Confidence Intervals for Predicted Outcomes in Regression Models for Categorical Outcomes

Jun Xu and J. Scott Long

Indiana University

August 23, 2005

1 Overview

The interpretation of regression models often involves the examination of predicted outcomes at specific values of the independent variables. The `prvalue` command by Long and Freese (forthcoming) makes it very simple to compute such post-estimation predictions. After estimating a model, the user specifies values of interest for the independent variables and `prvalue` computes the predicted value (or values) of the outcome. Changes in the predictions as values of the independent variables change can also be computed. In this article, we describe enhancements to `prvalue` that allow the computation of confidence intervals for both predicted outcomes and discrete changes in predictions. Depending on the regression model and options chosen, confidence intervals are computed using standard maximum likelihood methods, the method of endpoint transformations, the delta method, or bootstrapping. The command works after estimating models using `cloglog`, `cnreg`, `intreg`, `logit`, `mlogit`, `nbreg`, `ologit`, `oprobit`, `poisson`, `probit`, `regress`, `tobit`, `zinb`, and `zip`. To complement `prvalue`, we also present enhancements to `prgen` that allow you to plot confidence intervals and marginal changes.

2 Predictions and Methods for Computing Confidence Intervals

For a variety of regression models, `prvalue` computes predicted outcomes along with confidence intervals for these predictions. For models with categorical outcomes, the probability of each outcome is computed. For example, with the binary probit model the probability of observing a one is estimated as $\widehat{\pi} = \Phi\left(\mathbf{x}'\widehat{\boldsymbol{\beta}}\right)$, where $\mathbf{x}$ is a vector of independent variables and $\widehat{\boldsymbol{\beta}}$ contain the estimated parameters. We can also estimate a confidence interval for π. For a given confidence level, the upper and lower bounds define the confidence interval. For example, the 95% confidence interval for π includes upper and lower bounds such that

$$\Pr\left(\pi_{\mathrm{LB}} \leq \pi \leq \pi_{\mathrm{UB}}\right) = .95 \ .$$

To understand what this means, imagine that we take repeated samples from our population and that for each sample we estimate the upper and lower bounds of the confidence interval for that probability. About 95% of these confidence intervals would contain the true probability π.

For count models, both the predicted rate and the probability of each count are computed along with confidence intervals. For example, with the Poisson regression model the expected rate is $\widehat{\mu} = \exp\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right)$ and we compute the confidence interval such that $\Pr\left(\mu_{\rm LB} \leq \mu \leq \mu_{\rm UB}\right) = .95$.With count models, we also compute probabilities of specific values of the outcome. For example, we might compute the probability $\Pr\left(y = 2 \mid \mathbf{x}\right)$ of observing a count of two given the levels of the independent variables, with the confidence interval such that

$$\Pr\left[\Pr\left(y = 2 \mid \mathbf{x}\right)_{\rm LB} \leq \Pr\left(y = 2 \mid \mathbf{x}\right) \leq \Pr\left(y = 2 \mid \mathbf{x}\right)_{\rm UB}\right] = .95 \ .$$

The values computed for each model are summarized here:[1]

Models	$\Pr\left(y\right)$	y or y^*	Rate μ	$\Pr\left(\text{count}\right)$
`cloglog`, `logit`, `probit`	default	`ystar`[1]	no	no
`ologit`, `oprobit`	default	`ystar`[1]	no	no
`mlogit`,	default	no	no	no
`zinb`, `zip`, `nbreg`, `poisson`	no	no	default	default
`regress`, `cnreg`, `intreg`, `tobit`	no	default	no	no

1: when the `ystar` option is specified.

We can also compute changes in the predicted outcome as levels of the x's change. For example, suppose that we have two sets of values for the independent variables, $\mathbf{x}_A$ and $\mathbf{x}_B$. We could compute how the predicted outcome changes when the values of $\mathbf{x}$ change from $\mathbf{x}_A$ and $\mathbf{x}_B$. For example, for a binary model this would be

$$\frac{\Delta \Pr\left(y = 1\right)}{\Delta \mathbf{x}} = \Pr\left(y = 1 \mid \mathbf{x}_A\right) - \Pr\left(y = 1 \mid \mathbf{x}_B\right) \ ,$$

along with the confidence interval

$$\Pr\left[\frac{\Delta \Pr\left(y = 1\right)}{\Delta \mathbf{x}}_{\rm LB} \leq \frac{\Delta \Pr\left(y = 1\right)}{\Delta \mathbf{x}} \leq \frac{\Delta \Pr\left(y = 1\right)}{\Delta \mathbf{x}}_{\rm UB}\right] = .95 \ .$$

For further details, see Long and Freese (forthcoming).

`prvalue` uses four methods to compute confidence intervals: 1) maximum likelihood, 2) endpoint transformation, 3) the bootstrap method, and 4) the delta method. While the technical details on each method for are given in Section 7, here we provide general information about each method.

1. **Maximum Likelihood** For models such as the linear regression model, the standard method of computing confidence intervals using maximum likelihood theory is used.

2. **Endpoint Transformation** For binary models, this method computes the upper and lower bounds of the confidence intervals for the linear combination $\mathbf{x}'\boldsymbol{\beta}$ and then

[1]With Stata 9, `prvalue` can also compute predicted outcomes for the models estimated by `mprobit`, `slogit`, `ztp` and `ztnb`. However, confidence intervals are not available for these commands.

transforms these bounds into the probability of observing a one. For example for probit, the lower bound $(\mathbf{x}'\boldsymbol{\beta})_{\mathrm{LB}}$ would be transformed into the lower bound for the probability as $\Pr(y=1)_{\mathrm{LB}} = \Phi[(\mathbf{x}'\boldsymbol{\beta})_{\mathrm{LB}}]$. One advantage of this method is that the bounds cannot be smaller than 0 or greater than 1. This method cannot be used for computing confidence intervals for changes in predictions.

3. **Delta Method** This method takes a function that is too complex for analytically computing the variance (for example, the change in the predicted probabilities in a multinomial logit model) and creates a linear approximation of that function. The variance of the simpler approximation is used for constructing the confidence interval. Since `prvalue` uses analytic formulas for the derivatives, rather than numerical estimation, the computation of confidence intervals is extremely fast. Unlike the method of endpoint transformation, the bounds computed by the delta method can include values that exceed the range of the statistic being estimated (e.g., a bound for a predicted probability could be negative or greater than one).

4. **Bootstrap Method** The idea of the bootstrap (see Guan 2003 for an introduction to the bootstrap using Stata) is that by taking repeated samples from the sample used to estimate your model, you can estimate the sampling variability that would occur by taking repeated samples from the population. This is done by taking a random sample from the estimation sample, computing the statistics of interest, and repeating this for some number of replications. The variation in the estimates across the replications is used to estimate the standard deviation of the sampling distribution. While the bootstrap method generally works quite well and avoids assumptions implicit in other methods, it is computationally intensive. For example, computing the confidence intervals for a multinomial logit with 5 outcomes, three variables, and 337 cases by the delta method took .15 seconds, while computing the confidence intervals by bootstrap took 141 seconds using 1,000 replications. Computations of confidence intervals for a multinomial logit with 6 outcomes, 4 variables and 7,357 cases using the delta method took .61 seconds, while computing the confidence intervals by bootstrap took 13 minutes 2 seconds for 1,000 replications.[2] Roughly speaking, each replication in a bootstrap takes as long as the entire computation for the delta method. The `zip` and `zinb` models are too complex for computing the derivatives necessary for the delta method, so only the bootstrap method is available.

