**glm — Generalized linear models**

### Description

`glm` fits generalized linear models. It can fit models by using either IRLS (maximum quasilikelihood) or Newton–Raphson (maximum likelihood) optimization, which is the default.

See [U] 27 Overview of Stata estimation commands for a description of all of Stata's estimation commands, several of which fit models that can also be fit using `glm`.

### Quick start

Model of \( y \) as a function of \( x \) when \( y \) is a proportion

\[
glm y x, \text{family(binomial)}
\]

Logit model of \( y \) events occurring in 15 trials as a function of \( x \)

\[
glm y x, \text{family(binomial 15) link(logit)}
\]

Probit model of \( y \) events as a function of \( x \) using grouped data with group sizes \( n \)

\[
glm y x, \text{family(binomial n) link(probit)}
\]

Model of discrete \( y \) with user-defined family `myfamily` and link `mylink`

\[
glm y x, \text{family(myfamily) link(mylink)}
\]

Bootstrap standard errors in a model of \( y \) as a function of \( x \) with a gamma family and log link

\[
glm y x, \text{family(gamma) link(log) vce(bootstrap)}
\]

### Menu

Statistics > Generalized linear models > Generalized linear models (GLM)
Syntax

```
glm depvar [ indepvars ] [ if ] [ in ] [ weight ] [ , options ]
```

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td></td>
</tr>
<tr>
<td><code>family(familyname)</code></td>
<td>distribution of depvar; default is family(gaussian)</td>
</tr>
<tr>
<td><code>link(linkname)</code></td>
<td>link function; default is canonical link for family() specified</td>
</tr>
<tr>
<td><strong>Model 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>noconstant</code></td>
<td>suppress constant term</td>
</tr>
<tr>
<td><code>exposure(varname)</code></td>
<td>include ln(varname) in model with coefficient constrained to 1</td>
</tr>
<tr>
<td><code>offset(varname)</code></td>
<td>include varname in model with coefficient constrained to 1</td>
</tr>
<tr>
<td><code>constraints(constraints)</code></td>
<td>apply specified linear constraints</td>
</tr>
<tr>
<td><code>asis</code></td>
<td>retain perfect predictor variables</td>
</tr>
<tr>
<td><code>mu(varname)</code></td>
<td>use varname as the initial estimate for the mean of depvar</td>
</tr>
<tr>
<td><code>init(varname)</code></td>
<td>synonym for mu(varname)</td>
</tr>
<tr>
<td><strong>SE/Robust</strong></td>
<td></td>
</tr>
<tr>
<td><code>vce(vcetype)</code></td>
<td>vcetype may be oim, robust, cluster clustvar, eim, opg, bootstrap, jackknife, hac kernel, jackknife1, or unbiased</td>
</tr>
<tr>
<td><code>vfactor(#)</code></td>
<td>multiply variance matrix by scalar #</td>
</tr>
<tr>
<td><code>disp(#)</code></td>
<td>quasilikelihood multiplier</td>
</tr>
<tr>
<td>`scale(x2</td>
<td>dev</td>
</tr>
<tr>
<td><strong>Reporting</strong></td>
<td></td>
</tr>
<tr>
<td><code>level(#)</code></td>
<td>set confidence level; default is level(95)</td>
</tr>
<tr>
<td><code>eform</code></td>
<td>report exponentiated coefficients</td>
</tr>
<tr>
<td><code>nocnsreport</code></td>
<td>do not display constraints</td>
</tr>
<tr>
<td><code>display_options</code></td>
<td>control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling</td>
</tr>
<tr>
<td><strong>Maximization</strong></td>
<td></td>
</tr>
<tr>
<td><code>ml</code></td>
<td>use maximum likelihood optimization; the default</td>
</tr>
<tr>
<td><code>irls</code></td>
<td>use iterated, reweighted least-squares optimization of the deviance</td>
</tr>
<tr>
<td><code>maximize_options</code></td>
<td>control the maximization process; seldom used</td>
</tr>
<tr>
<td><code>fisher(#)</code></td>
<td>use the Fisher scoring Hessian or expected information matrix (EIM)</td>
</tr>
<tr>
<td><code>search</code></td>
<td>search for good starting values</td>
</tr>
<tr>
<td><code>noheader</code></td>
<td>suppress header table from above coefficient table</td>
</tr>
<tr>
<td><code>notable</code></td>
<td>suppress coefficient table</td>
</tr>
<tr>
<td><code>nodisplay</code></td>
<td>suppress the output; iteration log is still displayed</td>
</tr>
<tr>
<td><code>collinear</code></td>
<td>keep collinear variables</td>
</tr>
<tr>
<td><code>coeflegend</code></td>
<td>display legend instead of statistics</td>
</tr>
</tbody>
</table>
### Options

