slash (`//`), or create blocks of comments starting with a `/*` and ending with `*/`.

Commands for analysis begin on line 14. To write legible do-files, organize the content to make it easier to locate later. Group related commands (e.g., creating demographic variables, estimating nested models for one outcome) keep the file orderly.

Part III: Examples to explore during lab

These examples highlight how the Stata code works. Each **mcolab** do-file corresponds to a more elaborate **mcolec** do-file. Use the lab file to explore the commands. You will learn the most, I think, if you experiment with the commands. Try new options and see what happens. Think of something you'd like to know and figure out the necessary commands.

You do not need to work through each example or all of the steps from an example. Spend your time on the things that are most useful to you.

Continuous Outcomes: mcolab-lrm-wages.do

Provenance tag

Locals are useful for documenting your do-file:

```
. local pgm mcolab-lrm-wages
. local dte 2018-05-21
. local who Scott Long
```

They can be combined to create a provenance tag used in documentations:

```
. local tag "`pgm'.do `who' `dte'"
. di "tag: `tag'"
tag: mcolab-lrm-wages.do Scott Long 2018-05-21
```

Graph format

PNG files do not print well. EMF works well in Win, but can't be created in Mac. PDF works best in MAC. Here's how to write a do-file that works in Mac or Win. This code, which you do not need to understand, figures out what your OS is choses the appropriate type of file:

```
. local graphfmt emf //
. if "`c(os)'"!="Windows" local graphfmt pdf
```

#1 Load and check the variables

The codebook command is handy since it shows the variable labels:

```
. use slid-ontario01, clear
(Canada's 1994 Survey of Labor and Income Dynamics \ 2011-04-04)

. codebook wages male age edyears, compact

Variable    Obs Unique      Mean  Min    Max  Label
-------------------------------------------------------------------------------
wages      3997   1531  15.54459  2.3  49.92  Composite hourly wage rate in d...
male       3997      2  .4978734    0      1  Is male?
age        3997     50  36.95822   16     65  age in years
edyears    3997     21  13.21191    0     20  years of education completed
-------------------------------------------------------------------------------
```

#2 Wages on fem age edyears

I use syntax notation to indicate whether a variable is continuous **c.** or categorical **i.** This is very important when using **margins** to compute effects. While Stata has rules for when you need to indicate **i.** or **c.**, the easiest things is to always use this notation when specifying regressors.

```
. regress wages i.male c.age c.edyears

      Source |       SS           df       MS      Number of obs   =     3,997
-------------+----------------------------------   F(3, 3993)      =    590.67
       Model |  75828.1741         3   25276.058   Prob > F        =    0.0000
```

```
    Residual |  170869.757     3,993  42.7923258   R-squared       =    0.3074
-------------+----------------------------------   Adj R-squared   =    0.3069
       Total |  246697.931     3,996   61.736219   Root MSE        =    6.5416

------------------------------------------------------------------------------
       wages |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        male |    3.47367   .2070092    16.78   0.000     3.067817    3.879524
         age |   .2612932    .008664    30.16   0.000      .244307    .2782794
     edyears |   .9296491   .0342567    27.14   0.000     .8624868    .9968115
       _cons |  -8.124231   .5989773   -13.56   0.000    -9.298561   -6.949902
------------------------------------------------------------------------------
```

#3 Comparing predictions from mtable and margins

`margins` makes predictions. It is an amazing command but the output is not compact:

```
. margins, atmeans at(age=(25(20)65) male=(0 1) edyears=20)

Adjusted predictions                            Number of obs     =      3,997
Model VCE    : OLS

Expression   : Linear prediction, predict()

1._at        : male            =           0
               age             =          25
               edyears         =          20

2._at        : male            =           0
               age             =          45
               edyears         =          20

3._at        : male            =           0
               age             =          65
               edyears         =          20

4._at        : male            =           1
               age             =          25
               edyears         =          20

5._at        : male            =           1
               age             =          45
               edyears         =          20

6._at        : male            =           1
               age             =          65
               edyears         =          20

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   17.00108   .2830003    60.07   0.000     16.44624    17.55592
          2  |   22.22695   .2873246    77.36   0.000     21.66363    22.79026
          3  |   27.45281   .3808846    72.08   0.000     26.70606    28.19956
          4  |   20.47475   .2878136    71.14   0.000     19.91048    21.03903
          5  |   25.70062    .290798    88.38   0.000     25.13049    26.27074
          6  |   30.92648   .3825464    80.84   0.000     30.17648    31.67649
------------------------------------------------------------------------------
```

The **`mtable`** command calls **`margins`** and presents the results more clearly. Here **`mtable`** does not show all of the statistics from **`margins`**, but you can use options if you want them (**`help mtable`** for details):

```
. mtable,  atmeans at(age=(25(20)65) male=(0 1) edyears=20)

Expression: Linear prediction, predict()
```

```
           |     male       age        xb
 ----------+------------------------------
         1 |        0        25    17.001
         2 |        0        45    22.227
         3 |        0        65    27.453
         4 |        1        25    20.475
         5 |        1        45    25.701
         6 |        1        65    30.926

Specified values of covariates

           |  edyears
 ----------+---------
   Current |       20
```

This would be a good time to experiment with different specifications for **`at()`** that are described in the lecture notes.

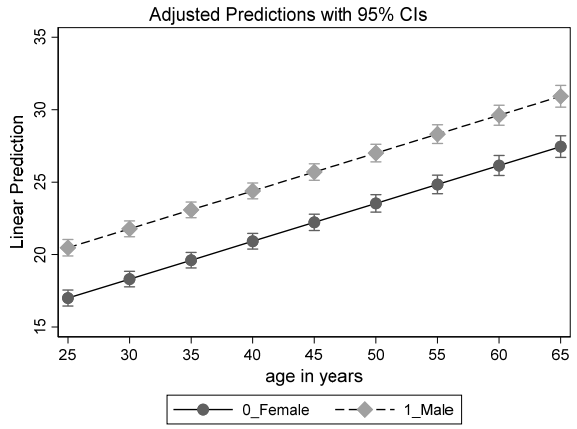
#4 Plot predictions quickly with marginsplot

I use **`margins`** to make predictions that I plot with **`marginsplot`**, which does not work with **`mtable`**.

```
. margins, atmeans at(age=(25(5)65) male=(0 1) edyears=20)

Adjusted predictions                            Number of obs     =      3,997
Model VCE    : OLS

1._at        : male            =           0
               age             =          25
               edyears         =          20
::output omitted ::
18._at       : male            =           1
               age             =          65
               edyears         =          20

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   17.00108   .2830003    60.07   0.000     16.44624    17.55592
::output omitted ::
         18  |   30.92648   .3825464    80.84   0.000     30.17648    31.67649
------------------------------------------------------------------------------
```

To make a quick graph:

```
. marginsplot
. graph export `pgm'-m1wages.`graphfmt', replace
```



See **mcolec*.do** for options to make the graph more elegant.

#5 LRM wages with age-squared using factor syntax

Factor syntax dynamically creates age-squared which is included as a regressor:

```
. sum wages male c.age##c.age edyears

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       wages |      3,997    15.54459     7.85724        2.3      49.92
        male |      3,997    .4978734     .500058          0          1
             |
         age |      3,997    36.95822      12.004         16         65
 c.age#c.age |      3,997     1509.97     934.969        256       4225
             |
     edyears |      3,997    13.21191    3.036544          0         20

. regress wages male c.age##c.age edyears

      Source |       SS           df       MS      Number of obs   =     3,997
-------------+----------------------------------   F(4, 3992)      =    518.78
       Model |  84377.7861         4  21094.4465   Prob > F        =    0.0000
    Residual |  162320.145     3,992   40.661359   R-squared       =    0.3420
-------------+----------------------------------   Adj R-squared   =    0.3414
       Total |  246697.931     3,996   61.736219   Root MSE        =    6.3766

------------------------------------------------------------------------------
       wages |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        male |   3.465888   .2017898    17.18   0.000     3.070267    3.861508
         age |   1.001166   .0517182    19.36   0.000     .8997691    1.102562
 c.age#c.age |  -.0096636   .0006664   -14.50   0.000    -.0109702    -.008357
     edyears |   .8312951   .0340748    24.40   0.000     .7644895    .8981007
       _cons |  -19.57354   .9820115   -19.93   0.000    -21.49883   -17.64825
------------------------------------------------------------------------------
```

#6 Predictions with margins to plot with marginsplot

With factor syntax **margins** knows that when age changes, so does age-squared. Accordingly, these commands are exactly the same as those used for our first model.

```
margins, atmeans at(age=(25(5)65) male=(0 1) edyears=20)
marginsplot // quick and dirty plot
```

Close log and exit program

```
log close
exit
```

Closing the logs makes sure nothing else is written to the file. **exit** tells Stata to ignore any commands that follow. We won't show these commands later in the guide, but always include them in your do-file.

Binary outcomes: mcolab-brm-lfp.do

#1 Load data and check variables

Always verify the variables used in later analyses.

```
use binlfp4, clear
codebook lfp k5 k618 agecat wc hc lwg inc, compactf
tab1 agecat, miss
```

#2 Fit logit

The iterations show the steps taken to "get to the top of the hill". Typically the model will converge quickly. If not, check your data and specification.

```
. logit lfp k5 k618 i.wc i.agecat i.hc lwg inc

Iteration 0:   log likelihood =  -514.8732
Iteration 1:   log likelihood = -453.10297
Iteration 2:   log likelihood = -452.72408
Iteration 3:   log likelihood = -452.72367
Iteration 4:   log likelihood = -452.72367

Logistic regression                             Number of obs     =        753
                                                LR chi2(8)        =     124.30
                                                Prob > chi2       =     0.0000
Log likelihood = -452.72367                     Pseudo R2         =     0.1207
```

Note variables entered as **i.**varname show the value label for the variable, such as **i.wc**:

```
------------------------------------------------------------------------------
         lfp |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          k5 |  -1.391567   .1919279    -7.25   0.000    -1.767739   -1.015395
        k618 |  -.0656678    .068314    -0.96   0.336    -.1995607    .0682251
          wc |
    college  |   .7977136   .2291814     3.48   0.001     .3485263    1.246901
<output omitted>
------------------------------------------------------------------------------
```

Storing estimation

After you fit a model you can store the results. This let's you fit other models and later restore the results from a prior model. Why do this? You don't want to have to repeatedly fit the same model. To store our logit resuls:

```
. estimates store basemodel
```

I can fit another model:

```
. logit lfp k5
::
. est store simplemodel
```

Then restore my earlier model:

```
. est restore basemodel
```

#3 In-sample predictions

```
. predict prlogit
```

Always label new variables:

```
. label var prlogit "Logit: predicted probabilty"
```

The local makes it easy to consistently name, label, and save graphs:

```
. local graphname "basemodel-phat"
```

Usually a simple dotplot is all you need, but you can add other options as shown in lecture. The caption labels the graph so I know where it came from:

```
. dotplot prlogit, ylab(0(.2)1, nogrid) ///
>     caption("`graphname' `tag'", size(*.5) pos(5) col(gs10))

graph export `pgm'-`graphname'.`graphfmt', replace
```

I won't show the **caption** command or exporting in later example. They are, however, included in the sample do-file.

#4.1 DCM: using atmeans

At the mean of all regressors

Here I compute the DCM for an indicator variable **wc** and a continuous variable **k5**:

```
. mchange wc k5, atmeans amount(one sd) stats(est p ll ul) dec(2)

logit: Changes in Pr(y) | Number of obs = 753

Expression: Pr(lfp), predict(pr)

               |    Change    p-value         LL         UL
---------------+-------------------------------------------
wc             |
 college vs no |      0.19       0.00       0.09       0.28
k5             |
           +1  |     -0.32       0.00      -0.40      -0.25
          +SD  |     -0.18       0.00      -0.23      -0.13
```

Next, the command shows the predicted probabilities at the mean of the regressors:

```
Predictions at base value

             | not in LF      in LF
-------------+----------------------
  Pr(y|base) |      0.42       0.58
```

Finally, the base values are shown. In this case they are the means. Always check these to make sure you didn't make an error in your command:

```
Base values of regressors

             |                                 2.         3.         1.         1.
             |        k5       k618     agecat     agecat         wc         hc
-------------+------------------------------------------------------------------
          at |       .24        1.4        .39        .22        .28        .39

             |
             |       lwg        inc
-------------+----------------------
          at |       1.1         20

1: Estimates with margins option atmeans.
```

Other mchange options

Here are some other examples. See Long and Freese or **help mchange** for more options and further explanations. Try exploring the options and see what they do.

Trimmed ranges

```
mchange inc, atmeans amount(range) trim(5) dec(2) brief
```

Holding regressors at other values and change by 1

```
mchange k5, atmeans at(k5=0) amount(sd) delta(1) stat(from to est p) dec(2)
```

Exercise

Compute the effect as the number of children increases from 0 to 2.

#4.2 ADC: removing the atmeans option

```
. mchange wc k5, amount(one sd) stats(est p ll ul) dec(2)

logit: Changes in Pr(y) | Number of obs = 753

Expression: Pr(lfp), predict(pr)

               |    Change    p-value         LL         UL
---------------+-------------------------------------------
wc             |
 college vs no |      0.16       0.00       0.08       0.25
k5             |
            +1 |     -0.28       0.00      -0.34      -0.22
           +SD |     -0.15       0.00      -0.19      -0.12
```

These are the average of the predictions for all observations.

```
Average predictions

             | not in LF      in LF
-------------+----------------------
  Pr(y|base) |      0.43       0.57
```

There are no base values since the change is averaging across observations.

#6 Ideal types

We are creating a table of predictions for a set of ideal types.