A summary of which methods work for which models is shown below:

[2] Computations were made using Stata/SE 8.2 (born 10 Jan 2005) on a Dell XPS with an Intel® Pentium® 4 running at 3.4GHz. Estimates of times for the delta method were based on the average of 1,000 computations.

Models	Maximum Likelihood (ML)	Endpoint Transformations	Delta Method	Bootstrap Method	Default Method
`cloglog`, `logit`, `probit`	yes[1]	yes[2]	yes	yes	delta
`ologit`, `oprobit`	yes[1]	no	yes	yes	delta
`mlogit`, `nbreg`, `poisson`	no	no	yes	yes	delta
`zinb`, `zip`	no	no	no	yes	no ci
`regress`, `cnreg`, `intreg`, `tobit`	ystar	no	no	no	ml

1: when the `ystar` option is specified; 2: this method does not work for changes in predictions.

We now discuss how to install `prvalue` and `prgen`, followed by a discussion of the syntax and options for each command.

3 Installation

`prvalue` and `prgen` are part of the SPost package of programs for postestimation analysis (see Long and Freese forthcoming and www.indiana.edu/~jslsoc/spost.htm for details). Installation should be simple if you are connected to the Internet. If you have installed a prior version of SPost, we suggest that you begin by uninstalling it. To do this, enter the command `ado`. This will list all packages that have been installed, with each package marked with a number in [ ]'s. Scan the list and record the number of the package containing the SPost ado files and any other packages related to SPost. While the ado files for Stata 9 are contained in the package `spost_ado9`, other names were used in the past. Uninstall these packages with the command `ado uninstall [#]`, where # is the package number (note that you must include the brackets, for example, `ado uninstall [3]`). You must run `ado uninstall` once for each package to be uninstalled.

Next, type `search spost, net` to get a list of packages related to SPost. One of these packages should be labeled `spost_ado9 from http://www.indiana.edu/~jslsoc/stata`. This contains the commands for Stata 9 and later. The package for Stata 8 is labeled `spostado from http://www.indiana.edu/~jslsoc/stata`. Click on the blue label for your version of Stata and you will be given information on how to install the programs. To install examples on using `prvalue` and `prgen`, click on the label
`spostci from http://www.indiana.edu/~jslsoc/stata`. These do and dta files will reproduce the examples in this article and will include other examples of using these commands.

If you have do files that used commands from prior versions of SPost, they should continue to work as before. The only exception is that now `prvalue, ystar` does not print predicted probabilities. To compute the predicted probabilities, simply run the command without the `ystar` option. The formatting of output has also changed to incorporate the confidence intervals.

4 Syntax for prvalue

prvalue [if *exp*] [in *range*] [, x(*variable1*=*value1*[...]) rest(*stat*) all save diff maxcnt(#) brief nobase nolabel ystar level(#) [delta | ept | bootstrap] reps(#) dots match size(#) saving(...)[biascorrected | percentile | normal]

Options

x(*variable1*=*value1*[...]) sets the values of independent variables for calculating predicted outcomes. The list must alternate variable names and either numeric values or terms describing the value to use. mean, median, min, and max are used to specify the mean, median, minimum value or maximum value for a variable. upper sets a variables to its minimum or maximum depending on which value yields the larger predicted value; lower sets a variables to its minimum or maximum depending on which value yields the smaller predicted value. If prvalue has already been run, previous will set variables to their prior values. For example, x(wc=1 age=median) or x(wc=1 hc=previous age=mean age=40).

rest(*stat*) sets the independent variables not specified by x() to their mean, min, max, or median when calculating predicted values. grmean sets these independent variables to the mean conditional on the variables and values specified in x(); grmedian, grmax, and grmin work in the same way using values for the median, maximum or minimum. If rest() is not specified, rest(mean) is assumed.

if *exp* and in *range* specify the sample used to compute the statistics (e.g., mean) used to set the values with x() and rest().

all specifies that calculation of means, medians, and other statistics should use the entire sample instead of the possibly smaller sample used to estimate the model.

save saves information from the current prvalue for use in computing changes in predictions using the diff option.

diff computes difference between current predictions and those that were saved using save. You must use the same method for computing confidence intervals with diff as was used for the save results.

ystar requests that the predicted value of y^* rather than predicted probabilities should be computed for binary and ordinal models.

maxcnt(#) is the maximum count value for which the probability is computed in count models. The default of 9 will list probabilities for values from 0 to 9.

brief prints only limited output showing predictions and confidence intervals, without listing values of the independent variables.

nobase suppresses the listing of the base values that were specified with x() or rest().

nolabel uses the numeric values of the outcome rather than value labels in the output.

Options for computing confidence intervals

`level(#)` specifies the confidence level, in percent, for confidence intervals. For example, `level(95)` specifies that you want a 95% confidence interval.

One of the following options can be chosen to specify the method used to compute confidence intervals:

`ept` computes confidence intervals for predicted probabilities for `cloglog`, `logit` and `probit` by endpoint transformation. This method cannot be used for changes in predictions.

`delta` calculates confidence intervals by the delta method using analytic derivatives.

`bootstrap` computes confidence intervals using the bootstrap method. This method takes roughly 1,000 times longer to compute than other methods.

Options used for bootstrapped confidence intervals

`reps(#)` specifies the number of bootstrap replications to be performed. The default is 1,000. The accuracy of a bootstrap estimate depends critically on the number of replications. While sources differ on the recommended number of replications, Efron and Tibshirani (1993:188) suggest 1,000 replications for confidence intervals. You can use `bssize` (Poi 2004) to calculate the number of bootstrap replications to be used. In our experience, this method suggests over 1,000 replications. See the documentation for `saving` below.

`dots` is used with `bootstrap` to write a . at the beginning of each replication and periodically prints the percent of total replications that has been completed. If computations appears to be stuck (i.e., new dots do not appear), it is likely that the estimation is not converging for the current bootstrap sample. We have found this to be most common with `zip` and `zinb`. When this happens, you can click on the break symbol to stop computations for the current sample or wait until the maximum number of iterations have been computed (by default, the maximum number of iterations is 16,000). When a model does not converge for a given bootstrap sample, that sample is dropped.

`match` specifies that the bootstrap will resample within each category of the dependent variable in proportion to the distribution of the outcome categories in the estimation sample. If `match` is not specified, the proportions in each category of the bootstrap sample are determined entirely by the random draw and it is possible to have samples with no cases in some of the categories. This option does not apply to `cnreg`, `intreg`, `nbreg`, `poisson`, `regress`, `tobit`, `zinb`, and `zip`.

`size(#)` specifies the number of cases to be sampled when bootstrapping. The default is the size of the estimation sample. If `size(#)` is specified, # must be less than or equal to the size of the estimation sample. In general, it is best to not specify `size` (see www.stata.com/support/faqs/stat/reps.html for further information).