- **family**(*familyname*) specifies the distribution of *depvar*; *family*(gaussian) is the default.
- **link**(*linkname*) specifies the link function; the default is the canonical link for the *family()* specified (except for *family*(nbinomial)).
- **noconstant**, **exposure**(varname), **offset**(varname), **constraints**(constraints); see [R] Estimation options. **constraints**(constraints) is not allowed with irls.
asis forces retention of perfect predictor variables and their associated, perfectly predicted observations and may produce instabilities in maximization; see [R] probit. This option is allowed only with option family(binomial) with a denominator of 1.

\texttt{mu(varname)} specifies \textit{varname} as the initial estimate for the mean of \textit{depvar}. This option can be useful with models that experience convergence difficulties, such as family(binomial) models with power or odds-power links. \texttt{init(varname)} is a synonym.

\texttt{vce(vcetype)} specifies the type of standard error reported, which includes types that are derived from asymptotic theory (\texttt{oim}, \texttt{opg}), that are robust to some kinds of misspecification (\texttt{robust}), that allow for intragroup correlation (\texttt{cluster clustvar}), and that use bootstrap or jackknife methods (\texttt{bootstrap}, \texttt{jackknife}); see [R] \texttt{vce option}.

In addition to the standard \texttt{vcetypes}, \texttt{glm} allows the following alternatives:

\texttt{vce(eim)} specifies that the EIM estimate of variance be used.

\texttt{vce(jackknife1)} specifies that the one-step jackknife estimate of variance be used.

\texttt{vce(hac kernel [#])} specifies that a heteroskedasticity- and autocorrelation-consistent (HAC) variance estimate be used. HAC refers to the general form for combining weighted matrices to form the variance estimate. There are three kernels built into \texttt{glm}. \texttt{kernel} is a user-written program or one of \texttt{nwest | gallant | anderson}

# specifies the number of lags. If # is not specified, \( N - 2 \) is assumed. If you wish to specify \texttt{vce(hac ...)}, you must \texttt{tsset} your data before calling \texttt{glm}.

\texttt{vce(unbiased)} specifies that the unbiased sandwich estimate of variance be used.

\texttt{vfactor(\#)} specifies a scalar by which to multiply the resulting variance matrix. This option allows you to match output with other packages, which may apply degrees of freedom or other small-sample corrections to estimates of variance.

\texttt{disp(\#)} multiplies the variance of \textit{depvar} by \# and divides the deviance by \#. The resulting distributions are members of the quasilikelihood family. This option is allowed only with option \texttt{irls}.

\texttt{scale(x2 | dev | \#)} overrides the default scale parameter. This option is allowed only with Hessain (information matrix) variance estimates.

By default, \texttt{scale(1)} is assumed for the discrete distributions (binomial, Poisson, and negative binomial), and \texttt{scale(x2)} is assumed for the continuous distributions (Gaussian, gamma, and inverse Gaussian).

\texttt{scale(x2)} specifies that the scale parameter be set to the Pearson \( \chi^2 \) (or generalized \( \chi^2 \)) statistic divided by the residual degrees of freedom, which is recommended by McCullagh and Nelder (1989) as a good general choice for continuous distributions.

\texttt{scale(dev)} sets the scale parameter to the deviance divided by the residual degrees of freedom. This option provides an alternative to \texttt{scale(x2)} for continuous distributions and overdispersed or underdispersed discrete distributions. This option is allowed only with option \texttt{irls}.