Start the table with an average person

The **`atmeans`** option holds variables at their mean; **`clear`** starts with a new table; **`ci`** requests the confidence interval.

```
. mtable, rowname(Average person) atmeans clear ci

Expression: Pr(lfp), predict()

                 |    Pr(y)        ll        ul
 ----------------+----------------------------
  Average person |    0.578     0.539     0.616

Specified values of covariates

           |                              2.        3.        1.        1.
           |       k5      k618      agecat    agecat        wc        hc
 ----------+-------------------------------------------------------------
   Current |     .238      1.35        .385      .219      .282      .392

           |
           |      lwg       inc
 ----------+-------------------
   Current |      1.1      20.1
```

Specifying value of all regressors with at()

```
. mtable, rowname(Younger lower educ w kids) ///
>     at(agecat=1 k5=2 k618=0 inc=10 lwg=.75 hc=0 wc=0) below ci twidth(28)
```

The table includes the results from the prior **`mtable`** command:

```
Expression: Pr(lfp), predict()

                             |    Pr(y)        ll        ul
 ----------------------------+----------------------------
              Average person |    0.578     0.539     0.616
   Younger lower educ w kids |    0.159     0.068     0.251
```

Next, set 1 contains predictions from the first **`mtable`** command:

```
Specified values of covariates

          |                              2.        3.        1.        1.
          |       k5      k618      agecat    agecat        wc        hc
----------+-------------------------------------------------------------
    Set 1 |     .238      1.35        .385      .219      .282      .392
  Current |        2         0           .         .         .         .

          |
          |      lwg       inc    agecat        wc        hc
----------+---------------------------------------------
    Set 1 |      1.1      20.1         .         .         .
  Current |      .75        10         1         0         0
```

Using sum and if to compute levels of variables

Our "young mothers with higher education" are defined as:

```
if agecat==1 & k5==2 & k618==0 & wc==1 & hc==1
```

We us this with sum to compute values of some regressors:

```
. sum lwg if agecat==1 & k5==2 & k618==0 & wc==1 & hc==1
<output omitted>
. local mnlwg = r(mean)

. sum inc if agecat==1 & k5==2 & k618==0 & wc==1 & hc==1
<output omitted>
. local mninc = r(mean)
```

These are use with the atspec:

```
. mtable, at(agecat==1 k5==2 k618==0 wc==1 hc==1 inc=`mninc' lwg=`mnlwg') ///
>     rowname(Young more educ w kids) atmeans below ci twidth(28)
<output omitted>
```

Exercise

Compute predictions for other, substantively motivated ideal types.

#7 Tables of predictions for categorical regressors

In addition to computing predictions for a single profile, you can create tables of predictions based on levels of multiple variables.

#7.1 Table with predictions for wc by k5 using atmeans

```
. mtable, atmeans at(wc=(0 1) k5=(0 1 2 3))
```

When **`at()`** has lists of values for multiple variables, predictions are made for all combinations of the values:

```
Expression: Pr(lfp), predict()

          |       k5        wc     Pr(y)
----------+----------------------------
        1 |        0         0     0.604
        2 |        0         1     0.772
        3 |        1         0     0.275
        4 |        1         1     0.457
        5 |        2         0     0.086
        6 |        2         1     0.173
        7 |        3         0     0.023
        8 |        3         1     0.049
<output omitted>
```

Computing DC's for indicator variables

Since we specified **i.wc**, we can use **dydx(wc)** to compute DC(wc):

```
. mtable, dydx(wc) atmeans at(k5=(0 1 2 3))

Expression: Pr(lfp), predict()

           |       k5   d Pr(y)
 ----------+------------------
         1 |        0     0.168
         2 |        1     0.182
         3 |        2     0.087
         4 |        3     0.027
<output omitted>
```

Exercise

The example holds other variables at their means. Use an if condition to hold variables at the mean conditional on other variables. For example, make predictions holding other variables at the means for those whose husbands went to college.

#8 Options for graphs

Use the example from lecture as templates for your graphs. If you do not like how a graphs looks, find options you like. Then create template do-files that have the options you like. When doing this, you will save time if you use locals with the options you want. After you create these locals, use them in each do-file and forget what they mean!

```
local titleopt "ring(2) pos(11) size(*1)"
local labYopt  "labsiz(*1.1) glwid(*.7) glcol(black*.3) grid gmin gmax"
local labXopt  "labsiz(*1.1) glwid(*.7) glcol(black*.3) nogrid"
local lin1opt  "lcol(blue*1.) lpat(solid) msym(i)  msiz(*1.) mcol(blue*1.)"
local lin2opt  "lcol(red*1.) lpat(dash) msym(i)  msiz(*.9) mcol(red*1.)"
local linDCopt "lcol(green*1.) lpat(solid) msym(i)  msiz(*.9) mcol(green*1.)"
```

These are used in later commands to specify options.

#9 Lowess plots to evaluate functional form

You can quickly check the functional form:

```
. lowess lfp inc
```

You can format the graph with options illustrated in the lecture do-files. Most often, I simply want a quick plot to explore the functional form.

#10 marginsplot plot of predicted probabilities

Making predictions

To perfect my plots, I might start with predictions for income at 0, 50, and 100. This is faster but the graphs are too rough to use:

```
margins, at(inc=(0(50)100)) atmeans
```

When my graph command is perfected, I make more predictions. The less linear the curve, the more points you will need.

```
margins, at(inc=(0(5)100)) atmeans
```

I could suppress the output since I am going to plot it:

```
quietly margins, at(inc=(0(5)100)) atmeans
```

I rarely do this since I might see something that suggests a problem.

Plotting the predictions

Most graph commands only work with variables. **marginsplot**, however, takes the predictions created by **margins** without creating variables. You can quickly look at your results simply running **marginsplot**. Or, I customize the graph using locals defined earlier:

```
. marginsplot, recastci(rarea) ciopts(color(black*.1)) ///
>     ylab(0(.25)1, `labYopt') xlab(0(20)100, `labXopt') ///
>     plot1opts(`lin1opt') plotopts(lwidth(*1)) ///
>     xtitle("Family income excluding wife's") ytitle("Pr(In Labor Force)") ///
>     title("Other variables at their means" " ", `titleopt') ///
>     caption("`graphname' `tag'", size(*.5) pos(5) col(gs10)) scale(1.1)
```

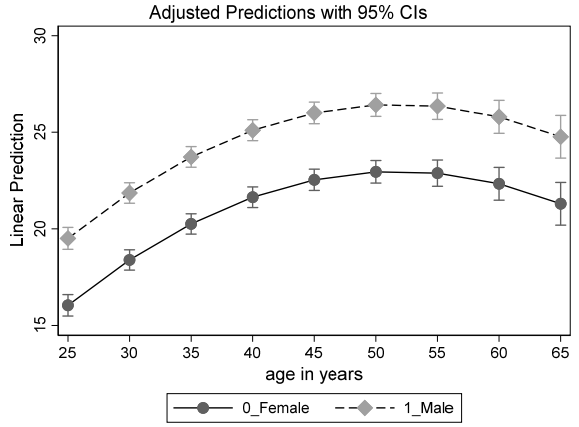
#10.3 Predictions at two levels of wc

Like **mtable** we can make predictions for multiple regressors:

```
. margins, at(inc=(0(5)100) wc=(0 1)) atmeans
<output omitted>

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   .6889161   .0399067    17.26   0.000     .6107004    .7671317
<output omitted>
         42  |   .1286839   .0744985     1.73   0.084    -.0173306    .2746983
------------------------------------------------------------------------------
```

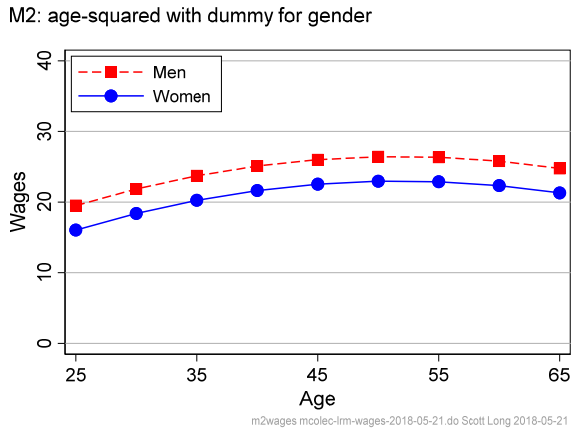
marginsplot figures out that I want to graph two curves:

```
. marginsplot
```



Or, you can add options:

```
. marginsplot, ///
>  recastci(rarea) ci1opts(color(black*.2)) ci2opts(color(black*.1)) ///
>  ylab(0(.25)1, `labYopt') xlab(0(20)100, `labXopt') ///
>  legend(order(4 "Attended college" 3 "Did not attend") ring(0) pos(1) rows(2)) ///
>  plot1opts(`lin1opt') plot2opts(`lin2opt') plotopts(lwidth(*1.2)) ///
>  xtitle("Family income excluding wife's") ytitle("Pr(In Labor Force)") ///
>  title("Other variables at their means" " ", `titleopt') ///
>     caption("`graphname' `tag'", size(*.5) pos(5) col(gs10)) scale(1.1)
```



#10.4 DC(wc|income, atmeans)

Since we specified **`i.wc`**, we use **`dydx(wc)`** to compute DC(wc) for plotting:

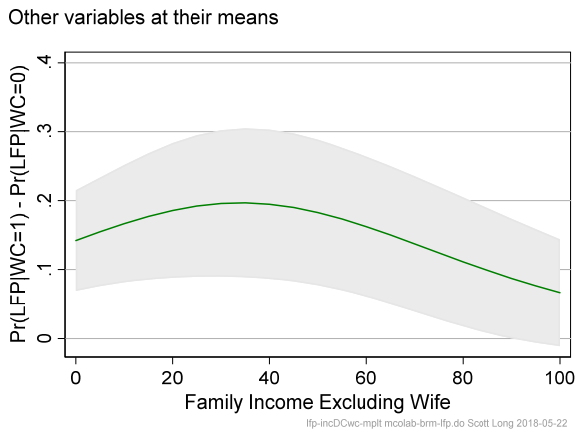
```
. margins, dydx(wc) at(inc=(0(5)100)) atmeans
<output omitted>

Expression   : Pr(lfp), predict()
dy/dx w.r.t. : 1.wc

1._at        : k5              =    .2377158 (mean)
               k618            =    1.353254 (mean)
               1.agecat        =    .3957503 (mean)
               2.agecat        =    .3851262 (mean)
               3.agecat        =    .2191235 (mean)
               0.wc            =    .7184595 (mean)
               1.wc            =    .2815405 (mean)
               0.hc            =    .6082337 (mean)
               1.hc            =    .3917663 (mean)
               lwg             =    1.097115 (mean)
               inc             =           0
<output omitted>
```

`margins` is comparing values of **`wc`** to the base value 0. The **`0.wc`** is the excluded value 0. **`1.wc`** is the comparison of **`wc`**=1 to wc at the base value:

```
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+------------------------------------------------------------------
0.wc         |  (base outcome)
-------------+------------------------------------------------------------------
1.wc         |
         _at |
          1  |   .1420894   .0372932     3.81   0.000     .0689961    .2151828

<output omitted>

         21  |   .0663191   .0395403     1.68   0.093    -.0111785    .1438166
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

`marginsplot` quickly plots results where **`recastci(rarea) ciopts(color(black*.1))`** controls how the confidence intervals look. If you want to know more about these options, check the manual!

```
. marginsplot, recastci(rarea) ciopts(color(black*.1)) ///
>     ylab(0(.1).4, `labYopt') xlab(0(20)100, `labXopt') ///
>     plot1opts(`linDCopt') plotopts(lwid(*1)) ///
>     xtitle("Family Income Excluding Wife") ///
>     ytitle("Pr(LFP|WC=1) - Pr(LFP|WC=0)") ///
>     title("Other variables at their means" " ", `titleopt')
```



#11 Plots using mgen

`margins` and **`marginsplot`** is very and convenient. A limitation of **`marginsplot`** is that it is not easy to combine graphs. For example, put a lowess curve and predicted probabilities in the same figure. You cannot combine predictions from different models. To do this I create variables with **`mgen`** and for use with **`graph`**.

locals for graphs

```
local linDCopt  msym(i) lcol(green) lpat(solid)
local linPR1opt msym(i) lcol(red)   lpat(dash)
local linPR0opt msym(i) lcol(blue)  lpat(solid)
```

Predictions over income with wc at 0

Generated variables begin with the stem from **`stub()`**. **`predlabel()`** creates a label for the variable with the predicted outcome.

```
. mgen, at(inc=(0(5)100) wc=0) atmean stub(PLTwc0) predlabel(Did not attend college)

Predictions from: margins, at(inc=(0(5)100) wc=0) atmean predict(pr)

Variable   Obs Unique      Mean        Min       Max  Label
------------------------------------------------------------------------------
PLTwc0pr1   21     21  .3177494   .0623648  .6889161  Did not attend college
PLTwc0ll1   21     21  .2309727  -.0151898  .6107004  95% lower limit
PLTwc0ul1   21     21   .404526   .1399194  .7671317  95% upper limit
PLTwc0inc   21     21        50          0       100  Family income excluding...
------------------------------------------------------------------------------

Specified values of covariates
<output omitted>
. label var PLTwc0pr "Did not attend college"
```

Predictions over income with wc at 1

```
. mgen, at(inc=(0(5)100) wc=1) atmean stub(PLTwc1) predlabel(Attended college)
<output omitted>
```

Plotting the results

```
. graph twoway ///
>     (rarea PLTwc1ul PLTwc1ll PLTwc1inc, color(black*.1)) ///
>     (rarea PLTwc0ul PLTwc0ll PLTwc0inc, color(black*.2)) ///
>     (connected PLTwc1pr PLTwc1inc, `linPR1opt') ///
>     (connected PLTwc0pr PLTwc0inc, `linPR0opt'), ///
>     xtitle("Family income excluding wife's") ///
>     ytitle("Pr(In Labor Force)") ylab(0(.25)1., grid gmin gmax) ///
>     subtitle("Regressors: k5 k618 agecat wc hc lwg inc", position(11)) ///
>     legend(on order(3 4))
```

Variables for discrete change for wc by inc

```
. mgen, dydx(wc) at(inc=(0(5)100)) atmean stub(PLTdc) predlabel(DC of wc by income)

Predictions from: margins, dydx(wc) at(inc=(0(5)100)) atmean predict(pr)

Variable    Obs Unique      Mean        Min       Max  Label
------------------------------------------------------------------------------
PLTdcd_pr1   21     21  .1507267    .066319  .1967745  DC of wc by income
PLTdcll1     21     21  .0556941  -.0111785  .0895455  95% lower limit
PLTdcul1     21     21  .2457593   .1438166  .3049388  95% upper limit
PLTdcinc     21     21        50          0       100  Family income excludin...
------------------------------------------------------------------------------
<output omitted>
```

Exercise

Create variables with predictions from two model specifications or from logit and probit. Combine the predictions into a single graph.

Complex sampling: mcolab-svy.do

#1 load and check variables

```
use svyhrs4, clear
local rhsvars "age female ed11less ed12 ed1315 ed16plus"
codebook arthritis `rhsvars', compact
```

#2 Declare the survey design

svyset tells Stata how to adjust for complex surveys. You use it once in each do-file.