`saving(`*filename, save_ options*`)` creates a data file with the estimates from each of the bootstrapped samples with one case for each replication. This option is useful when you need to examine the distribution of bootstrapped estimates. For example, this option is required if you plan to use `bssize` to calculate the number of replications to be used (Poi 2004).

One of the following can be chosen if you are computing bootstrapped confidence intervals:

`percentile` computes the bootstrapped confidence interval using the percentile method. This is the default method.

`biascorrected` computes the bootstrapped confidence interval using the bias-corrected method.

`normal` computes the bootstrapped confidence interval using the normal approximation method.

5 Syntax for prgen

`prgen` *varname* [`if` *exp*] [`in` *range*]`,` `generate(`*prefix*`)` [`from(#)` `to(#)` `ncases(#)` `gap(#)` `x(`*variable1=value1* [...]`)` `rest(`*stat*`)` `maxcnt(#)` `brief` `all` `noisily` `marginal` `ci`][options from `prvalue`]

All options from `prvalue`, except for `save` and `diff`, can be used to specify how the upper and lower bounds are created when the `ci` option is used. The options `if`, `in`, `x()`, `rest()`, `maxcnt`, `brief` and `all` work in the same way as for `prvalue`.

Options

varname is the name of the variable that changes while all other variables are held at specified values.

`generate(`*prefix*`)` sets the prefix for the new variables created by `prgen`. Choosing a prefix that is different than the beginning letters of any of the variables in your data set makes it easier to examine the results. For example, if you choose the prefix `abcd` then you can use the command `sum abcd*` to examine all newly created variables.

`from(#)` and `to(#)` are the start and end values for *varname*. The default is for *varname* to range from the observed minimum to the observed maximum of *varname*.

`ncases(#)` specifies the number of predicted values `prgen` computes as *varname* varies from the start value to the end value. The default is 11.

`gap(#)` is an alternative to `ncases`. You specify the gap or size between tic marks and `prgen` determines if the specified value divides evenly into the range specified with `from` and `to`. If it does, `prgen` determines the appropriate value for `ncases`.

`ci` indicates that you want to generate confidence intervals corresponding to the predictions that are created.

`marginal` requests that a variable or variables are created containing the marginal change in the outcome relative to *varname*, holding all other variables constant.

`noisily` indicates that you want to see the output from `prvalue` that was used to generate the predicted values.

Variables generated The following table indicates the variables that are created by `prgen`. The observations contain predicted values or probabilities for a range of values for the variable *varname*, holding the other variables at the specified values. n observations are created, where n is 11 by default or can be specified by `ncases()`or `gap()`. The new variable names start with the *prefix* specified by `gen()`. The variables created are:

For which models	Name	Content
All models	*prefix*`x`	Values of *varname*.
`logit`, `probit`	*prefix*`p0`	Predicted probability $\Pr(y = 0)$.
	prefix`p1`	Predicted probability $\Pr(y = 1)$.
`ologit`, `oprobit`	*prefix*`p`k	Predicted probability $\Pr(y = k)$ for all outcomes.
	prefix`s`k	Cumulative probability $\Pr(y \leq k)$ for all outcomes.
`mlogit`	*prefix*`p`k	Predicted probability $\Pr(y = k)$ for all outcomes.
`poisson`, `nbreg`, `zip`, `zinb`	*prefix*`mu`	Predicted rate μ.
	prefix`p`k	Predicted probability $\Pr(y = k)$, for $0 \leq k \leq$ `maxcnt()`.
	prefix`s`k	Cumulative probability $\Pr(y \leq k)$, for $0 \leq k \leq$ `maxcnt()`.
`zip`, `zinb`	*prefix*`inf`	Predicted probability $\Pr(\text{Always } 0 = 1) = \Pr(\textit{inflate})$.
`regress`, `tobit`, `cnreg`, `intreg`	*prefix*`xb`	Predicted value $\mathbf{x}\widehat{\boldsymbol{\beta}}$.

If `ci` is specified as an option for `prgen`, variables are created containing the upper and lower bounds for the confidence interval for the outcome. These variables have the same names as those in the table above, except for adding `ub` at the end for the variable with the upper bound and `lb` for the lower bound. If `marginal` is specified, variables are created that contain the marginal change in the outcome with respect to *varname*, holding all other variables constant. The variables containing marginals have the same names as those in the table above, except for adding a `D` prior to the outcome abbreviation and `D`*varname* after. For example, the marginal for *prefix*`p0` is named *prefix*`Dp0D`*varname.* Marginals are only computed for those models for which the `prchange` command can compute the marginal change (see Long and Freese forthcoming).

6 Examples

The examples that follow illustrate increasingly complex ways in which `prvalue` can be used. In addition to these examples, other examples are downloaded when you install the `spostci` package. For further details on the data and models, see either Long (1997) or Long and Freese (forthcoming).

6.1 Simple predictions

The commands generating the output in this section are in `prvalue_predict.do` in the `spostci` package.

Binary logit In this example, we use a binary logit model to predict labor force participation for a sample of women. The independent variables are the number of children less than six, the number of children from six to 18, the wife's age, whether the wife went to college, whether the husband went to college, the log of the wife's estimated wages, and the family income excluding wife's income.

```
. use binlfp2, clear
. logit lfp k5 k618 age wc hc lwg inc, nolog
```

(*output omitted*)

After the model is estimated, we compute the predicted probability of labor force participation for a woman who is 35 years old (`age=35`), has two young children (`k5=2`), did not attend college (`wc=0`), and is average on all other characteristics (`rest(mean)`):

```
. prvalue, x(age=35 k5=2 wc=0) rest(mean)

logit: Predictions for lfp

Confidence intervals by delta method

                              95% Conf. Interval
  Pr(y=inLF|x):       0.1174   [ 0.0495,    0.1852]
  Pr(y=NotInLF|x):    0.8826   [ 0.8148,    0.9505]

           k5       k618        age         wc         hc        lwg        inc
x=          2  1.3532537         35          0  .39176627  1.0971148  20.128965
```

For someone with these characteristics (which are listed in the line beginning `x=`), the predicted probability of being in the labor force is .117, with a 95% confidence interval from .050 to .185.