\texttt{scale(\#)} sets the scale parameter to \#. For example, using \texttt{scale(1)} in family(gamma) models results in exponential-errors regression. Additional use of link(log) rather than the default link(power -1) for family(gamma) essentially reproduces Stata’s \texttt{streg}, dist(exp) nohr command (see [ST] \texttt{streg}) if all the observations are uncensored.
Reporting

level(#)\text{\textbackslash;} see \texttt{[R] Estimation options}.

eform displays the exponentiated coefficients and corresponding standard errors and confidence intervals. For family(binomial) link(logit) (that is, logistic regression), exponentiation results are odds ratios; for family(nbinomial) link(log) (that is, negative binomial regression) and for family(poisson) link(log) (that is, Poisson regression), exponentiated coefficients are incidence-rate ratios.

nocnsreport; see \texttt{[R] Estimation options}.

display\texttt{\_options}: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(style), cformat(\%fmt), pformat(\%fmt), sformat(\%fmt), and nolstretch; see \texttt{[R] Estimation options}.

Maximization

\texttt{ml} requests that optimization be carried out using Stata’s \texttt{ml} commands and is the default.

\texttt{irls} requests iterated, reweighted least-squares (IRLS) optimization of the deviance instead of Newton–Raphson optimization of the log likelihood. If the \texttt{irls} option is not specified, the optimization is carried out using Stata’s \texttt{ml} commands, in which case all options of \texttt{ml maximize} are also available.

\texttt{maximize\_options}: difficult, technique(algorithm\_spec), iterate(#), \texttt{[no]}log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and \texttt{from}(init\_specs); see \texttt{[R] Maximize}. These options are seldom used.

Setting the optimization method to technique(bhhh) resets the default \texttt{vcetype} to \texttt{vce(opg)}.

If option \texttt{irls} is specified, only \texttt{maximize\_options} iterate(), nolog, trace, and ltolerance() are allowed. With \texttt{irls} specified, the convergence criterion is satisfied when the absolute change in deviance from one iteration to the next is less than or equal to ltolerance(), where ltolerance(1e-6) is the default.

\texttt{fisher(#)} specifies the number of Newton–Raphson steps that should use the Fisher scoring Hessian or EIM before switching to the observed information matrix (OIM). This option is useful only for Newton–Raphson optimization (and not when using \texttt{irls}).

\texttt{search} specifies that the command search for good starting values. This option is useful only for Newton–Raphson optimization (and not when using \texttt{irls}).

The following options are available with \texttt{glm} but are not shown in the dialog box:

\texttt{noheader} suppresses the header information from the output. The coefficient table is still displayed.

\texttt{notable} suppresses the table of coefficients from the output. The header information is still displayed.

\texttt{nodisplay} suppresses the output. The iteration log is still displayed.

\texttt{collinear}, \texttt{coeflegend}; see \texttt{[R] Estimation options}. \texttt{collinear} is not allowed with \texttt{irls}.

Remarks and examples

Remarks are presented under the following headings:

\texttt{General use}
\texttt{Variance estimators}
\texttt{User-defined functions}
**General use**

`glm` fits generalized linear models of `y` with covariates `x`:

\[
g\{E(y)\} = x\beta, \quad y \sim F
\]

`g()` is called the link function, and `F` is the distributional family. Substituting various definitions for `g()` and `F` results in a surprising array of models. For instance, if `y` is distributed as Gaussian (normal) and `g()` is the identity function, we have

\[
E(y) = x\beta, \quad y \sim \text{Normal}
\]
or linear regression. If `g()` is the logit function and `y` is distributed as Bernoulli, we have

\[
\logit\{E(y)\} = x\beta, \quad y \sim \text{Bernoulli}
\]
or logistic regression. If `g()` is the natural log function and `y` is distributed as Poisson, we have

\[
\ln\{E(y)\} = x\beta, \quad y \sim \text{Poisson}
\]
or Poisson regression, also known as the log-linear model. Other combinations are possible.

Although `glm` can be used to perform linear regression (and, in fact, does so by default), this regression should be viewed as an instructional feature; `regress` produces such estimates more quickly, and many postestimation commands are available to explore the adequacy of the fit; see [R] `regress` and [R] `regress postestimation`.