```
. svyset secu             /// clusters
>     [pweight=kwgtr],    /// weights
>     strata(stratum)     /// stratum
>     vce(linearized) singleunit(missing) // method to compute SEs

      pweight: kwgtr
          VCE: linearized
  Single unit: missing
     Strata 1: stratum
         SU 1: secu
        FPC 1: <zero>
```

#3 Check the svy variables

It is easy to have use the wrong variables for **svyset**. Always check them carefully and go to the dataset's web site for suggestions on how to adjust for the sampling design. If you get them wrong, everything that follows is incorrect.

```
tab1 secu, miss
tab1 stratum, miss
sum kwgtr
dotplot kwgtr
```

Comparing results with different assumptions

If you are unfamiliar with complex surveys in general or with a new dataset, I suggest running experiments to see how sampling adjustments affect the results. Look for things that suggest errors in **svyset**. Here I compare the results from different survey adjustments and store the estimates for later comparison.

#6 logits without survey

```
logit arthritis age i.female ed11less ed1315 ed16plus
estimates store nosvy
```

```
predict nosvyphat
label var nosvyphat "nosvy phat"
```

#7 non-svy with weights and cluster

Most Stata estimation commands allow adjustments for weights and clustering. If there is not a **svy** command for your model, this might be the best alternative for fitting your model.

```
logit arthritis age i.female ed11less ed1315 ed16plus ///
     [pweight=kwgtr], cluster(secu)
estimates store wtclstr
predict wtclstrphat
label var wtclstrphat "wtclstr phat"
```

#8 logits with survey

```
svy: logit arthritis age i.female ed11less ed1315 ed16plus
estimates store svy
predict svyphat
label var svyphat "svy phat"
```

#9 tables of estimated coefficients

estimates table is a convenient way to compare models. **esttab** is a more powerful command for tables. As expected the biggest differences are in the t-values. What is going on with **age** for **wtclstr**? I don't know but if I was doing serious research, I would pursue this further.

```
. estimates table nosvy wtclstr svy, b(%9.3f) t(%9.2f) stats(N) eform

-------------------------------------------------
    Variable |   nosvy      wtclstr       svy
-------------+-----------------------------------
         age |     1.046       1.049       1.049
             |     29.57      910.60       21.92
      female |
     female  |     1.759       1.779       1.779
             |     17.68       12.10       12.99
    ed11less |     1.162       1.206       1.206
             |      3.50        2.57        3.16
      ed1315 |     0.961       0.937       0.937
             |     -0.92       -0.94       -1.21
    ed16plus |     0.703       0.638       0.638
             |     -8.20      -11.47       -8.54
       _cons |     0.054       0.046       0.046
             |    -26.60     -226.92      -19.54
-------------+-----------------------------------
           N |     18341       16862       18375
-------------------------------------------------
                                     legend: b/t
```

#10 Comparing predictions

Sampling adjustments typically have the biggest effect on significance levels, not estimates.

```
. pwcorr nosvyphat wtclstrphat svyphat

             | nosvyphat  wtclstrphat  svyphat
-------------+---------------------------------
   nosvyphat |    1.0000
 wtclstrphat |    0.9984  1.0000
     svyphat |    0.9984  1.0000       1.0000
```

Plot predictions by range

You could also explore model differences with plots:

```
local agerng "(25(2.5)105)"
local cntrlvars "female=1 ed11less=0 ed1315=0 ed16plus=0"
```

```
estimates restore svy
mgen, at(age=`agerng' `cntrlvars') atmeans stub(svyPr)

estimates restore wtclstr
mgen, at(age=`agerng' `cntrlvars') atmeans stub(wtclPr)

estimates restore nosvy
mgen, at(age=`agerng' `cntrlvars') atmeans stub(nosvyPr)
```

Testing regression parameters and assessing model fit: mcolab-test-fit.do

Fit the model and test the data

```
use binlfp4, clear
codebook lfp k5 k618 agecat wc hc lwg inc, compact
tab agecat, missing

logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
```

#2 Legend names of coefficients

With syntax formatting, Stata creates virtual variables with names based on the variables you specify. Regression coefficients are named based on these names. For some commands you need to know the name Stata uses. The option **coeflegend** displays the legend (i.e., name) which you need to use in expressions for testing hypotheses and other things.

```
. logit, coeflegend
<output omitted>

         lfp |      Coef.  Legend
-------------+----------------------------------------------------------------
          k5 |  -1.391567  _b[k5]
        k618 |  -.0656678  _b[k618]
      agecat |
      40-49  |  -.6267601  _b[2.agecat]
        50+  |  -1.279078  _b[3.agecat]
          wc |
    college  |   .7977136  _b[1.wc]
          hc |
    college  |   .1358895  _b[1.hc]
         lwg |   .6099096  _b[lwg]
         inc |  -.0350542  _b[inc]
       _cons |   1.013999  _b[_cons]
```

#3 Wald tests of multiple coefficients

Using the legends I test hypotheses.

Ho: wc = hc = 0

```
. test 1.wc 1.hc // not test wc hc

 ( 1)  [lfp]1.wc = 0
 ( 2)  [lfp]1.hc = 0

           chi2(  2) =   17.83
         Prob > chi2 =    0.0001
```

Ho: wc = hc

```
. test 1.wc = 1.hc

 ( 1)  [lfp]1.wc - [lfp]1.hc = 0
```

```
          chi2(  1) =    3.24
        Prob > chi2 =    0.0719
```

#4 LR tests

Fit the full (unconstrained) model

```
logit lfp k5 k618 i.agecat i.wc i.hc lwg inc
estimates store full
```

Fit the model with constraints wc = hc = 0

```
logit lfp k5 k618 i.agecat          lwg inc, nolog
estimates store dropwchc
```

Compute LR test of Ho: wc=hc=0

```
. lrtest full dropwchc

Likelihood-ratio test                                 LR chi2(2)  =     18.68
```

(Assumption: dropwchc nested in full) Prob > chi2 = 0.0001

Assessing model fit: mcolab-test-fit.do

Information criteria to compare non-nested models

m1: base model

```
logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
estimates store m1
```

After fitting a model **`estat`** (for estimation statistic) show IC statistics:

```
. estat ic

Akaike's information criterion and Bayesian information criterion

------------------------------------------------------------------------------
       Model |        Obs  ll(null)  ll(model)      df         AIC        BIC
-------------+----------------------------------------------------------------
          m1 |        753 -514.8732  -452.7237       9    923.4473   965.0639
------------------------------------------------------------------------------
               Note: N=Obs used in calculating BIC; see [R] BIC note.
```

SPost's **`fitstat`** lets you compare statistics from two model. **`quietly`** suppresses the output until later.

```
. quietly fitstat, ic save
```

m2: adding inc squared and dropping k618 & hc

```
logit lfp k5      i.agecat i.wc      lwg c.inc##c.inc, nolog
estimates store m2

. estat ic

Akaike's information criterion and Bayesian information criterion

------------------------------------------------------------------------------
       Model |        Obs  ll(null)  ll(model)      df         AIC        BIC
-------------+----------------------------------------------------------------
          m2 |        753 -514.8732  -451.7455       8    919.4911   956.4836
------------------------------------------------------------------------------
               Note: N=Obs used in calculating BIC; see [R] BIC note.
```

Comparing models with fitstat

```
. fitstat, ic diff // compares IC statistics from m2 and m1

                         |     Current        Saved   Difference
```

```
-------------------------+-------------------------------------
AIC                      |
                     AIC |     919.491      923.447       -3.956
          (divided by N) |       1.221        1.226       -0.005
-------------------------+-------------------------------------
BIC                      |
         BIC (df=8/9/-1) |     956.484      965.064       -8.580
 BIC (based on deviance) |   -4031.438    -4022.857       -8.580
    BIC' (based on LRX2) |     -79.887      -71.307       -8.580

Difference of    8.580 in BIC provides strong support for current model.
```

Comparing fit with estimates tables

```
. estimates table m1 m2, stats(N bic) b(%9.3f) t(%6.2f)

-------------------------------------
    Variable |    m1          m2
-------------+-----------------------
<output omitted>
-------------+-----------------------
           N |       753         753
         bic |   965.064     956.484
-------------------------------------
                          legend: b/t
```

R2 measures

I rarely find these measures to be useful, but we wrote **`fitstat`** so you do not waste time computing them!

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
<output omitted>
. estimates store m1
. qui fitstat, save // computes fit statistics (including pseudo-R2's)

logit lfp k5      i.agecat i.wc      lwg c.inc##c.inc, nolog
estimates store m2

. fitstat, diff

                         |     Current        Saved   Difference
-------------------------+-------------------------------------
Log-likelihood           |
                   Model |    -451.746     -452.724        0.978
          Intercept-only |    -514.873     -514.873        0.000
-------------------------+-------------------------------------
Chi-square               |
        D (df=745/744/1) |     903.491      905.447       -1.956
          LR (df=7/8/-1) |     126.255      124.299        1.956
                 p-value |       0.000        0.000        1.000
-------------------------+-------------------------------------
R2                       |
                McFadden |       0.123        0.121        0.002
     McFadden (adjusted) |       0.107        0.103        0.004
      McKelvey & Zavoina |       0.216        0.215        0.001
            Cox-Snell/ML |       0.154        0.152        0.002
  Cragg-Uhler/Nagelkerke |       0.207        0.204        0.003
                   Efron |       0.156        0.153        0.003
               Tjur's D  |       0.156        0.153        0.003
                   Count |       0.684        0.676        0.008
        Count (adjusted) |       0.268        0.249        0.018
-------------------------+-------------------------------------
IC                       |
                     AIC |     919.491      923.447       -3.956
        AIC divided by N |       1.221        1.226       -0.005
         BIC (df=8/9/-1) |     956.484      965.064       -8.580
-------------------------+-------------------------------------
Variance of              |
```

```
           e |       3.290        3.290        0.000
     y-star |       4.195        4.192        0.003

Note: Likelihood-ratio test assumes current model nested in saved model.

Difference of    8.580 in BIC provides strong support for current model.
```

Do the Pseudo-R^2s tell you anything useful?

Testing marginal effects: mcolab-test-meffects.do

Fit the model

```
. use binlfp4, clear
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
<output omitted>
```

Plot DCs

You need the experimental **`xmchange`** until **`mchange`** is updated for **`coefplot`**.

Compute DC for changes of 1 or from 0 to 1.

```
. xmchange k5 k618 wc hc, stats(est se z pvalue ll ul) amount(1)
<output omitted>
```

Save predictions to a matrix

You don't have to understand this, just do it!

```
. mat mpart1 = _mctocp
```

Compute DC for an SD change

```
. xmchange lwg inc, stats(est se z pvalue ll ul) amount(sd)
<output omitted>
```

Create matrix with the combined DC results

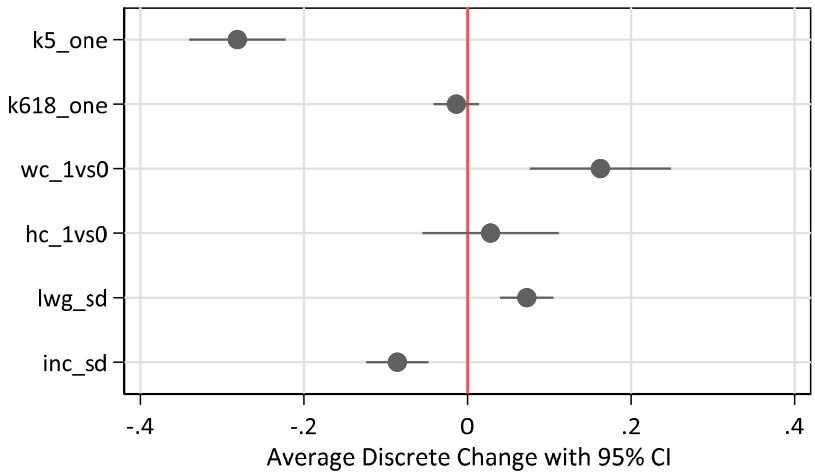
```
. mat mpart2 = _mctocp
. mat dcmat = mpart1 \ mpart2 // combine two matrices
```

To see what is in the matrix, run **`matlist dcmat`**

Make the plot

Jann's **`coefplot`** is great for plotting results. Google coefplot to find his Stata Journal article.

```
. coefplot (matrix(dcmat[,1]), ci((dcmat[,2] dcmat[,3]))) , ///
>     xtitle("Average Discrete Change with 95% CI",size(*.9)) ///
>     xline(0, lcolor(red*.7) lwid(*1.2)) xlabel(-.4(.2).4, grid gmax gmin) ///
>     scale(1.5) ysize(3.5) xsize(6)
```



Comparing DC(wc) and DC(hc)

Posting

To make tests of estimates from **`margins`**, the estimates must be *posted*. This means replacing the regression estimates with estimates from **`margins`**. For example, we fit a logit model:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
<output omitted>
```

Estimates are saved in memory. For example, the coefficient estimates are in e(b):

```
. matlist e(b)

             | lfp
             |                              1b.         2.         3.        0b.
             |        k5       k618     agecat     agecat     agecat         wc
-------------+------------------------------------------------------------------
          y1 | -1.391567  -.0656678          0  -.6267601  -1.279078          0
<output omitted>
```

Since we will replace these estimates with those from **`margins`**, we save them:

```
. estimates store logitmodel
```

I can restore the estimates with **`est restore logitmodel`**.

ADC(wc) and ADC(hc)

The two ADC's are jointly estimated:

```
. margins, dydx(wc hc) post
<output omitted>

             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          wc |
    college  |   .1624037   .0440211     3.69   0.000      .076124    .2486834
             |
          hc |
    college  |   .0281828    .042534     0.66   0.508    -.0551824    .1115479
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

Since we posted the estimates,

```
. matlist e(b) // margins estimates
```

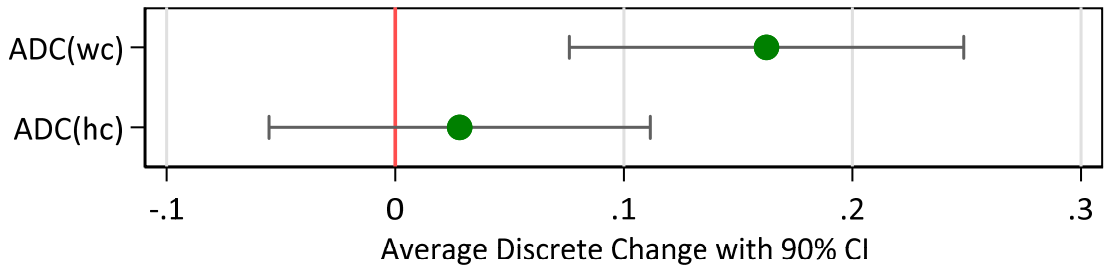
```
             |        0b.         1.         0b.         1.
             |        wc          wc         hc          hc
-------------+--------------------------------------------
          y1 |         0   .1624037          0   .0281828
```

0b.wc and **0b.hc** are the excluded, omitted, or reference categories that can be ignored.