With binary and ordinal models, we can also predict the latent y^* used in the latent variable formulation of the model. To do this, we add the option `ystar`:

```
. prvalue , x(age=35 k5=2 wc=0) rest(mean) ystar

logit: Predictions for lfp

                              95% Conf. Interval
  Predicted y*:      -2.0177   [-2.6723,   -1.3631]

           k5       k618        age         wc         hc        lwg        inc
x=          2  1.3532537         35          0  .39176627  1.0971148  20.128965
```

Negative binomial regression In this example, we examine a count model predicting the number of papers published by biochemists. Independent variables are whether the scientist is a woman, whether the scientist is married, the number of young children in the scientist's family, the prestige of scientist's Ph.D. department and the number of articles published by the scientist's mentor.

```
. use couart2, clear
. nbreg art fem mar kid5 phd ment, nolog
```

(*output omitted*)

Since very few scientists publish more than six papers, we use the option `maxcnt(6)` so that `prvalue` will only compute predicted probabilities for publications from 0 to 6. For a single (`mar=0`) woman (`fem=0`) without young children (`kid5=0`) who is average on all other characteristics, we find:

```
. prvalue , x(mar=0 fem=1 kid5=0) rest(mean) maxcnt(6)

nbreg: Predictions for art

Confidence intervals by delta method

                              95% Conf. Interval
  Rate:               1.4079   [ 1.2237,    1.5921]
  Pr(y=0|x):          0.3346   [ 0.2966,    0.3726]
  Pr(y=1|x):          0.2905   [ 0.2809,    0.3000]
  Pr(y=2|x):          0.1818   [ 0.1731,    0.1904]
  Pr(y=3|x):          0.0991   [ 0.0863,    0.1118]
  Pr(y=4|x):          0.0500   [ 0.0395,    0.0604]
  Pr(y=5|x):          0.0240   [ 0.0170,    0.0310]
  Pr(y=6|x):          0.0111   [ 0.0070,    0.0153]

          fem        mar       kid5        phd       ment
x=          1          0          0  3.1031093  8.7672131
```

We predict 1.41 publications, with a confidence interval from 1.22 to 1.59. Below the rate, the probabilities that a person with these characteristics will publish a given number of papers are listed along with confidence intervals computed with the delta method.

Predictions at observed values `prvalue` is designed to compute predictions at values specified by the user (e.g., `x(k5=0)`) or at values corresponding to summary statistics in the sample (e.g., `rest(mean)`). If you want to compute predictions along with confidence intervals at observed values for a given observation, you need to explicitly indicate those values. For example, the first observation in the data used with `nbreg` is:

```
. list in 1
```

```
     +-----------------------------------------+
     | art   fem       mar   kid5    phd   ment |
     |-----------------------------------------|
  1. |   0   Men   Married      0   2.52      7 |
     +-----------------------------------------+
```

To compute the predicted outcomes at these values, we can use the command:

```
. prvalue, x(fem=0 mar=1 kid5=0 phd=2.52 ment=7)

nbreg: Predictions for art

Confidence intervals by delta method

                               95% Conf. Interval
  Rate:                 1.913   [ 1.6587,    2.1673]
  Pr(y=0|x):           0.2499   [ 0.2155,    0.2843]
  Pr(y=1|x):           0.2591   [ 0.2421,    0.2762]
  Pr(y=2|x):           0.1937   [ 0.1925,    0.1949]
  Pr(y=3|x):           0.1261   [ 0.1162,    0.1360]
  Pr(y=4|x):           0.0760   [ 0.0646,    0.0874]
  Pr(y=5|x):           0.0436   [ 0.0339,    0.0533]
  Pr(y=6|x):           0.0242   [ 0.0171,    0.0313]
  Pr(y=7|x):           0.0131   [ 0.0083,    0.0179]
  Pr(y=8|x):           0.0069   [ 0.0039,    0.0100]
  Pr(y=9|x):           0.0036   [ 0.0018,    0.0055]

     fem   mar  kid5   phd  ment
x=     0     1     0  2.52     7
```

Using a bit of simple Stata programming, it is possible to automate this process to compute predictions for all observations. This is shown in the sample do file `pravlue_observed.do` that is installed as part of the `spostci` package.

6.2 Discrete change

The commands generating the output in this section are in `prvalue_change.do` in the `spostci` package.

Binary probit One way to interpret the results of regression type models is to see how the predictions change when levels of the independent variables change. To illustrate this, we examine how the probability of being in the labor force is expected to increase when a woman has gone to college. We begin by estimating a probit model:

```
. use binlfp2, clear
. probit lfp k5 k618 age wc hc lwg inc, nolog
```

(*output omitted*)

The next command predicts the probability for someone who is average on all characteristics and who has not gone to college (`wc=0`). Notice that this command is prefixed by `quietly` since we do not want to see the results yet. But, we save the results with the `save` option.

```
. quietly prvalue, x(wc=0) rest(mean) save
```

We run `prvalue` again, this time for someone who attended college (`wc=1`). We use the `diff` option to compute changes in predictions from the saved results:

```
. prvalue, x(wc=1) rest(mean) diff

probit: Change in Predictions for lfp

Confidence intervals by delta method

                     Current     Saved    Change   95% CI for Change
  Pr(y=inLF|x):       0.7082    0.5238    0.1844  [ 0.0892,   0.2795]
  Pr(y=NotInLF|x):    0.2918    0.4762   -0.1844  [-0.2795,  -0.0892]

                k5       k618        age         wc         hc        lwg
Current=   .2377158  1.3532537  42.537849          1  .39176627  1.0971148
  Saved=   .2377158  1.3532537  42.537849          0  .39176627  1.0971148
   Diff=          0          0          0          1          0          0

               inc
Current=  20.128965
  Saved=  20.128965
   Diff=          0
```

For someone who is average on all other variables, attending college increases the probability of being in the labor force by .18 with a 95% confidence interval from .09 to .28.

Poisson regression Our next example computes the change in the rate of publication when two variables change at the same time. We also illustrate the computation of bootstrapped confidence intervals.

```
. use couart2, clear
. poisson art fem mar kid5 phd ment, nolog
```

(*output omitted*)

We want to compare the predicted productivity for an unmarried woman without children to the productivity for a married, female scientist with two young children:

```
. quietly prvalue , x(mar=0 fem=1 kid5=0) maxcnt(5) boot save
```

```
. prvalue , x(mar=1 fem=1 kid5=2) maxcnt(5) boot diff

poisson: Change in Predictions for art

Bootstrapped confidence intervals using percentile method
(1000 of 1000 replications completed)

                    Current      Saved     Change   95% CI for Change
  Rate:               1.138     1.4102   -.27228   [-0.5110,  -0.0300]
  Pr(y=0|x):         0.3205     0.2441    0.0764   [ 0.0080,   0.1535]
  Pr(y=1|x):         0.3647     0.3442    0.0205   [ 0.0022,   0.0384]
  Pr(y=2|x):         0.2075     0.2427   -0.0352   [-0.0728,  -0.0034]
  Pr(y=3|x):         0.0787     0.1141   -0.0354   [-0.0652,  -0.0039]
  Pr(y=4|x):         0.0224     0.0402   -0.0178   [-0.0327,  -0.0020]
  Pr(y=5|x):         0.0051     0.0113   -0.0062   [-0.0121,  -0.0007]

                fem        mar       kid5        phd       ment
Current=          1          1          2  3.1031093  8.7672131
  Saved=          1          0          0  3.1031093  8.7672131
   Diff=          0          1          2          0          0
```

By default, 1,000 replications are computed, all of which completed successfully as indicated by the count of the number of replications completed.

6.3 Plotting confidence intervals

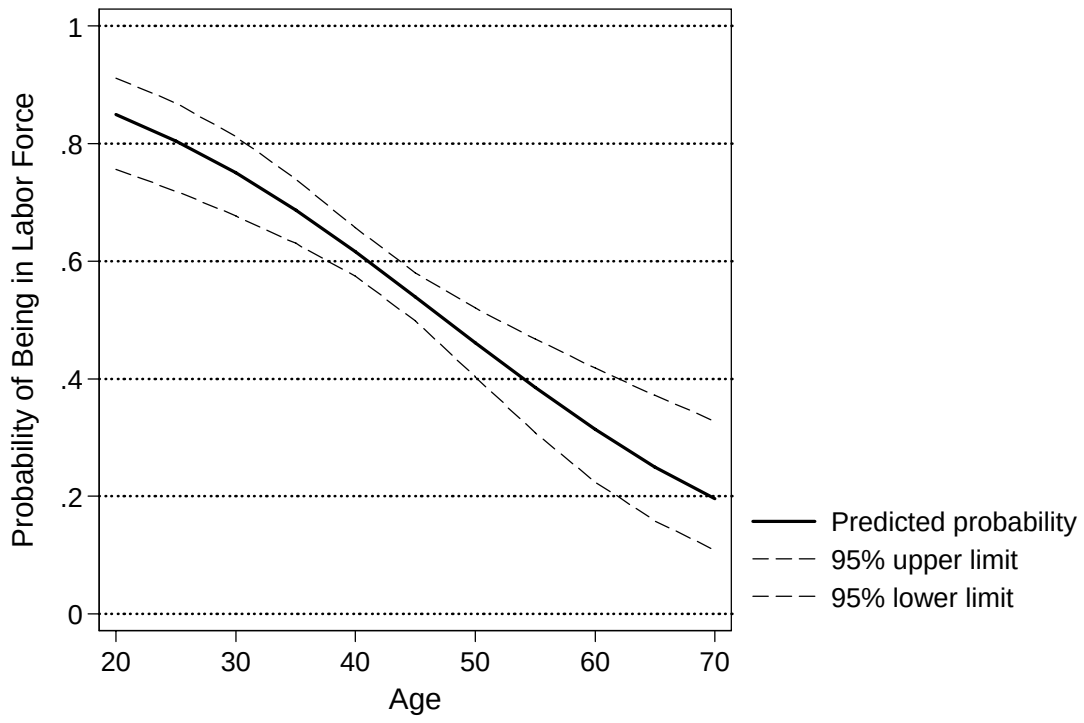
The commands generating the output in this section are in `prgen_plotpred.do` in the `spostci` package.

Predicted probabilities with logit `prgen` makes it simple to plot predictions and confidence bands for those predictions as one variable changes, holding all other variables constant. First, we estimate a binary logit model:

```
. use binlfp2, clear
. logit lfp k5 k618 age wc hc lwg inc, nolog
```

(*output omitted*)

We want to plot the predicted probability of being in the labor force for average women at various ages. The resulting plot looks like this:



As would be expected, the probability of being in the labor force decreases with age and the confidence interval narrows at the center of our data. To create this graph, we use `prgen` to compute predictions at many different values of `age` and to store the predictions. The command that follows includes several options governing the confidence interval. First, `ci` tells `prgen` that you want to save values for plotting the confidence interval. The `ept` option indicates that the confidence interval should be computed using endpoint transformations, rather than the default which is the delta method. Finally, `noisily` indicates that you want to see the results of `prvalue` each time it is used to compute a prediction and its confidence interval; in practice, you would only use `noisily` if you were having problems getting the results to converge. Using `prgen`, we now compute predictions as `age` ranges from 20 to 70 in steps of five years:

```
. prgen age, from(20) to(70) gap(5) gen(prlfp) ci ept noisily

Results from prvalue called by prgen

logit: Predictions for lfp

                                  95% Conf. Interval
  Pr(y=inLF|x):       0.8495   [ 0.7565,    0.9111]
  Pr(y=NotInLF|x):    0.1505   [ 0.0889,    0.2435]

           k5       k618        age         wc         hc        lwg        inc
x=   .2377158  1.3532537         20   .2815405  .39176627  1.0971148  20.128965
```

(*output omitted*)

```
logit: Predicted values as age varies from 20 to 70.

           k5       k618        age         wc         hc        lwg        inc
x=   .2377158  1.3532537  42.537849   .2815405  .39176627  1.0971148  20.128965
```

The variables that are created all begin with the prefix `prlfp`:

```
. desc prlfp*

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
prlfpx          float  %9.0g                  Changing value of age
prlfpp0         float  %9.0g                  pr(NotInLF)=Pr(0)
prlfpp1         float  %9.0g                  pr(inLF)=Pr(1)
prlfpp0ub       float  %9.0g                  UB pr(NotInLF)=Pr(0)
prlfpp1ub       float  %9.0g                  UB pr(inLF)=Pr(1)
prlfpp0lb       float  %9.0g                  LB pr(NotInLF)=Pr(0)
prlfpp1lb       float  %9.0g                  LB pr(inLF)=Pr(1)
```

`prlfpx` contains values of `age` between 20 and 70. `prlfpp1` is the probability of being in the labor force, while `prlfpp1ub` and `prlfpp1lb` hold the upper and lower bounds. We can plot these with the following commands, which generated the graph above. First, we create variable labels that are used to label the plot:

```
. label var prlfpp1 "Predicted probability"
. label var prlfpp1ub "95% upper limit"
. label var prlfpp1lb "95% lower limit"
. label var prlfpx "Age"
```

Next, we use `twoway` to plot the results:

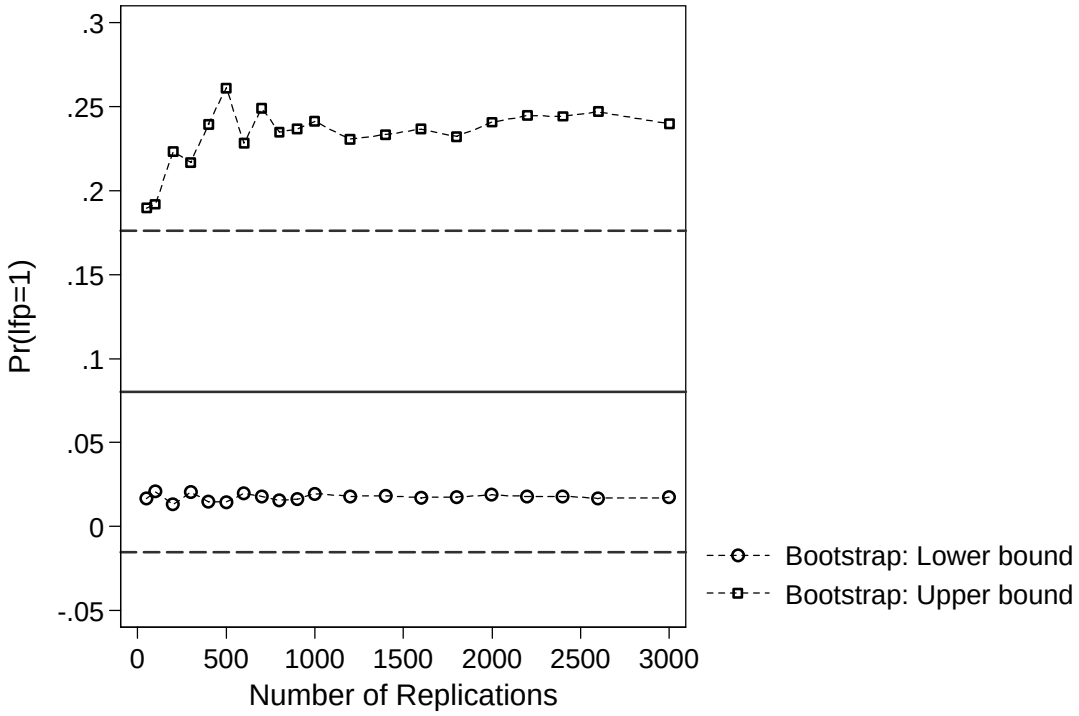
```
. twoway ///
>     (connected prlfpp1 prlfpx, clcolor(black) clpat(solid) ///
>         clwidth(medthick) msymbol(i) mcolor(none))         ///
>     (connected prlfpp1ub prlfpx, msymbol(i) mcolor(none)   ///
>         clcolor(black) clpat(dash) clwidth(thin))          ///
>     (connected prlfpp1lb prlfpx, msymbol(i) mcolor(none)   ///
>         clcolor(black) clpat(dash) clwidth(thin)),         ///
>     ytitle("Probability of Being in Labor Force")          ///
>     yscale(range(0 .35)) ylabel(, grid glwidth(medium)     ///
>     glpattern(dot)) xscale(range(20 70)) xlabel(20(10)70)
```