Plotting the effects

coefplot automatically finds the estimates.

```
. coefplot, levels(95) ///
>     xline(0,lcol(red*.7) lwid(*1.2)) xlabel(-.1(.1).3, grid) ///
>     xtitle("Average Discrete Change with 90% CI",size(*.825)) ///
>     grid(none) coeflabels(1.wc="ADC(wc)" 1.hc="ADC(hc)") ///
>     ciopts(recast(rcap)) mcol(green) scale(3.5) ysize(1.5) xsize(6)
```



Testing DC(wc) and DC(hc): indicator variables

Testing a single DC

```
. test 1.wc

 ( 1)  1.wc = 0

           chi2(  1) =   13.61
         Prob > chi2 =    0.0002

. test 1.hc

 ( 1)  1.hc = 0

           chi2(  1) =    0.44
         Prob > chi2 =    0.5076
```

Testing ADC(wc) = DC(hc) = 0

```
. test 1.wc 1.hc

 ( 1)  1.wc = 0
 ( 2)  1.hc = 0

           chi2(  2) =   20.21
         Prob > chi2 =    0.0000
```

Testing if the effects are equal

Using test

```
. test 1.wc = 1.hc

 ( 1)  1.wc - 1.hc = 0

           chi2(  1) =    3.33
         Prob > chi2 =    0.0680
```

Using lincom

lincom computes linear combinations of estimates. Here I estimate the difference in DCs:

```
. lincom  1.wc - 1.hc

 ( 1)  1.wc - 1.hc = 0

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |   .1342209    .073553     1.82   0.068    -.0099403    .2783822
------------------------------------------------------------------------------
```

1.82*1.82 = 3.33 shows that the results are the same as those from **test** since chi-square with one degree of freedom is a z-statistic squared.

Testing DC(inc) and DC(lwg): continuous variables

Since these are continuous, **margins** does cannot compute DCs using **dydx()**. Since the computations are more complex, I start by checking what the right answer should be.

Using mchange to check our work

```
. mchange inc lwg, stats(est se z p ll ul) amount(sd) width(8)
<output omitted>
```

Using margins to compute the DCs

Compute the amount of change

```
. qui sum inc
. local sdinc = r(sd) // needed from margins + SD change
. qui sum lwg
. local sdlwg = r(sd) // needed from margins + SD change
```

Make four predictions

```
. margins, at(inc=gen(inc)) at(inc=gen(inc+`sdinc')) ///
>          at(lwg=gen(lwg)) at(lwg=gen(lwg+`sdlwg')) post

Predictive margins                              Number of obs     =        753
Model VCE    : OIM

Expression   : Pr(lfp), predict()

1._at        : inc             = inc

2._at        : inc             = inc+11.63479853339243

3._at        : lwg             = lwg

4._at        : lwg             = lwg+.5875564251146244

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   .5683931   .0166014    34.24   0.000      .535855    .6009312
          2  |   .4825886   .0257951    18.71   0.000     .4320312    .5331459
          3  |   .5683931   .0166014    34.24   0.000      .535855    .6009312
          4  |   .6408189   .0228361    28.06   0.000      .596061    .6855768
------------------------------------------------------------------------------
```

Compute DCs and second differences

```
lincom  2._at-1._at
```

```
lincom  4._at-3._at
lincom (2._at-1._at)+(4._at-3._at)
```

mlincom is for those who have better things to do than enter long names

```
. qui mlincom 2-1, rowname(DCinc+sd) stats(all) clear
. qui mlincom 4-3, rowname(DClwg+sd) stats(all) add
. mlincom (2-1)+(4-3), rowname(2nd difference) stats(all) add

             |   lincom        se    zvalue    pvalue        ll        ul
-------------+------------------------------------------------------------
    DCinc+sd |   -0.086     0.019    -4.404     0.000    -0.124    -0.048
    DClwg+sd |    0.072     0.017     4.344     0.000     0.040     0.105
2nd differ~e |   -0.013     0.026    -0.521     0.602    -0.064     0.037
```

Test gives us the same result: always double check your results!

```
. test (2._at-1._at)=(-1*(4._at-3._at))

 ( 1)  - 1bn._at + 2._at - 3._at + 4._at = 0

           chi2(  1) =    0.27
         Prob > chi2 =    0.6023
```

#6 Comparing profiles or ideal types

Restore the model since margins, post was run

```
. estimates restore logitmodel // restore logit estimates
```

Simultaneously compute all profiles

```
. mtable, clear ci ///
>     at((mean) _all) ///
>     at(agecat=1 k5=2 k618=0    wc=0 hc=0 inc=10   lwg=0.75) ///
>     at(agecat=1 k5=2 k618==0   wc=1 hc=1 inc=16.6 lwg=1.62) ///
>     at(agecat=2 k5=0 k618=1.37 wc=1 hc=1 inc=27.7 lwg=1.41) ///
>     at(agecat=3 k5=0 k618==0   wc=1 hc=1 inc=27.9 lwg=1.38) ///
>     post

Expression: Pr(lfp), predict()

           |                            2.        3.        1.        1.
           |       k5      k618    agecat    agecat        wc        hc
 ----------+------------------------------------------------------------
         1 |     .238      1.35      .385      .219      .282      .392
::
         5 |        0         0         0         1         1         1

           |
           |      lwg       inc     Pr(y)        ll        ul
 ----------+-------------------------------------------------
         1 |      1.1      20.1     0.578     0.539     0.616
::
         5 |     1.38      27.9     0.630     0.527     0.733
```

Test differences in profile probabilities

```
. mlincom 4 - 5, rowname(MidEdDad-OldHiEd) clear twidth(20)

                     |   lincom    pvalue        ll        ul
---------------------+--------------------------------------
    MidEdDad-OldHiEd |    0.124     0.007     0.034     0.214
```

And so on

```
                          |   lincom    pvalue        ll        ul
--------------------------+--------------------------------------
       Average-YngLoEdMom |    0.418     0.000     0.327     0.510
       Average-YngHiEdMom |    0.184     0.019     0.030     0.337
```

```
        Average-MidEdDad |   -0.176     0.000    -0.242    -0.110
         Average-OldHiEd |   -0.052     0.284    -0.147     0.043
    YngLoEdMom-YngHiEdMom |   -0.235     0.000    -0.340    -0.130
      YngLoEdMom-MidEdDad |   -0.594     0.000    -0.728    -0.461
       YngLoEdMom-OldHiEd |   -0.471     0.000    -0.622    -0.319
      YngHiEdMom-MidEdDad |   -0.360     0.000    -0.525    -0.195
       YngHiEdMom-OldHiEd |   -0.236     0.007    -0.407    -0.064
         MidEdDad-OldHiEd |    0.124     0.007     0.034     0.214
```

Exercise

Use these commands to test hypotheses about marginal effects and profiles that you find substantively interesting. Try different options. If you get errors, you will, try to figure out how to fix them.

Nonlinear right-hand-side

This section deals with models with nonlinearities on the right-hand-side, but explains tools that you can use for many things:

- Constructing graphs that look uniform
- Comparing model specifications using IC measures and predictions
- Plotting the distribution of effects across observations

Graph options

To make graphs consistent, I start by defining options.

```
global nogapnoline "plotregion(margin(zero) lcol(white)) "
global captionopt  "size(*.5) pos(5) col(gs10)"
global lineprob    "msym(i) mcol(blue) lcol(blue) lpat(solid) lwid(*1.4)"
global lined       "msym(i) lcol(green) lpat(dash) lwid(*1.4)" // diabetes
global M1line      "msym(i) lcol(red*1.2) lpat(solid) lwid(*1.4)"
global M2line      "msym(i) lcol(green*1.2) lpat(dash) lwid(*1.4)"
global M3line      "msym(i) lcol(blue*1.2) lpat(shortdash) lwid(*1.4)"
global ylab        "ylab(0(.25)1, grid gmin gmax)"
global ylab4       "ylab(0(.1).4, grid gmax gmin)"
```

Why globals? With locals you need to run the entire do-file so that locals are defined when used in graphs. Globals stay in memory making it easier to debug my plots. I sometimes use globals, but feel guilty when I do.

Set up analysis data

```
use svyhrs4, clear
drop if ///
     arthritis>=. | diabetes>=. | goodhlth>=. | age>=. | female>=. | ed4cat>=.
drop if age<53 // young ages were probably miscoded

svyset secu [pweight=kwgtr], ///
    strata(stratum) vce(linearized) singleunit(missing)

codebook diabetes age female ed4cat, compact

svy: mean diabetes age female
svy: tab ed4cat
```

#1 Local polynomial smoothing compared to logit on age

To assess the functional form, I use **`lpoly`** which is nonparametric. It creates a variable which we compare to predictions from a logit.

```
. lpoly diabetes age if age<100, gen(d_age d_poly) nograph n(200) bwidth(5)
. label var d_poly "Polynomial smooth"
```

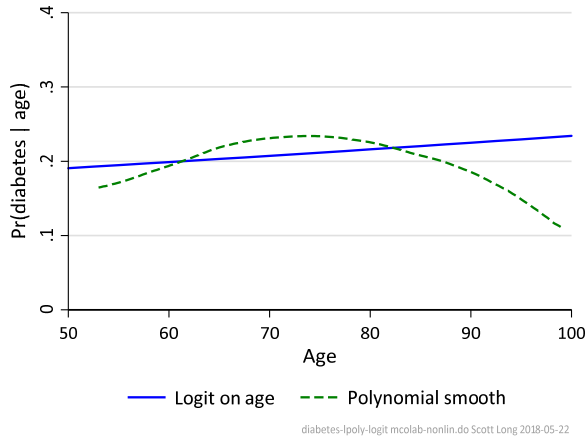
Fit a model with only age and generate predictions to plot

```
. logit diabetes age
<output omitted>

. mgen, at(age=(50(2.5)100)) stub(diab) atmeans replace
<output omitted>
. label var diabpr1 "Logit on age"
```

Compare the nonparametric results to the logit predictions

You should explore the options below by deleting an option and see how the graph changes. Change options and see what changes. The best way to learn **graph** is to try different things. This would also be a good time to try the dialog box for creating graphs.

```
. graph twoway ///
>     (connected diabpr1 diabage, $lineprob) ///
>     (connected d_poly d_age, $lined), ///
>     xlab(50(10)100) xtitle(Age) ///
>     ylin(0 1, lcol(black)) $ylab4 ytitle(Pr(diabetes | age))  ///
>     legend(symxsize(6.9) order(1 2) siz(*1.1) region(lcol(white))) ///
>     $nogapnoline scale(1.2) caption("`graphname' `tag'", $captionopt)
```



#3.1 Fit models with age, age-squared, and age-cubed

```
. svy: logit diabetes c.age i.female i.ed4cat, or
<output omitted>
. est store dMage1

. svy: logit diabetes c.age##c.age i.female i.ed4cat, or
<output omitted>
. est store dMage2

. svy: logit diabetes c.age c.age#c.age c.age#c.age#c.age i.female i.ed4cat
<output omitted>
. est store dMage3
```

Since we stored the estimates we can combine them in a table

```
. estimates table dMage1 dMage2 dMage3, title(diabetes) eform b(%9.5f) p(%9.3f)

diabetes

----------------------------------------------------
    Variable |  dMage1      dMage2      dMage3
-------------+--------------------------------------
         age |   1.00656     1.29691     1.25235
             |     0.003       0.000       0.511
      female |
     female  |   0.80854     0.81816     0.81815
             |     0.000       0.000       0.000
      ed4cat |
```

```
   12 years  |   0.66281     0.65679     0.65678
             |     0.000       0.000       0.000
13-15 years  |   0.54123     0.55383     0.55378
             |     0.000       0.000       0.000
  16+ years  |   0.44993     0.45797     0.45794
             |     0.000       0.000       0.000
 c.age#c.age |               0.99819     0.99869
             |                 0.000       0.784
 c.age#c.age#|
       c.age |                           1.00000
             |                             0.915
             |
       _cons |   0.25513     0.00004     0.00010
             |     0.000       0.000       0.254
-------------------------------------------------
                                       legend: b/p
```

3.2 Compute BICs & AICs

IC measures are not defined with svy estimates. To get a rough idea of the fit, I fit the models without **svy:**

```
. logit diabetes c.age i.female i.ed4cat, or
<output omitted>
. est store nosvyM1

. logit diabetes c.age##c.age i.female i.ed4cat, or
<output omitted>
. est store nosvyM2

. logit diabetes c.age c.age#c.age c.age#c.age#c.age i.female i.ed4cat
<output omitted>
. est store nosvyM3

. estimates table nosvyM1 nosvyM2 nosvyM3, ///
>     stats(bic aic) b(%9.5f) p(%9.3f) stfmt(%9.2f)

-------------------------------------------------
    Variable |  nosvyM1     nosvyM2     nosvyM3
-------------+-----------------------------------
<output omitted>
-------------+-----------------------------------
         bic |  17569.00    17458.86    17467.79
         aic |  17522.40    17404.50    17405.66
-------------------------------------------------
                                       legend: b/p
```

3.3 Predictions against age

I compute predictions for each specification:

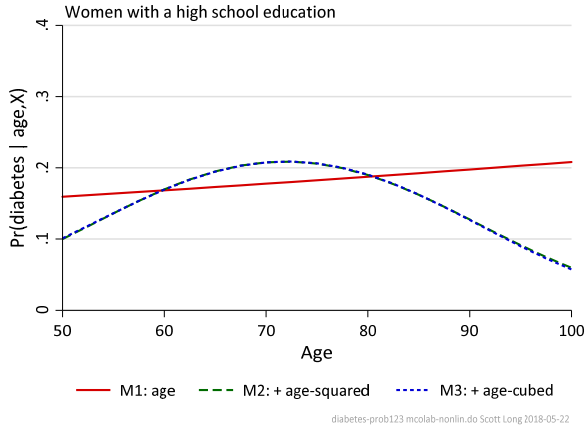
```
. est restore dMage1
. mgen, at(age=(50(2.5)100) female=1 ed4cat=2) atmeans stub(dM1) replace
<output omitted>

. est restore dMage2
. mgen, at(age=(50(2.5)100) female=1 ed4cat=2) atmeans stub(dM2) replace
<output omitted>

. est restore dMage3
. mgen, at(age=(50(2.5)100) female=1 ed4cat=2) atmeans stub(dM3) replace
<output omitted>
```

Plotting predictions from three models

As an exercise, try plotting only two models.

```
. global title     "Women with a high school education"
. label var dM1pr1 "M1: age"
```

```
. label var dM2pr1 "M2: + age-squared"
. label var dM3pr1 "M3: + age-cubed"
. global ytitled   "ytitle(Pr(diabetes | age,X))"

. local graphname "diabetes-prob123"
. graph twoway ///
>     (connected dM1pr1 dM1age, $M1line) ///
>     (connected dM2pr1 dM2age, $M2line) ///
>     (connected dM3pr1 dM3age, $M3line), ///
>     title("$title", position(11) size(*.8)) ///
>     xtitle("Age") xlab(50(10)100) ///
>     $ytitled $ylab4 ///
>     legend(symxsize(6.9) rows(1) size(*1) region(lcolor(white))) ///
>     $nogapnoline scale(1.1) ///
>     caption("`graphname' `tag'", $captionopt)
```



3.4 What is the effect of age

While the plot makes clear that M1 will distort the results, another way to see this is to compare the effect of age at different values of age:

```
estimates restore dMage1
mchange age, amount(delta) delta(10) stats(est p se)                    brief
mchange age, amount(delta) delta(10) stats(est p se) atmeans at(age=50) brief
mchange age, amount(delta) delta(10) stats(est p se) atmeans at(age=70) brief
mchange age, amount(delta) delta(10) stats(est p se) atmeans at(age=90) brief

estimates restore dMage2
mchange age, amount(delta) delta(10) stats(est p se)                    brief
mchange age, amount(delta) delta(10) stats(est p se) atmeans at(age=50) brief
mchange age, amount(delta) delta(10) stats(est p se) atmeans at(age=70) brief
mchange age, amount(delta) delta(10) stats(est p se) atmeans at(age=90) brief
```

#3.5 Plot distribution of effects of age

Plotting the distribution of effects across observations also highlights important differences between the two models. To do this, I use **`margins`** to compute effects.