6.4 Effects of the number of replications

The number of replications is important for the quality of the bootstrapped confidence interval. To illustrate this, we consider our earlier example of labor force participation:

```
use binlfp2, clear
logit lfp k5 k618 age wc hc lwg inc, nolog
```

For this model, we compute the 95% confidence interval using the bootstrap method with the number of replications varying from 50 to 3,000. This leads to the following graph, which shows that the upper and lower bounds from the bootstrap vary substantially when the number of replications is less than 1,000 (especially for the upper bounds), but stabilize after 1,000.



Here is how the graph was generated. First, we compute the predicted probability of being in the labor force with `inc` set to 100 and other variables held at the mean, with the 95% confidence interval computed by the delta method:

```
prvalue, x(inc=100) rest(mean) delta
local PrLFP = pepred[2,2]
local UpperDelta = peupper[2,2]
local LowerDelta = pelower[2,2]
```

The values of predicted probability along with the upper and lower bounds are saved to the local variables `PrLFP`, `UpperDelta` and `LowerDelta` that are used to plot the sold line for the predicted probability at 0.080 and the dashed lines for the lower and upper bounds at -0.015 and 0.176 in the graph above). Next, we we create variables that will hold the values for

the number of bootstrap replications used and the bounds computed using a given number of replications. These variables are plotted to create our graph:

```
gen reps = .
    label var reps "# of replications"
gen ubboot = .
    label var ubboot "Bootstrap: Upper bound"
gen lbboot = .
    label var lbboot "Bootstrap: Lower bound"
```

Before computing the first bootstrapped confidence interval, we set the random number seed so that we can reproduce our results later (without this, a new sequence of random numbers are generated each time we run the program). We then loop through values for the number of replications ranging from 50 to 3000. Within the loop, we compute bootstrapped confidence intervals and save the values into the variables we created above:

```
set seed 2399194
local j = 0
foreach reps in 50 100 200 300 400 500 600 700 800 900 1000 ///
        1200 1400 1600 1800 2000 2200 2400 2600 3000 {
    local ++j
    qui replace reps = `reps' if _n==`j'
    di "= Start for `reps' replications: " c(current_time)
    di "= Seed: " c(seed)
    prvalue , x(inc=100) rest(mean) boot rep(`reps')
    scalar UpperBoot = peupper[2,2]
    qui replace ubboot = UpperBoot if _n==`j'
    scalar LowerBoot = pelower[2,2]
    qui replace lbboot = LowerBoot if _n==`j'
    di
    di "= End for `reps' replications: " c(current_time)
    di
}
```

The results were then plotted. Consistent with the recommendations given above, we see that the bounds begin to stabilize at around 1,000 replications, which is our default. The commands generating the output in this section are in `prvalue_boot_reps.do`.

7 Methods for Computing Confidence Intervals

In this section we provide a more technical discussion of the ideas behind the methods used by `prvalue` to compute confidence intervals. Let δ be some parameter estimated by your model (e.g., $\Pr(y = 1)$ in the logit model). We are interested in computing the lower and upper bounds such that $\Pr(\text{LB} \leq \delta \leq \text{UB}) = \alpha$ .This can be interpreted as saying that the population parameter δ is contained within the interval with confidence $1 - \alpha$. In the simplest case, the confidence interval can be computed directly using results from

maximum likelihood theory. Under the usual conditions for ML, $\widehat{\boldsymbol{\beta}} \overset{a}{\sim} N\left(\boldsymbol{\beta}, Var\left(\widehat{\boldsymbol{\beta}}\right)\right)$. Then, $\mathbf{x}\widehat{\boldsymbol{\beta}} \overset{a}{\sim} N\left(\mathbf{x}'\boldsymbol{\beta}, \mathbf{x}' Var\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{x}\right)$, which can be used to compute confidence intervals for linear combinations $\mathbf{x}\widehat{\boldsymbol{\beta}}$, such as $\widehat{y}$ in linear regression. For example, to compute the $100\left(1-\alpha\right)$ confidence interval, define z as the $\left(1-\frac{\alpha}{2}\right)$ percentile from a standard normal distribution (i.e., the probability of being greater than z is $\alpha/2$ and the probability of being less than $-z$ is $\alpha/2$). Since $\mathbf{x}\widehat{\boldsymbol{\beta}}$ is asymptotically normal with variance $\mathbf{x}' Var\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{x}$, the $100\left(1-\alpha\right)$ confidence interval for $\mathbf{x}'\boldsymbol{\beta}$ is

$$\left(\mathbf{x}\widehat{\boldsymbol{\beta}} - z\sqrt{\mathbf{x}' Var\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{x}}\right) \leq \mathbf{x}'\boldsymbol{\beta} \leq \left(\mathbf{x}\widehat{\boldsymbol{\beta}} + z\sqrt{\mathbf{x}' Var\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{x}}\right) \ . \tag{1}$$

Since many of the quantities computed by `prvalue` are nonlinear transformations of $\mathbf{x}'\boldsymbol{\beta}$, other methods are required.