DC(age+sd) in M1

```
. estimates restore dMage1
```

Predictions are made at the observed age and the observed age plus 10:

```
. margins, at(age=gen(age)) at(age=gen(age + 10)) generate(PR1age)
::
------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         _at |
```

```
          1  |   .1838638   .0034758    52.90   0.000      .176901    .1908266
          2  |   .1937208   .0048042    40.32   0.000     .1840968    .2033448
------------------------------------------------------------------------------
```

ADC(age+10) is the difference in the average predictions:

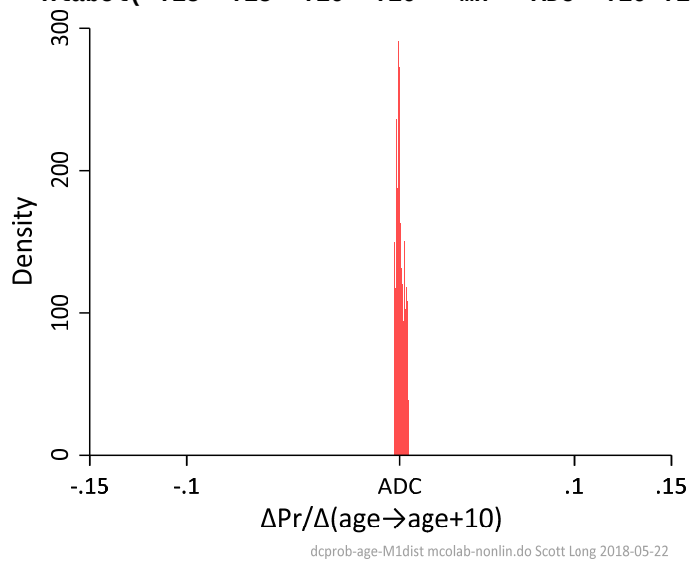
```
. gen DC1age = PR1age2 - PR1age1
. lab var DC1age "DC for 10 year increase in age"
```

The svy mean is the same as the result from **mchange age**

```
. svy: mean DC1age
::
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
      DC1age |    .009857   .0000343      .0097883    .0099256
-------------------------------------------------------------

. estimates restore dMage1
. mchange age, amount(sd) delta(10) dec(7) // verify restuls
::
Expression: Pr(diabetes), predict(pr)

             |    Change    p-value
-------------+----------------------
age          |
      +delta | 0.0098570  0.0031531
::
```

The distribution of effects across observations

```
. histogram DC1age, fcol(red*.7) lcol(red*.7) ///
>   xtitle(`"{&Delta}Pr/{&Delta}(age{&rarr}age+10)"') ///
>   scale(1.3) xsize(8) ysize(6.5) plotregion(margin(zero) lcol(white)) ///
>   xlabel(-.15 -.15 -.10 -.10  `mn' "ADC" .10 .10 .15 .15)
```

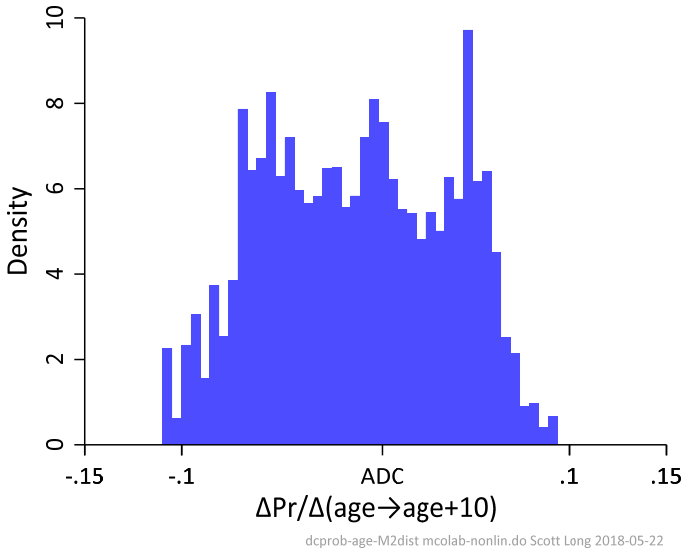


The same thing can be done for M2

```
. estimates restore dMage2
. margins, at(age=gen(age)) at(age=gen(age + 10)) generate(PR2age)
::
. gen DC2age = PR2age2 - PR2age1
. lab var DC2age "DC for 10 year increase in age"
. svy: mean DC2age
::
. estimates restore dMage2
. mchange age, amount(sd) delta(10) dec(7) // verify restuls
```

::

```
. histogram DC2age, fcol(blue*.7) lcol(blue*.7) ///
>   xtitle(`"{&Delta}Pr/{&Delta}(age{&rarr}age+10)"') ///
>   scale(1.3) xsize(8) ysize(6.5) plotregion(margin(zero) lcol(white)) ///
>   xlabel(-.15 -.15 -.10 -.10  `mn' "ADC" .10 .10 .15 .15)
```



Comparing groups: mcolab-groups-diabetes.do

The lecture notes also have a detailed discussion of the Stata code needed to make group comparisons.

#1 Overview on specifying the RHS

While we are not primarily interested in the regression coefficients, it is useful to fit the model in a way that all coefficients are shown for each group. To do this, use this mysterious specification:

```
logit diabetes ibn.white ibn.white#(c.age), nocon
```

noncon suppresses estimating the constant and **ibn** means the indicator variable does not have a base value (**b**=base; **n**=none). My advice is to take my word for it and specify your models this way. If you want to understand why this works, try these:

```
logit diabetes            i.white##(c.age)
logit diabetes          ibn.white##(c.age)
logit diabetes ibn.white ibn.white#(c.age)
```

Specify our model

I create a global with the RHS varialbles. Recall, ## means that age and age-squared are in the model. The **c.** is essential (to understand why, run the do-file with the **c.**s).

```
global rhsvarsfs i.female i.highschool c.age##c.age i.obese
```

The specification for the model with interactions by group:

```
global rhsfull ibn.white ibn.white#($rhsvarsfs)
```

#2 Setup for complex survey data

```
use cda-hrs4, clear
svyset secu [pweight=kwgtr], strata(stratum) vce(linearized) singleunit(centered)
```

#3 Descriptive statistics using svy adjustments

With complex samples, you use **svy** commands compute descriptive statistics and test differences. To look at the distribution of the outcome:

```
. svy: tab white, obs
(running tabulate on estimation sample)

Number of strata   =        56                  Number of obs     =      18,452
Number of PSUs     =       112                  Population size   =  76,507,925
                                                Design df         =          56

--------------------------------
Is white? | proportion         obs
----------+---------------------
 nonwhite |      .1402        3516
    white |      .8598     1.5e+04
          |
    Total |           1     1.8e+04
--------------------------------
  Key:  proportion  =  cell proportion
        obs         =  number of observations
```

To compute means by group, I use **svy: mean** with an **if** statement. For example,

```
. svy: mean diabetes if white==1

Survey: Mean estimation

Number of strata =      56       Number of obs   =      14,922
Number of PSUs   =     112       Population size =  65,706,420
                                 Design df       =          56

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
    diabetes |    .166764   .0040873      .1585761    .1749518
--------------------------------------------------------------
```

The Std. Err. is the standard error of the estimated mean, not the SD of diabetes. To compute the SD:

```
. estat sd

------------------------------------
             |       Mean   Std. Dev.
-------------+----------------------
    diabetes |    .166764    .3727786
------------------------------------
```

Since I only want the mean and SD, I suppress the output from **svy: mean**:

```
qui svy: mean diabetes if white==1
estat sd
```

Similar commands are run for other variables and for **white==0**.

Testing for group differences in means of variables

tabulate is used to test for group differences in binary variables:

```
. svy: tabulate white diabetes

Number of strata   =        56                  Number of obs     =      18,432
Number of PSUs     =       112                  Population size   =  76,421,808
                                                Design df         =          56

--------------------------------------
          |   Respondent has diabetes?
```

```
Is white? |       no  diabetes     Total
----------+----------------------------
 nonwhite |    .1002       .04     .1402
    white |    .7164     .1434     .8598
          |
    Total |    .8167     .1833         1
-------------------------------------------
  Key:  cell proportion

  Pearson:
    Uncorrected   chi2(1)         =  207.6136
    Design-based  F(1, 56)        =  175.4538     P = 0.0000
```

For continuous variables, I use **`regress`** (there is probably an easier way to do this that I am not aware of). The test of **`white`** indicates differences in the mean of **`age`**:

```
. svy: reg age white

Survey: Linear regression

Number of strata   =        56                 Number of obs     =      18,459
Number of PSUs     =       112                 Population size   =  76,507,925
                                               Design df         =          56
                                               F(   1,     56)   =       76.68
                                               Prob > F          =      0.0000
                                               R-squared         =      0.0095

             |             Linearized
         age |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       white |   2.921424    .333632     8.76   0.000     2.253079     3.58977
       _cons |   63.99542   .2831165   226.04   0.000     63.42827    64.56257
------------------------------------------------------------------------------
```

The **`cdalec`** do-file illustrates how to use loops and matrices to automate this process.

#4 Logit of diabetes

In the left column, the names **`white`**#*var* indicates the coefficient for *var* for whites; **`nonwhite`**#*var* is the coefficient for non-whites:

```
. svy: logit diabetes $rhsfull, nocon nolog

Survey: Logistic regression

Number of strata   =        56                 Number of obs     =      17,966
Number of PSUs     =       112                 Population size   =  74,498,325
                                               Design df         =          56
                                               F(  12,     45)   =      354.63
                                               Prob > F          =      0.0000

             |             Linearized
    diabetes |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       white |
   nonwhite  |  -11.36481   1.968854    -5.77   0.000     -15.3089   -7.420726
      white  |  -11.06384   1.332771    -8.30   0.000     -13.7337   -8.393981
             |
<output omitted>
             |
  white#c.age |
   nonwhite  |   .2889341    .057565     5.02   0.000     .1736175    .4042507
      white  |   .2554641   .0374087     6.83   0.000     .1805254    .3304028

<output omitted>
             |
  white#obese |
   nonwhite #|
```

```
       Obese  |   .9088643   .1120878     8.11   0.000     .6843254    1.133403
 white#Obese  |   1.173811   .0611695    19.19   0.000     1.051274    1.296348
------------------------------------------------------------------------------

. estimates store fullmodel
```

#6 Plotting diabetes over age by race

Predictions are made using **mgen** for each group:

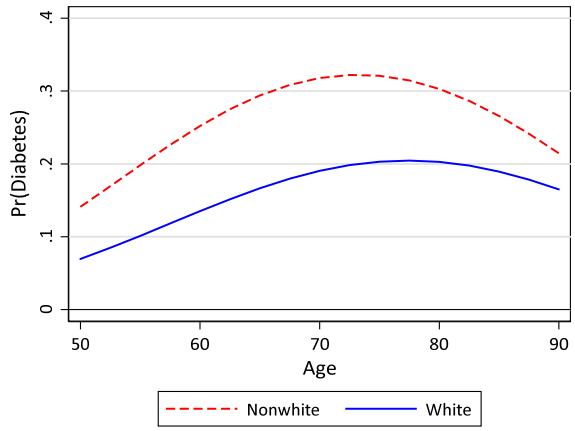
```
. mgen, at(white=1 age=(50(2.5)90)) atmeans stub(ALprW) ///
>     predlabel(White) replace // probabilities white
<output omitted>

. mgen, at(white=0 age=(50(2.5)90)) atmeans stub(ALprN) ///
>     predlabel(Nonwhite) replace // probabilities nonwhite
<output omitted>
```

Group differences at each age are computed with the **dydx(white)** option, where the variable labels are added since these will appear in the graph:

```
. mgen, dydx(white) at(age=(50(2.5)90)) atmeans stub(ALdc) predlabel(DC) replace // DCrace
<output omitted>
. label var ALdcll1 "95% confidence interval"
. label var ALdcul1 "95% confidence interval"
```

#6.1 Plotting probabilities over age by group

```
. global LINwht clcol(blue) lpat(solid) lwid(*1.2) msym(i)
. global LINnon clcol(red) lpat(dash) lwid(*1.2) msym(i)

. twoway  (connected ALprWpr1  ALprWage, $LINwht) ///
>         (connected ALprNpr1  ALprWage, $LINnon), ///
>         ytitle("Pr(Diabetes)") xtitle(Age) ///
>         xlabel(50(10)90) ylabel(0(.1).4, grid gmin gmax) ///
>         yline(0, lcol(black) lwid(*.6)) ///
>         legend(rows(1) order(2 1)) scale(1.2)
```