7.1 Endpoint Transformations

The method of endpoint transformation can compute confidence intervals for monotonic functions of $\mathbf{x}'\boldsymbol{\beta}$ (i.e., a function that is always increasing or always decreasing), such as the predicted probabilities in binary logit or probit. First, the confidence interval for $\mathbf{x}\widehat{\boldsymbol{\beta}}$ is computed as the symmetric interval $[\mathrm{LB}_{\mathbf{x}'\boldsymbol{\beta}} \leq \mathbf{x}'\boldsymbol{\beta} \leq \mathrm{UB}_{\mathbf{x}'\boldsymbol{\beta}}]$. Since $\Pr\left(\mathbf{x}'\boldsymbol{\beta}\right) = F\left(\mathbf{x}'\boldsymbol{\beta}\right)$ is a monotonic transformation of $\mathbf{x}'\boldsymbol{\beta}$, the confidence interval for $F\left(\mathbf{x}'\boldsymbol{\beta}\right)$ is computed by transforming the bounds using the same function F:

$$\left[\Pr\left(\mathbf{x}'\boldsymbol{\beta}\right)_{\mathrm{LB}} \leq \Pr\left(\mathbf{x}'\boldsymbol{\beta}\right) \leq \Pr\left(\mathbf{x}'\boldsymbol{\beta}\right)_{\mathrm{UB}}\right] = \left[F\left(\left[\mathbf{x}'\boldsymbol{\beta}\right]_{\mathrm{LB}}\right) \leq F\left(\mathbf{x}'\boldsymbol{\beta}\right) \leq F\left(\left[\mathbf{x}'\boldsymbol{\beta}\right]_{\mathrm{UB}}\right)\right] \ .$$

For example, consider the binary logit model. Using equation 1, we compute the $100\left(1-\alpha\right)$ confidence interval for $\mathbf{x}\widehat{\boldsymbol{\beta}}$. To compute the predicted probability, we take the logit transformation of $\mathbf{x}\widehat{\boldsymbol{\beta}}$:

$$\Pr\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right) = \frac{\exp\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right)}{1+\exp\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right)} = \Lambda\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right) \ ,$$

where $\Lambda\left(.\right)$ is the cdf for the logistic distribution. By applying the logit transformation to the endpoints from equation 1, we obtain the asymmetric confidence interval for $\Pr\left(\mathbf{x}'\boldsymbol{\beta}\right)$:

$$\Lambda\left(\mathbf{x}\widehat{\boldsymbol{\beta}} - z\sqrt{\mathbf{x}' Var\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{x}}\right) \leq \Pr\left(\mathbf{x}'\boldsymbol{\beta}\right) \leq \Lambda\left(\mathbf{x}\widehat{\boldsymbol{\beta}} + z\sqrt{\mathbf{x}' Var\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{x}}\right) \ .$$

While computationally simple, this method is limited to those few cases in which the outcome of interest is a monotonic function of $\mathbf{x}'\boldsymbol{\beta}$. For further discussion, see Cox and Ma (1995) and Liao (2000).

7.2 Delta Method

The delta method is a more general method for computing confidence intervals. This method takes a function that is too complex for analytically computing the variance (e.g., $Var\left(\exp\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right)\left[1+\exp\left(\mathbf{x}\widehat{\boldsymbol{\beta}}\right)\right]^{-1}\right)$, creates a linear approximation of the function, and then computes the variance of the simpler linear function that is used for large sample inference. While we illustrate this approach with a simple, one-parameter example, the approach generalizes readily to the case with multiple parameters. Details on the equations used to implement the delta method for the models in `prvalue` are available at www.indiana.edu\~jslsoc\stata\spostci\spost_deltaci.pdf.

Let $F(x\beta)$ be the estimator of interest, for example, $F(x\beta) = \Pr(x\beta) = \Phi(x\beta)$ where Φ is the cumulative density function for the standard, normal distribution. The first step is to use a Taylor expansion to linearize the function evaluated at $\widehat{\beta}$:

$$F\left(x\widehat{\beta}\right) \approx F(x\beta) + (\widehat{\beta}-\beta)f(x\beta)\ ,$$

where $f(\beta) = F'(\beta)$ is the derivative of F evaluated at β. Then, we take the variance of both sides of the equation:

$$Var\left[F\left(x\widehat{\beta}\right)\right] \approx Var\left[F(x\beta) + (\widehat{\beta}-\beta)f(x\beta)\right]\ .$$

We can easily simplify the right-hand side:

$$\begin{aligned} Var\left[F(x\beta) + (\widehat{\beta}-\beta)f(x\beta)\right] &= Var\left[F(x\beta)\right] + Var\left[(\widehat{\beta}-\beta)f(x\beta)\right] \\ &\quad + 2Cov\left[F(x\beta), (\widehat{\beta}-\beta)f(x\beta)\right] \\ &= 0 + Var\left[(\widehat{\beta}-\beta)f(x\beta)\right] + 0 \\ &= \left[f(\beta)\right]^2 Var\left(\widehat{\beta}-\beta\right) \\ &= \left[f(\beta)\right]^2 Var\left(\widehat{\beta}\right)\ , \end{aligned}$$

where we use the fact that β, $f(x\beta)$, and $F(x\beta)$ are constants.

To make our example concrete, consider binary probit where $\Pr\left(x\widehat{\beta}\right) = \Phi\left(x\widehat{\beta}\right)$ and x is any specific value. The linear expansion is:

$$\Phi\left(x\widehat{\beta}\right) \approx \Phi(x\beta) + \left(\widehat{\beta}-\beta\right)\frac{\partial\Phi(x\beta)}{\partial\beta}\ , \tag{2}$$

where

$$\frac{\partial\Phi(x\beta)}{\partial\beta} = x\phi(\beta x)\ .$$

Then,

$$Var\left[\Phi(x\beta) + \left(\widehat{\beta}-\beta\right)\phi(x\beta)\right] = \left[x\phi(\beta x)\right]^2 Var\left(\widehat{\beta}\right)\ ,$$

which leads to the symmetric confidence interval

$$\left[\Pr\left(x\widehat{\beta}\right) - z\sqrt{\left[x\phi\left(x\widehat{\beta}\right)\right]^2 Var\left(\widehat{\beta}\right)}\right] \leq \Pr\left(x\beta\right) \leq \left[\Pr\left(x\widehat{\beta}\right) + z\sqrt{\left[x\phi\left(x\widehat{\beta}\right)\right]^2 Var\left(\widehat{\beta}\right)}\right] .$$

Unlike the asymmetric confidence interval based on end-point transformations, this confidence interval could include values less than 0 or greater than 1.

Next, consider a discrete change $\Pr\left(x_a\widehat{\beta}\right) - \Pr\left(x_b\widehat{\beta}\right) = \Phi\left(x_a\widehat{\beta}\right) - \Phi\left(x_b\widehat{\beta}\right)$, where x_a and x_b are two values of x. The linearization is

$$\Phi\left(x_a\widehat{\beta}\right) - \Phi\left(x_b\widehat{\beta}\right) \approx \left[\Phi\left(\beta x_a\right) - \Phi\left(\beta x_b\right)\right] + \left(\widehat{\beta} - \beta\right)\frac{\partial\left[\Phi\left(x_a\beta\right) - \Phi\left(x_b\beta\right)\right]}{\partial\beta} .$$

Taking the variance of the right-hand-side and simplifying:

$$\begin{aligned}
&Var\left(\left[\Phi\left(x_a\beta\right) - \Phi\left(x_b\beta\right)\right] + \left(\widehat{\beta} - \beta\right)\frac{\partial\left[\Phi\left(x_a\beta\right) - \Phi\left(x_b\beta\right)\right]}{\partial\beta}\right) \\
&= Var\left[\left(\widehat{\beta} - \beta\right)\frac{\partial\left[\Phi\left(x_a\beta\right) - \Phi\left(x_b\beta\right)\right]}{\partial\beta}\right] \\
&= \left(\frac{\partial\left[\Phi\left(x_a\beta\right) - \Phi\left(x_b\beta\right)\right]}{\partial\beta}\right)^2 Var\left(\widehat{\beta}\right) \\
&= \left[x_a^2\phi\left(x_a\beta\right)^2 + x_b^2\phi\left(x_b\beta\right)^2 - 2x_a\phi\left(x_a\beta\right)x_b\phi\left(x_b\beta\right)\right] Var\left(\widehat{\beta}\right) .
\end{aligned}$$

To evaluate it, we simply replace β with $\widehat{\beta}$.