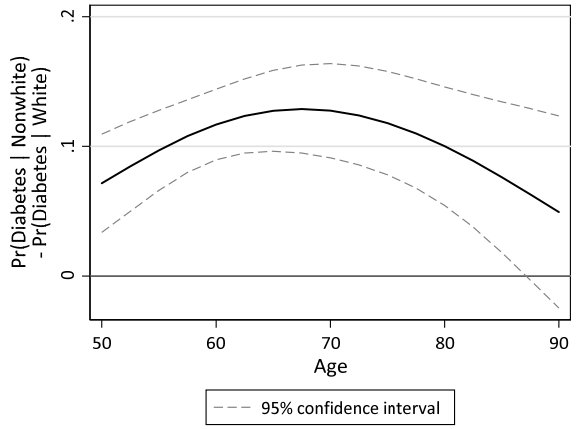
#6.2 Plotting race differences

I define more some globals with plot options:

```
. global LINdc  clcol(black*1) lpat(solid) lwid(*1.2) msym(i)
. global LINci  msym(i i) clwid(*.7 *.7) clcol(black*.5 black*.5) clpat(dash dash)
```

When I plotted the results from **mgen, dydx(white)** I preferred the nonwhite – white difference rather than the white – nonwhite. To see what I did this, try plotting the white – nonwhite differences. To change the direction of the comparison, I multiply variable by -1 (a difference of .2 between whites and nonwhites becomes a difference of -.2 between non-whites and whites):

```
. gen RALdcd_pr1  = -1 * ALdcd_pr1
. gen RALdcll1    = -1 * ALdcll1
. gen RALdcul1    = -1 * ALdcul1
. label var RALdcll1 "95% confidence interval"
. label var RALdcul1 "95% confidence interval"

. twoway ///
>     (connected RALdcd_pr1        ALdcage,  $LINdc)  /// line
>     (connected RALdcll1 RALdcul1 ALdcage,  $LINci), /// ci
>     ::
```



#7 DC(obesity | race)

I want to test if the effect of obesity is the same for both groups. Since the commands are complicated, I start by computing DC(obese|white=1) and DC(obese|white=0) with **mchange**. While this does not let me test if they are equal, it gives me results to compare with those computed using **mtable** and **mlincom**. With complex analyses, always to compute the same thing two ways.

Determining what the DC's should be

The trick to using **mchange** to compute the effects for both groups with the same means is to use the option **(omeans) _all** to specify the overall means for all variables. Then:

```
. mchange obese, at((omeans) _all white=1)

Expression: Pr(diabetes), predict(pr)

              |    Change    p-value
--------------+----------------------
obese         |
 Obese vs Not |     0.196      0.000

Base values of regressors

              |                   1.         1.                   1.
              |     white     female  highsch~l        age      obese
--------------+-----------------------------------------------------
          at  |         1       .538       .807       66.5       .297

. mchange obese, at((omeans) _all white=0)

Expression: Pr(diabetes), predict(pr)

              |    Change    p-value
--------------+----------------------
obese         |
 Obese vs Not |     0.203      0.000

Base values of regressors
```

```
            |                    1.         1.                    1.
            |     white     female  highsch~l        age       obese
------------+------------------------------------------------------
         at |         0       .538       .807       66.5        .297
```

The base values are the same for both sets of predictions, except, appropriately, for the value of **white**.

Probability by race by female by obese at means

These predictions allow me to compute the effects I want:

```
. mtable, at(white=(0 1) obese=(0 1)) atmeans stats(est pvalue) post

Expression: Pr(diabetes), predict()

           |    white     obese     Pr(y)         p
 ----------+-----------------------------------------
         1 |        0         0     0.249     0.000
         2 |        0         1     0.452     0.000
         3 |        1         0     0.130     0.000
         4 |        1         1     0.326     0.000
::
```

With **mlincom**, I compute DC(obese|race) and test if the effects are the same across groups:

```
. mlincom 2-1, rowname(DCobese | nonwhite) clear twidth(32)
::
. mlincom 4-3, rowname(DCobese | white) add twidth(32)
::
. mlincom (4-3)-(2-1), rowname(Race difference in DCobese) add twidth(32)

                                |   lincom    pvalue        ll        ul
--------------------------------+-----------------------------------------
             DCobese | nonwhite |    0.203     0.000     0.154     0.251
                DCobese | white |    0.196     0.000     0.174     0.218
     Race difference in DCobese |   -0.007     0.810    -0.062     0.049
```

The effects match those from **mchange**.

#8 ADC as summary measures

For indicator variables I can compute ADC(female) for each group if I use the **over(white)** option. **over()** tells **margins** and **mtable** to make estimates using the subset of observations for each value of the over variable:

```
. est restore fullmodel
. mtable, dydx(female) over(white) post brief dec(4)

Expression: Pr(diabetes), predict()

           |  d Pr(y)
 ----------+---------
  nonwhite |   0.0018
     white |  -0.0394
```

mlincom puts these into a table and tests if they are equal:

```
. qui mlincom 2,   rowname(ADC female: White)      twidth(25) add clear
. qui mlincom 1,   rowname(ADC female: Nonwhite)   twidth(25) add
. mlincom (2-1), rowname(ADC female: Difference) twidth(25) add

                          |   lincom    pvalue        ll        ul
--------------------------+-----------------------------------------
ADC female                |
                    White |   -0.039     0.000    -0.055    -0.024
                 Nonwhite |    0.002     0.927    -0.038     0.042
               Difference |   -0.041     0.069    -0.086     0.003
```

Computing ADCs for continuous is more complicated. An example shows how:

```
. est restore fullmodel
. mtable, at(age=gen(age)) at(age=gen(age+5)) over(white) post brief dec(4)

Expression: Pr(diabetes), predict()

               |    Pr(y)
 --------------+---------
    0.white#c.1 |   0.2869
    1.white#c.1 |   0.1671
    0.white#c.2 |   0.3126
    1.white#c.2 |   0.1835
```

The probability labeled **`1.white#c.1`** is for whites from the first atspec; **`1.white#c.2`** is for whites from the second atspec. Then:

```
. qui mlincom (4-2), rowname(ADC age+5: White)    clear
. qui mlincom (3-1), rowname(ADC age+5: Nonwhite) add
. mlincom (4-2) - (3-1), rowname(ADC age+5: Difference) twidth(25) add

                          |   lincom    pvalue        ll        ul
--------------------------+---------------------------------------
ADC age+5                 |
                    White |    0.016     0.000     0.013     0.020
                 Nonwhite |    0.026     0.000     0.015     0.037
               Difference |   -0.009     0.099    -0.020     0.002
```

As David Drukker likes to say, "Done and done."

Comparing marginal effects: mcolab-cme-mediate.do

There are several **`cdalec`** do-files that demonstrate estimating and testing marginal effects from different models. These take the same approach: (1) fit the models simultaneously using **`gsem`**; (2) simultaneously estimate the effects to be compared using **`margins`**; and (3) test equality with **`test`**, **`lincom`**, or **`mlincom`**. The example here illustrates how to compare the effect of a variable across nested models.

#1 Data management

```
use gss7216
drop if year < 2000
drop if employed != 1
drop if missing(vhappy,college,wages,occprest,age,married, ///
                parent,woman,conserv,reltrad)
```

Control variables

Since several models use the same control variables, I place them in a local:

```
local controls ///
    i.married i.parent i.woman i.conserv i.reltrad i.year c.age##c.age i.married
```

Before fitting the model, I verify the variables and check the sample size.

#2 Run separate models to check AME(college)

Since simultaneous estimation is more complicated, I first fit the models individual and estimate effects:

```
. logit vhappy i.college, or
::
       vhappy | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------+----------------------------------------------------------------
      college |
College De..  |   1.392215   .0653343     7.05   0.000     1.269874    1.526341
        _cons |   .3973388   .0112859   -32.49   0.000     .3758232    .4200862
```

```
. mchange college, stat(est se p)

Expression: Pr(vhappy), predict(pr)

                                |    Change    Std Err    p-value
--------------------------------+---------------------------------
college                         |
 College Degree vs No Col Degree |     0.072     0.0103      0.000
```

Adding controls:

```
logit vhappy i.college `controls', or
mchange college, stat(est se p)
```

Plus wages:

```
logit vhappy i.college c.wages `controls', or
mchange college, amount(sd) stat(est se p)
```

and occupational prestige:

```
logit vhappy i.college c.wages c.occprest `controls', or
mchange college wages occprest, amount(sd) stat(est se p)
```

#3 simultaneous fitting with GSEM

Clone outcome so that **gsem** can fit multiple models with the same outcome:

```
clonevar vhappyM1 = vhappy // M1: College only
lab var  vhappyM1 "M1 vhapppy college only"
clonevar vhappyM2 = vhappy // M2: add controls
lab var  vhappyM2 "M2 vhapppy college + controls"
clonevar vhappyM3 = vhappy // M3: add wages
lab var  vhappyM3 "M3 vhapppy college + controls + wages"
clonevar vhappyM4 = vhappy // M4: add occprest
lab var  vhappyM4 "M4 vhapppy college + controls + wages + occprest"

. gsem (vhappyM1 <- i.college, logit) ///
>      (vhappyM2 <- i.college `controls', logit) ///
>      (vhappyM3 <- i.college c.wages `controls', logit) ///
>      (vhappyM4 <- i.college c.wages c.occprest `controls', logit) ///
>      , vce(robust)

<output omitted>

Response         : vhappyM1
Family           : Bernoulli
Link             : logit
::
Response         : vhappyM4
Family           : Bernoulli
Link             : logit

Log pseudolikelihood = -21863.725

------------------------------------------------------------------------------
             |               Robust
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
vhappyM1     |
             |
     college |
College De..  |   .3308957   .0469309     7.05   0.000     .2389129    .4228785
       _cons |  -.9229659   .0284054   -32.49   0.000    -.9786394   -.8672924
-------------+----------------------------------------------------------------
vhappyM2     |
             |
     college |
```

```
College De..  |   .2934779   .0508658     5.77   0.000     .1937827    .3931732
<output omitted>
--------------+------------------------------------------------------------------
vhappyM3      |
              |
      college |
College De..  |   .1779871   .0543726     3.27   0.001     .0714187    .2845555
<output omitted>
--------------+------------------------------------------------------------------
vhappyM4      |
              |
      college |
College De..  |   .0956036   .0583111     1.64   0.101    -.0186841    .2098913
<output omitted>
------------------------------------------------------------------------------

. est store gsemmodel
```

#4 compute ADC(college) for each model and compare across models

Since all models were fit simultaneously, margins computes the DC for college from all models simultaneously.

```
. margins, dydx(college) post

Average marginal effects                        Number of obs     =      9,216

dy/dx w.r.t. : 1.college
1._predict   : Predicted mean (M1 vhapppy college only), predict(mu
               outcome(vhappyM1))
2._predict   : Predicted mean (M2 vhapppy college + controls), predict(mu
               outcome(vhappyM2))
3._predict   : Predicted mean (M3 vhapppy college + controls + wages),
               predict(mu outcome(vhappyM3))
4._predict   : Predicted mean (M4 vhapppy college + controls + wages +
               occprest), predict(mu outcome(vhappyM4))

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
0.college    |  (base outcome)
-------------+----------------------------------------------------------------
1.college    |
    _predict |
           1 |    .071806   .0103344     6.95   0.000     .0515509    .0920611
           2 |   .0599197   .0105299     5.69   0.000     .0392813     .080558
           3 |   .0359563   .0111027     3.24   0.001     .0141954    .0577172
           4 |   .0191807   .0117766     1.63   0.103     -.003901    .0422625
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

Then **`mlincom`** creates a table of results:

```
qui {
    mlincom 1,   rowname(ADCcollege: M1) stat(est se p) clear
    mlincom 2,   rowname(ADCcollege: M2) stat(est se p) add
    mlincom 1-2, rowname(ADCcollege: M1 - M2) stat(est se p) add
    mlincom 1,   rowname(ADCcollege: M1) stat(est se p) add
    mlincom 3,   rowname(ADCcollege: M3) stat(est se p) add
    mlincom 1-3, rowname(ADCcollege: M1 - M3) stat(est se p) add
    mlincom 1,   rowname(ADCcollege: M1) stat(est se p) add
    mlincom 4,   rowname(ADCcollege: M4) stat(est se p) add
    mlincom 1-4, rowname(ADCcollege: M1 - M4) stat(est se p) add
    mlincom 2,   rowname(ADCcollege: M2) stat(est se p) add
    mlincom 3,   rowname(ADCcollege: M3) stat(est se p) add
    mlincom 2-3, rowname(ADCcollege: M2 - M3) stat(est se p) add
    mlincom 2,   rowname(ADCcollege: M2) stat(est se p) add
    mlincom 4,   rowname(ADCcollege: M4) stat(est se p) add
```

```
    mlincom 2-4, rowname(ADCcollege: M2 - M4) stat(est se p) add
    mlincom 3,   rowname(ADCcollege: M3) stat(est se p) add
    mlincom 4,   rowname(ADCcollege: M4) stat(est se p) add
}
    mlincom 3-4, rowname(ADCcollege: M3 - M4) stat(est se p) add twidth(15)
```

When **`mlincom`** has a single number, like **`mlincom 1`**, it lists the DC. When there is a difference, like **`mlincom 1-2`**, it computes and tests if the difference is 0.

```
                |   lincom        se    pvalue
----------------+------------------------------
ADCcollege      |
             M1 |    0.072     0.010     0.000
             M2 |    0.060     0.011     0.000
        M1 - M2 |    0.012     0.004     0.003
             M1 |    0.072     0.010     0.000
             M3 |    0.036     0.011     0.001
        M1 - M3 |    0.036     0.006     0.000
             M1 |    0.072     0.010     0.000
             M4 |    0.019     0.012     0.103
        M1 - M4 |    0.053     0.007     0.000
             M2 |    0.060     0.011     0.000
             M3 |    0.036     0.011     0.001
        M2 - M3 |    0.024     0.004     0.000
             M2 |    0.060     0.011     0.000
             M4 |    0.019     0.012     0.103
        M2 - M4 |    0.041     0.006     0.000
             M3 |    0.036     0.011     0.001
             M4 |    0.019     0.012     0.103
        M3 - M4 |    0.017     0.004     0.000
```

Nominal Outcomes: mcolab-nrm-nomocc.do

While MNLM has many parameters you can interpret the model easily if you focus on the marginal effects of each variable on each outcome. Or, you plot the probability of each outcome as a regressor changes. For categorical predictors, you can create tables of predicted probabilities. Except for methods of interpretation based on odds ratios, each of these methods can be used for ordinal models. Many of the methods in the section for ordinal models could also be used for the MNLM.

#1 Load and check the data

```
use partyid4, clear
tab1 party educ, miss
codebook party age income black female educ, compact
```

#12 Fit model and run omnibus tests

```
. mlogit party age10 income10 i.black i.female i.educ, nolog

Multinomial logistic regression                 Number of obs     =      1,382
                                                LR chi2(24)       =     311.25
                                                Prob > chi2       =     0.0000
Log likelihood = -1960.9107                     Pseudo R2         =     0.0735

       party |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
StrDem       |
       age10 |   .2361662    .049609     4.76   0.000     .1389342    .3333981
    income10 |  -.0727873   .0409511    -1.78   0.075      -.15305    .0074754
       black |
         yes |   .9963281   .1983914     5.02   0.000     .6074881    1.385168
      female |
         yes |  -.2408435   .1668491    -1.44   0.149    -.5678617    .0861747
        educ |
    hs only  |  -.3451019   .2228717    -1.55   0.122    -.7819225    .0917187
    college  |  -.6286363   .2952132    -2.13   0.033    -1.207244    -.050029
       _cons |  -1.149873   .3706106    -3.10   0.002    -1.876256    -.423489
```

```
-------------+------------------------------------------------------------------
Dem          |  (base outcome)
-------------+------------------------------------------------------------------
Indep        |
```

<output omitted>

The coefficients are for comparisons with the based category **Dem** (the largest category). You can make comparisons to other categories with the option **baseoutcome()**.

Since some post-estimation commands replace the model estimates, I store the estimates:

```
. estimates store nrm
```

Examining coefficients for all contrasts

If you want to look at the coefficients for all comparisons, use **listcoef** which shows regression coefficients along with factor change coefficients.

```
. listcoef

mlogit (N=1382): Factor change in the odds of party

Variable: age10 (sd=1.678)
------------------------------------------------------------------------------
                            |         b        z    P>|z|       e^b   e^bStdX
----------------------------+-------------------------------------------------
StrDem       vs Dem         |    0.2362    4.761    0.000     1.266     1.486
StrDem       vs Indep       |    0.3162    4.781    0.000     1.372     1.700
StrDem       vs Rep         |    0.2453    4.576    0.000     1.278     1.509
StrDem       vs StrRep      |    0.0282    0.438    0.662     1.029     1.048
Dem          vs StrDem      |   -0.2362   -4.761    0.000     0.790     0.673
Dem          vs Indep       |    0.0800    1.287    0.198     1.083     1.144
Dem          vs Rep         |    0.0092    0.195    0.845     1.009     1.015
Dem          vs StrRep      |   -0.2080   -3.508    0.000     0.812     0.705
Indep        vs StrDem      |   -0.3162   -4.781    0.000     0.729     0.588
Indep        vs Dem         |   -0.0800   -1.287    0.198     0.923     0.874
Indep        vs Rep         |   -0.0708   -1.099    0.272     0.932     0.888
Indep        vs StrRep      |   -0.2880   -3.875    0.000     0.750     0.617
Rep          vs StrDem      |   -0.2453   -4.576    0.000     0.782     0.662
Rep          vs Dem         |   -0.0092   -0.195    0.845     0.991     0.985
Rep          vs Indep       |    0.0708    1.099    0.272     1.073     1.126
Rep          vs StrRep      |   -0.2171   -3.594    0.000     0.805     0.695
StrRep       vs StrDem      |   -0.0282   -0.438    0.662     0.972     0.954
StrRep       vs Dem         |    0.2080    3.508    0.000     1.231     1.418
StrRep       vs Indep       |    0.2880    3.875    0.000     1.334     1.621
StrRep       vs Rep         |    0.2171    3.594    0.000     1.243     1.440
------------------------------------------------------------------------------
<output omitted>
```

The odds ratios can be plotted **mlogitplot**. For details, see Long and Freese.

Joint tests that all coefficients for a variable are 0

Testing that a variable has no effect requires test that *J*-1 coefficients are simultaneously zero. While this can be done **test**, SPost's **mlogtest** makes this simpler. The **set()** option requests tests that combine coefficients from multiple regressors. For example, **set(educ_set=2.educ 3.educ)** tests the coefficients from both **2.educ** and **3.educ**:

```
. mlogtest, set(educ_set=2.educ 3.educ)

Wald tests for independent variables (N=1382)

  Ho: All coefficients associated with given variable(s) are 0

                |      chi2    df   P>chi2
----------------+-------------------------
```

```
          age10 |    43.815     4    0.000
       income10 |    22.985     4    0.000
        1.black |    83.978     4    0.000
       1.female |     9.087     4    0.059
         2.educ |     5.569     4    0.234
         3.educ |    20.613     4    0.000
       educ_set |    26.188     8    0.001

   educ_set contains: 2.educ 3.educ
```

Test for combining outcomes

`test` can test if two outcomes can be combined. SPost's **`mlogtest, comb`** automatically runs **`test`** for all pairs of outcomes. I rarely run these tests since I usually have a good idea what outcome categories I want.

```
. mlogtest, comb

Wald tests for combining alternatives (N=1382)

  Ho: All coefficients except intercepts associated with a given pair
      of alternatives are 0 (i.e., alternatives can be combined)

                 |      chi2    df   P>chi2
-----------------+------------------------
    StrDem & Dem |    72.854     6    0.000
  StrDem & Indep |    40.334     6    0.000
    StrDem & Rep |   126.561     6    0.000
<output omitted>
```

#2+ Discrete change coefficients

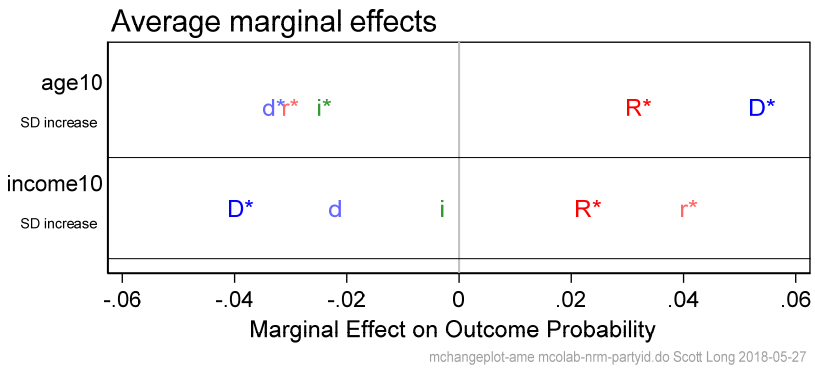
`mchange` without **`atmeans`** computes average effects. The option **`amount(sd)`** computes the DC for a standard deviation change:

```
. mchange age10 income10, amount(sd)

mlogit: Changes in Pr(y) | Number of obs = 1382

Expression: Pr(party), predict(outcome())

             |    StrDem        Dem      Indep        Rep     StrRep
-------------+-------------------------------------------------------
age10        |
         +SD |     0.054     -0.033     -0.024     -0.030      0.032
     p-value |     0.000      0.006      0.001      0.009      0.003
income10     |
         +SD |    -0.039     -0.022     -0.003      0.041      0.023
     p-value |     0.001      0.122      0.752      0.002      0.019
<output omitted>
```

Even with two variables, there is a lot of information to absorb that is easier to understand if plotted:

```
. global partycol "blue blue*.6 green*.8 red*.6 red"
. mchangeplot age10 income10, min(-.06) max(.06) gap(.02) ///
>     amount(sd sd) symbols(D d i r R) /// SD change is plotted
>     mcol($partycol) aspect(.3) sig(.05) ///
>     title(Average marginal effects, position(11))
```

The **`significance()`** option adds stars to indicate statistical significance.



#4+ probability plots

I use **mgen** to generate variables for predicted probabilities as a regressor changes:

```
. mgen, atmeans at(age10=(2(.5)8.5)) stub(MNLMage)

Predictions from: margins, atmeans at(age10=(2(.5)8.5)) predict(outcome())

Variable      Obs Unique      Mean       Min       Max  Label
-------------------------------------------------------------------------------
MNLMagepr1     14     14  .2046258  .1047786  .3256354  pr(y=StrDem) from mar...
MNLMagell1     14     14  .1675598  .0768825  .2487619  95% lower limit
MNLMageul1     14     14  .2416917  .1326748  .4025089  95% upper limit
MNLMageage10   14     14      5.25         2       8.5  Age in decades
MNLMageCpr1    14     14  .2046258  .1047786  .3256354  pr(y<=StrDem)

MNLMagepr2     14     14   .329003  .2561781  .3826132  pr(y=Dem) from margins
<output omitted>
```

Notice that there are variables for each outcome category (even though I only show some of them above). And, variables like **MNLMageCpr#** are cumulative probabilities that indicate the probability of being in a category less than or equal to #.

I create locals for the graph options:

```
local REGPnogap "plotregion(margin(zero) lcol(black) lwid(*.9))"
local yaxis_p "ytitle(Probability of party affiliation)"
local yaxis_p "`yaxis_p' ylab(0(.1).4, grid) ylin(0 .4, lcol(gs14))"
local titleopt "position(11) size(medium)"
local title "Multinomial logit model"
```

Options for the five lines are lengthy, so I build the local in steps:

```
local line5_opts "msym(O Oh dh sh s)"
local line5_opts "`line5_opts' lwid(*1.3 *1.3 *1.3 *1.3 *1.3)"
local line5_opts "`line5_opts' lpat(solid dash shortdash dash solid)"
local line5_opts "`line5_opts' mcol(red red*.8 black blue*.8 blue)"
local line5_opts "`line5_opts' lcol(red red*.8 black blue*.8 blue)"
```

While I could add labels by using options in **graph**, it is easier to create variable labels that will appear in the legend:

```
label var MNLMagepr1 "Strong Dem"
label var MNLMagepr2 "Democrat"
label var MNLMagepr3 "Independent"
label var MNLMagepr4 "Republican"
label var MNLMagepr5 "Strong Rep"
```

With these options, the command for the graph is simple:

```
graph twoway (connected MNLMagepr1 MNLMagepr2 MNLMagepr3 ///
    MNLMagepr4 MNLMagepr5 PLTage, `line5_opts'), ///
    title("`title'", `titleopt') `yaxis_p' `xaxis_age' ///
```

```
    `REGPnogap' legend(rows(2))
```

#50 Testing for IIA

I do not think there is a test for the IIA (independence of irrelevant alternatives) assumption that works well enough to be used. This example illustrates that the Small-Hsiao test is extremely sensitive to the arbitrary assumption of the seed for your random number generator. I start by choosing the random number generator used before Stata 15. If you are using Stata 14 or earlier, comment out the **`set rng`** command.

```
. set rng kiss32 // method used by Stata 14 and earlier
```

With my first seed, none of the tests are significant:

```
. set seed 124386
. mlogtest, smhsiao

Small-Hsiao tests of IIA assumption (N=1382)

  Ho: Odds(Outcome-J vs Outcome-K) are independent of other alternatives

                 | lnL(full)  lnL(omit)       chi2    df   P>chi2
-----------------+-----------------------------------------------
          StrDem |  -696.753   -690.654     12.198    21    0.934
             Dem |  -565.571   -557.488     16.166    21    0.760
           Indep |  -764.563   -758.290     12.547    21    0.924
             Rep |  -621.562   -615.492     12.140    21    0.936
          StrRep |  -761.598   -752.804     17.587    21    0.675

  Note: A significant test is evidence against Ho.
```

Using a different seed, 4 out of 5 are significant:

```
. set seed 254331
. mlogtest, smhsiao

Small-Hsiao tests of IIA assumption (N=1382)

  Ho: Odds(Outcome-J vs Outcome-K) are independent of other alternatives

                 | lnL(full)  lnL(omit)       chi2    df   P>chi2
-----------------+-----------------------------------------------
          StrDem |  -727.367   -692.048     70.639    21    0.000
             Dem |  -610.636   -573.268     74.736    21    0.000
           Indep |  -783.456   -747.654     71.604    21    0.000
             Rep |  -650.962   -615.434     71.057    21    0.000
          StrRep |  -751.887   -740.193     23.388    21    0.324

  Note: A significant test is evidence against Ho.
```

Exercise

The commands for estimating and plotting DCs and those for plotting predicted probabilities also work for ordinal logit or probit. Use the commands above to compare the model for party preference using **`mlogit`** and **`ologit`**.

Ordinal Outcomes: mcolab-orm-ordwarm.do

Just as the many commands from the last section can be used for ordinal models, commands in this section for tables of predictions can be use with **`mlogit`**. One of the powerful features of **`margins`** and commands based on **`margins`** is that the same commands work with many estimation commands. However, some SPost commands might not work with some estimation commands. If this occurs, try **`margins`** which is more general. And, let me know so I can update SPost.

#1 Load the data, fit the model, and save estimates

```
use ordwarm4, clear
tab warm, miss
codebook warm yr89 male white age ed prst, compact

ologit warm i.yr89 i.male i.white age ed prst, nolog

estimates store olm
```

#3+ Table of probabilities

Here examine how the probability of attitudes toward working women vary by the gender of the respondent and the year in which the question was asked.

Predicted probabilities

I compute predicted probabilities by gender and year with other variables at their means:

```
. mtable, at(yr89=(0 1) male=(0 1)) atmeans clear

Expression: Pr(warm), predict(outcome())

           |     yr89      male      1_SD       2_D       3_A      4_SA
 ----------+------------------------------------------------------------
         1 |        0         0     0.099     0.308     0.413     0.180
         2 |        0         1     0.186     0.403     0.316     0.095
         3 |        1         0     0.061     0.228     0.441     0.270
         4 |        1         1     0.119     0.339     0.390     0.151

Specified values of covariates

           |        1.
           |    white       age        ed      prst
 ----------+-------------------------------------
   Current |     .877      44.9      12.2      39.6
```

DC(male) by year

While I did not need p-values for probabilities (testing if the probability is 0 is not substantively interesting), I want to test if the effects of gender are significant so I add the option **`stats(est p)`**.

```
. mtable, dydx(male) at(yr89=(0 1)) atvars(1.yr89) atmeans stats(est p)

Expression: Marginal effect of Pr(warm), predict(outcome())

           |        1.
           |     yr89      1 SD       2 D       3 A      4 SA
 ----------+-------------------------------------------------
   d Pr(y) |        0     0.087     0.094    -0.097    -0.085
         p |        0     0.000     0.000     0.000     0.000
   d Pr(y) |        1     0.058     0.111    -0.050    -0.119
         p |        1     0.000     0.000     0.000     0.000
::
```

Did I do it right?

`margins` and related commands have many options. Until I am sure that my command is right, I try to compute everything multiple ways. Here I use **`mchange`** to confirm the results above.