7.3 Bootstrap

The bootstrap is a computationally expensive, nonparametric technique for making statistical inferences. The bootstrap method allows us to approximate the variation of parameter estimates (or function of these estimates). For a basic introduction with specific applications using Stata, we recommend Poi (2004) and Guan (2003). For a thorough treatment of this method, see Efron and Tibshirani (1986) and Mooney and Duval (1993). Here we present only the most basic information.

To compute bootstrap confidence intervals for predicted outcome, the following steps are taken:

1. From the original estimation sample, draw a simple random sample of size N with replacement. This is called a resample. Using the resample, estimate the model and compute the quantities of interest.

2. Repeat step 1 R times and collect the estimates from each subsample. Use the R estimates to create an empirical probability distribution of the quantities of interest. This distribution, known as the bootstrap distribution, is used to construct the confidence interval. Essentially, the variation in estimates among the resamples is used to estimate the standard error of the sample estimate.

We use three methods for computing confidence intervals from the R empirical estimates of each parameter. The *normal method* assumes that the bootstrap distribution is approximately normal and uses the standard deviation from the bootstrap distribution to compute the appropriate percentiles from a normal distribution. The *percentile method* determines the α and $1-\alpha$ percentile from the bootstrap distribution without any assumption about the shape of that distribution. With this method, the bounds cannot exceed possible values for the statistic in question. The *bias-corrected method* adjusts for bias between the predicted probabilities and the average of simulated predicted probabilities. By default, `prvalue` uses the percentile method.

8 Saved Results

All results that might be useful for Monte Carlo simulations, plotting, or other post-estimation analysis are saved to global strings or matrices. While the typical user will not need this information, it is needed when you are writing programs that use the results computed by `prvalue`. Additional information can be obtained with `help _pecollect`.

petype is a global string with the type of model being estimated. The string contains three words. The first word contains `e(cmd)` for the model being analyzed. Word 2 classifies the type of model as either `typical` for all model except `zip` and `zinb` which are classified as `twoeq`. Word 3 indicates the general type of outcome and is one of the words: `binary`, `count`, `mlogit`, `ordered`, `regress`, or `tobit`.

pecimethod is a global string indicating the type of confidence interval that was computed. The first word is `ml`, `delta`, `ept`, or `bootstrap`. The second word indicates how the bootstrap confidence intervals were computed, either `normal`, `percentile` or `biascorrected`.

peinfo is a 3×11 matrix with information about the model and options used with `prvalue`. Row 1 contains information on the current call to `prvalue`. Row 2 contains information on `prvalue` for the model last saved with the `save` option. Row 3 contains information on the differences between the current and saved information. Normally, rows 1 and 2 have identical information. Columns contain the following information:

Column 1: number of right-hand-side variables.
Column 2: number of categories in the outcome.
Column 3: level for confidence interval (e.g., 95 not .95).
Column 4: z-value for confidence interval at given level.
Column 5: number of right-hand-side variables for inflation in `zip` and `zinb`.
Column 6: 1 if model has no constant, else 0.
Column 7: base category for `mlogit`.
Column 8: `stdp` for binary models.
Column 9: number of requested replications for bootstrap (i.e., the number specified by the `rep` option).
Column 10: number of completed replications for bootstrap. When a estimates cannot be computed for a given bootstrap sample, it is not counted.

Column 11: value of the maximum number of values of predicted probabilities in count models, corresponding to the `maxcnt` option.

pebase and pebase2 contain the base values for the x's in computing the predictions. The jth column of `pebase` is the jth right-hand-side variable in the model. The jth column of `pebase2` is the jth right-hand-side inflation variable in `zip` or `zinb`. If `save` and `dif` are used, the three rows are in the matrix correspond to: 1) the current model; 2) the saved model; and 3) differences between the current and save values.

pepred contains the predictions computed by `prvalue`. This matrix has 7 rows. I has one column for each outcome, with a minimum of 4 columns. Rows contain the following information.

Row 1: values of the outcome category (e.g., 0, 1, 2).
Row 2: predicted probabilities for the value in row 1 for the current model.
Row 3: predictions other than probabilities for the current model.
Row 4: predicted probabilities for the value in row 1 for the saved model.
Row 5: predictions other than probabilities for the saved model.
Row 6: the difference between rows 2 and 4.
Row 7: the difference between rows 3 and 5.

For rows 2, 4 and 6, the columns contained predicted probabilities corresponding to the categories in row 1. Rows 3, 5 and 7 contain predictions for other quantities (where not all columns are used for all models). Column 1 contains $\mathbf{x}'\widehat{\boldsymbol{\beta}}$ from the first part of model. Column 2 contains $\widehat{\mu}$ for count models. Column 3: $\mathbf{x}'\widehat{\boldsymbol{\beta}}$ from the inflation part of `zip` and `zinb`. Column 4: $\widehat{\Pr}$ (always 0) for `zip` and `zinb`.

peupper and pelower contains the lower and upper confidence limits corresponding to the information in `pepred`.

peuppct, pelopct, peupbias, pelobias, peupnorm and pelonorm are created when the bootstrap method is used. They contain the upper and lower limits for all three methods of computing confidence intervals: percentile, bias-corrected, and normal approximation. Whichever method is selected as an option in the `prvalue` command (e.g., `prvalue, boot normal`) is also contained in `peupper` and `pelower`. The information in each matrix corresponds to the information in `pepred`.

9 References

Cox, Christopher and Guangqin Ma. 1995. "Asymptotic Confidence Bands for Generalized Nonlinear Regression Models." *Biometrics* 51:142-50.

Efron, Bradley and R. Tibshirani. 1986. "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy." *Statistical Science* 1:54-75.

Efron, Bradley and Robert Tibshirani. 1993. *An introduction to the bootstrap*. New York: Chapman & Hall.

Guan, Weihua. 2003. From the help desk: Bootstrapped standard errors. *The Stata Journal* 3:71-80.

Liao, Tim Futing. 2000. "Estimated Precision for Predictions from Generalized Linear Models in Sociological Research." *Quality & Quantity* 34:137-52.

Long, J. Scott. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: SAGE Publications.

Long, Scott and Jeremy Freese. Forthcoming. *Regression Models for Categorical Dependent Variables Using Stata*. Second Edition. College Station, Texas: Stata Corporation.

Mooney, Christopher Z. and Robert D. Duval. 1993. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Newbury Park: SAGE Publications.

Poi, Brian P. 2004. From the help desk: Some bootstrapping techniques. *The Stata Journal* 4:312-328.

File xulong-prvalue-23Aug2005.tex