```
. mchange male, at(yr89=0) atmeans brief

ologit: Changes in Pr(y) | Number of obs = 2293

Expression: Pr(warm), predict(outcome())
```

```
                |      1 SD        2 D        3 A       4 SA
----------------+-----------------------------------------
male            |
 Male vs Female |     0.087      0.094     -0.097     -0.085
        p-value |     0.000      0.000      0.000      0.000

. mchange male, at(yr89=1) atmeans brief
<output omitted>
```

Creating a nicer table

The results above have everything I want, but sometimes it is worth creating a nicer table. To do this, I started with predictions for men in 1977:

```
. mtable, at(yr89=0 male=1) atmeans rowname(Men) clear roweqnm(1977)

Expression: Pr(warm), predict(outcome())

           |     1_SD       2_D       3_A      4_SA
 ----------+-------------------------------------
 1977      |
       Men |    0.186     0.403     0.316     0.095
::
```

Note that **`rowname()`** and **`roweqnm()`** add labels to the table. Next I add results for women in 1977, but suppress printing the intermediate table using **`qui`** (for **`quietly`**):

```
. qui mtable, at(yr89=0 male=0) atmeans rowname(Women) below roweqnm(1977)
```

Using **`dydx()`** I compute the DCM(male|year):

```
. mtable, dydx(male) at(yr89=0) atmeans rowname(Men-Women) ///
>     below roweqnm(1977) stats(est p)

Expression: Marginal effect of Pr(warm), predict(outcome())

                    |     1 SD       2 D       3 A      4 SA
 -------------------+-------------------------------------
 1977               |
                Men |    0.186     0.403     0.316     0.095
              Women |    0.099     0.308     0.413     0.180
  Men-Women d Pr(y) |    0.087     0.094    -0.097    -0.085
        Men-Women p |    0.000     0.000     0.000     0.000
::
```

For 1989:

```
. qui mtable, at(yr89=1 male=1) atmeans rowname(Men) below roweqnm(1989)
. qui mtable, at(yr89=1 male=0) atmeans rowname(Women) below roweqnm(1989)
. qui mtable, dydx(male) at(yr89=1) atmeans rowname(Men_Women) ///
>     below roweqnm(1989) stats(est p)
```

Next I add the DC(year) for men and women:

```
. qui mtable, dydx(yr89) at(male=1) atmeans rowname(77to89) ///
>     below roweqnm(Men) stats(est p)
. mtable, dydx(yr89) at(male=0) atmeans rowname(77to89) ///
>     below roweqnm(Women) stats(est p)

Expression: Marginal effect of Pr(warm), predict(outcome())

                    |     1 SD       2 D       3 A      4 SA
 -------------------+-------------------------------------
 1977               |
                Men |    0.186     0.403     0.316     0.095
              Women |    0.099     0.308     0.413     0.180
  Men-Women d Pr(y) |    0.087     0.094    -0.097    -0.085
        Men-Women p |    0.000     0.000     0.000     0.000
 1989               |
```

```
              Men |    0.119     0.339     0.390     0.151
            Women |    0.061     0.228     0.441     0.270
  Men Women d Pr(y) |    0.058     0.111    -0.050    -0.119
        Men Women p |    0.000     0.000     0.000     0.000
 Men                |
     77to89 d Pr(y) |   -0.067    -0.063     0.074     0.056
           77to89 p |    0.000     0.000     0.000     0.000
 Women              |
     77to89 d Pr(y) |   -0.038    -0.080     0.028     0.090
           77to89 p |    0.000     0.000     0.000     0.000
::
```

This is close to what I would present in a paper.

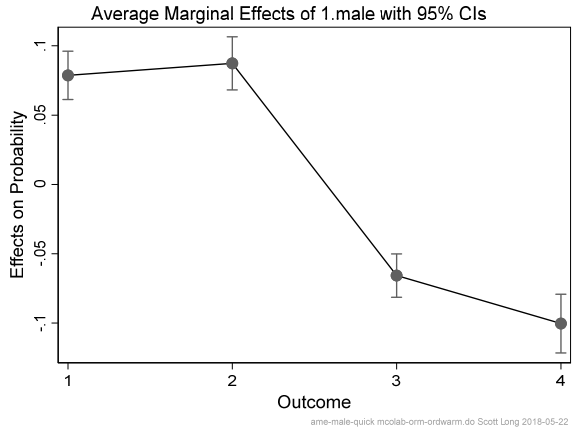
#4+ Plotting discrete changes

If I compute ADC(male) with **`margins,`** I can plot the effects with **`marginsplot.`**

```
. estimates restore olm
. margins, dydx(male)

Average marginal effects                        Number of obs     =      2,293
Model VCE    : OIM

dy/dx w.r.t. : 1.male
1._predict   : Pr(warm==1), predict(pr outcome(1))
2._predict   : Pr(warm==2), predict(pr outcome(2))
3._predict   : Pr(warm==3), predict(pr outcome(3))
4._predict   : Pr(warm==4), predict(pr outcome(4))

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
0.male       |  (base outcome)
-------------+----------------------------------------------------------------
1.male       |
    _predict |
           1 |   .0786721   .0088963     8.84   0.000     .0612356    .0961085
           2 |   .0873376   .0097506     8.96   0.000     .0682269    .1064484
           3 |   -.065683   .0079752    -8.24   0.000    -.0813142   -.0500519
           4 |  -.1003267   .0108098    -9.28   0.000    -.1215134   -.0791399
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

Without options, here is what I get:

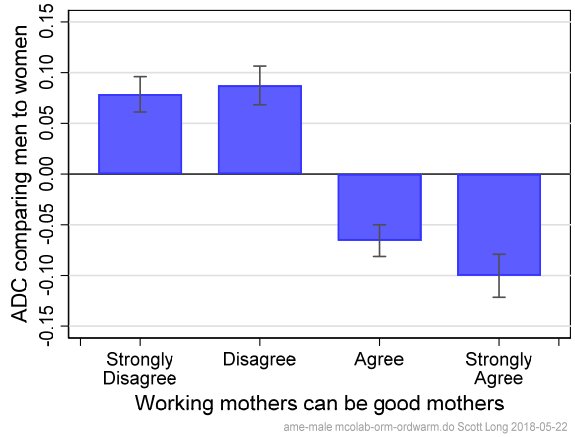
```
. marginsplot
```



Adding options:

```
. marginsplot, recast(bar) ciopts(color(gs5)) ///
```

```
>     xlab(0.5 " " 1 `""Strongly" "Disagree""' 2 "Disagree" ///
>          3 "Agree" 4 `""Strongly" "Agree""' 4.5 " ") ///
>     level(95) yline(0) plotopts(barw(0.7) color(blue*.8)) ///
>     ylab(-.15(.05).15, format(%4.2f) grid gmin gmax) ///
>     xtitle("Working mothers can be good mothers") ///
>     ytitle("ADC comparing men to women") scale(1.2) title("")
```



#45 Compare ADC(male|year) with 2nd differences

I want to know if the effect of being male has changed from 1977 to 1989. I start by computing the ADC's using **mchange**. This does not let me test if the effects are equal over time, but tells me if my later commands are computing what I want:

```
. mchange male, at(yr89=1)

ologit: Changes in Pr(y) | Number of obs = 2293

Expression: Pr(warm), predict(outcome())

                |      1 SD        2 D        3 A       4 SA
----------------+-------------------------------------------
male            |
 Male vs Female |     0.062      0.099     -0.041     -0.119
        p-value |     0.000      0.000      0.000      0.000
::

. mchange male, at(yr89=0)
::
```

Computing probabilities with margins

Here I compute four probabilities for each combination of year and gender which are used to compute discrete changes:

```
. margins, at(yr89=(0 1) male=(0 1)) post

Predictive margins                              Number of obs     =      2,293
Model VCE    : OIM

1._predict   : Pr(warm==1), predict(pr outcome(1))
2._predict   : Pr(warm==2), predict(pr outcome(2))
3._predict   : Pr(warm==3), predict(pr outcome(3))
4._predict   : Pr(warm==4), predict(pr outcome(4))

1._at        : yr89            =           0
               male            =           0

2._at        : yr89            =           0
               male            =           1
```

```
3._at         : yr89            =           1
                male            =           0

4._at         : yr89            =           1
                male            =           1

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
_predict#_at |
        1 1  |   .1085645   .0077007    14.10   0.000     .0934714    .1236576
        1 2  |   .1980111   .0116351    17.02   0.000     .1752067    .2208155
        1 3  |   .0680602   .0058309    11.67   0.000     .0566319    .0794886
        1 4  |   .1298218   .0099298    13.07   0.000     .1103597    .1492839
        2 1  |   .3032542   .0116638    26.00   0.000     .2803935     .326115
        2 2  |    .384566   .0120404    31.94   0.000     .3609673    .4081647
        2 3  |   .2313422   .0112666    20.53   0.000     .2092601    .2534243
        2 4  |   .3303547   .0125313    26.36   0.000     .3057938    .3549157
        3 1  |   .3963817   .0112756    35.15   0.000     .3742818    .4184815
        3 2  |   .3131865   .0111838    28.00   0.000     .2912666    .3351064
        3 3  |   .4188626   .0112497    37.23   0.000     .3968137    .4409116
        3 4  |   .3774672   .0120295    31.38   0.000     .3538897    .4010446
        4 1  |   .1917996   .0111773    17.16   0.000     .1698925    .2137068
        4 2  |   .1042364   .0074335    14.02   0.000     .0896671    .1188057
        4 3  |   .2817349   .0147241    19.13   0.000     .2528762    .3105937
        4 4  |   .1623563   .0111352    14.58   0.000     .1405316     .184181
------------------------------------------------------------------------------
```

Using lincom

With **lincom** I can compute a second difference of predicted probabilities to test if ADC(male|77) = ADC(male|89). To do this, I need to know the name, referred to as the legend, for each prediction:

```
. margins, coeflegend
::
             |     Margin  Legend
-------------+----------------------------------------------------------------
_predict#_at |
        1 1  |   .1085645  _b[1bn._predict#1bn._at]
        1 2  |   .1980111  _b[1bn._predict#2._at]
        1 3  |   .0680602  _b[1bn._predict#3._at]
        1 4  |   .1298218  _b[1bn._predict#4._at]
        2 1  |   .3032542  _b[2._predict#1bn._at]
        2 2  |    .384566  _b[2._predict#2._at]
        2 3  |   .2313422  _b[2._predict#3._at]
        2 4  |   .3303547  _b[2._predict#4._at]
        3 1  |   .3963817  _b[3._predict#1bn._at]
        3 2  |   .3131865  _b[3._predict#2._at]
        3 3  |   .4188626  _b[3._predict#3._at]
        3 4  |   .3774672  _b[3._predict#4._at]
        4 1  |   .1917996  _b[4._predict#1bn._at]
        4 2  |   .1042364  _b[4._predict#2._at]
        4 3  |   .2817349  _b[4._predict#3._at]
        4 4  |   .1623563  _b[4._predict#4._at]
------------------------------------------------------------------------------
```

Using these legends, I compute the difference:

```
. lincom (_b[1._predict#2._at]-_b[1._predict#1._at]) /// ADC male in 1977
>      - (_b[1._predict#4._at]-_b[1._predict#3._at]) //  ADC male in 1989

 ( 1)  - 1bn._predict#1bn._at + 1bn._predict#2._at + 1bn._predict#3._at -
       1bn._predict#4._at = 0

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
```

```
         (1) |    .027685   .0049492     5.59   0.000     .0179847    .0373854
------------------------------------------------------------------------------
```

This works, but I the command incredibly tedious. That is why **mlincom** was written for SPost. Instead of using the legend, you refer to predictions by number: 1 is the first prediction, 2 the second, and so on. Here I build a table with the results I want for outcome SD:

```
. mlincom (2-1), rowname(SD|77) stats(est p) clear

             |   lincom    pvalue
-------------+--------------------
       SD|77 |    0.089     0.000

. mlincom (4-3), rowname(SD|89) stats(est p) add

             |   lincom    pvalue
-------------+--------------------
       SD|77 |    0.089     0.000
       SD|89 |    0.062     0.000

. mlincom (2-1)-(4-3), rowname(Difference) stats(est p) add

             |   lincom    pvalue
-------------+--------------------
       SD|77 |    0.089     0.000
       SD|89 |    0.062     0.000
  Difference |    0.028     0.000
```

Continuing for other outcomes:

```
. qui mlincom (6-5), rowname(D|77) stats(est p) add
. qui mlincom (8-7), rowname(D|89) stats(est p) add
. qui mlincom (6-5)-(8-7), rowname(Difference) stats(est p) add
. qui mlincom (10-9), rowname(A|77) stats(est p) add
. qui mlincom (12-11), rowname(A|89) stats(est p) add
. qui mlincom (10-9)-(12-11), rowname(Difference) stats(est p) add
. qui mlincom (14-13), rowname(SA|77) stats(est p) add
. qui mlincom (16-15), rowname(SA|89) stats(est p) add
. mlincom (14-13)-(16-15), rowname(Difference) stats(est p) add
```

I get this table:

```
             |   lincom    pvalue
-------------+--------------------
       SD|77 |    0.089     0.000
       SD|89 |    0.062     0.000
  Difference |    0.028     0.000
        D|77 |    0.081     0.000
        D|89 |    0.099     0.000
  Difference |   -0.018     0.000
        A|77 |   -0.083     0.000
        A|89 |   -0.041     0.000
  Difference |   -0.042     0.000
       SA|77 |   -0.088     0.000
       SA|89 |   -0.119     0.000
  Difference |    0.032     0.000
```

Exercise

Use these commands with multinomial logit. After comparing the results, would you use multinomial logit or ordinal logit?

Generalized marginal effects: mcolab-gme.do

The commands in **`mcolab-gme.do`** build on what was been done in other sections of the guide. It extends those commands in ways that might not be obvious at first. By studying these examples, you learn a lot about the flexibility of **`margins`**. For me, one of the best ways to understand a command is to work my way through an example and figure it out on my own. While Long and Freese might have great explanations, you won't really understand the commands till you work through examples without having a detailed explanation. Try the commands in **`mcolab-gme.do`** one at a time. If you don't understand something, use **`help`**, review other examples, or build the command one step at a time. Finally, ask me!