In any case, you specify the link function by using the `link()` option and specify the distributional family by using `family()`. The available link functions are

<table>
<thead>
<tr>
<th>Link function</th>
<th><code>glm</code> option</th>
</tr>
</thead>
<tbody>
<tr>
<td>identity</td>
<td><code>link(identity)</code></td>
</tr>
<tr>
<td>log</td>
<td><code>link(log)</code></td>
</tr>
<tr>
<td>logit</td>
<td><code>link(logit)</code></td>
</tr>
<tr>
<td>probit</td>
<td><code>link(probit)</code></td>
</tr>
<tr>
<td>complementary log–log</td>
<td><code>link(cloglog)</code></td>
</tr>
<tr>
<td>odds power</td>
<td><code>link(opower #)</code></td>
</tr>
<tr>
<td>power</td>
<td><code>link(power #)</code></td>
</tr>
<tr>
<td>negative binomial</td>
<td><code>link(nbinomial)</code></td>
</tr>
<tr>
<td>log–log</td>
<td><code>link(loglog)</code></td>
</tr>
<tr>
<td>log-complement</td>
<td><code>link(logc)</code></td>
</tr>
</tbody>
</table>

Define \( \mu = E(y) \) and \( \eta = g(\mu) \), meaning that \( g(\cdot) \) maps \( E(y) \) to \( \eta = x\beta + \text{offset} \).
Link functions are defined as follows:

- **identity** is defined as \( \eta = g(\mu) = \mu \).
- **log** is defined as \( \eta = \ln(\mu) \).
- **logit** is defined as \( \eta = \ln\left(\frac{\mu}{1 - \mu}\right) \), the natural log of the odds.
- **probit** is defined as \( \eta = \Phi^{-1}(\mu) \), where \( \Phi^{-1}(\cdot) \) is the inverse Gaussian cumulative.
- **cloglog** is defined as \( \eta = \ln\left\{ -\ln\left(1 - \mu\right) \right\} \).
- **opower** is defined as \( \eta = \left[\left(\frac{\mu}{1 - \mu}\right)^{n} - 1\right]/n \), the power of the odds. The function is generalized so that \( \text{link(opower 0)} \) is equivalent to \( \text{link(logit)} \), the natural log of the odds.
- **power** is defined as \( \eta = \mu^n \). Specifying \( \text{link(power 1)} \) is equivalent to specifying \( \text{link(identity)} \). The power function is generalized so that \( \mu^0 \equiv \ln(\mu) \). Thus, \( \text{link(power 0)} \) is equivalent to \( \text{link(log)} \). Negative powers are, of course, allowed.
- **nbinomial** is defined as \( \eta = \ln\left(\frac{\mu}{\mu + k}\right) \), where \( k = 1 \) if \( \text{family(nbinomial)} \) is specified, \( k = \#k \) if \( \text{family(nbinomial \#k)} \) is specified, and \( k \) is estimated via maximum likelihood if \( \text{family(nbinomial ml)} \) is specified.
- **loglog** is defined as \( \eta = -\ln\left\{ -\ln(\mu) \right\} \).
- **logc** is defined as \( \eta = \ln(1 - \mu) \).

The available distributional families are

<table>
<thead>
<tr>
<th>Family</th>
<th>glm option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian (normal)</td>
<td>family(gaussian)</td>
</tr>
<tr>
<td>inverse Gaussian</td>
<td>family(igaussian)</td>
</tr>
<tr>
<td>Bernoulli/binomial</td>
<td>family(binomial)</td>
</tr>
<tr>
<td>Poisson</td>
<td>family(poisson)</td>
</tr>
<tr>
<td>negative binomial</td>
<td>family(nbinomial)</td>
</tr>
<tr>
<td>gamma</td>
<td>family(gamma)</td>
</tr>
</tbody>
</table>

family(normal) is a synonym for family(gaussian).

The binomial distribution can be specified as 1) family(binomial), 2) family(binomial \#N), or 3) family(binomial varnameN). In case 2, \#N is the value of the binomial denominator \( N \), the number of trials. Specifying family(binomial 1) is the same as specifying family(binomial). In case 3, varnameN is the variable containing the binomial denominator, allowing the number of trials to vary across observations.

The negative binomial distribution can be specified as 1) family(nbinomial), 2) family(nbinomial \#k), or 3) family(nbinomial ml). Omitting \#k is equivalent to specifying family(nbinomial 1). In case 3, the value of \#k is estimated via maximum likelihood. The value \#k enters the variance and deviance functions. Typical values range between 0.01 and 2; see the technical note below.

You do not have to specify both family() and link(); the default link() is the canonical link for the specified family() (except for nbinomial):
### Family Default link

<table>
<thead>
<tr>
<th>Family</th>
<th>Default link</th>
</tr>
</thead>
<tbody>
<tr>
<td>family(gaussian)</td>
<td>link(identity)</td>
</tr>
<tr>
<td>family(igaussian)</td>
<td>link(power -2)</td>
</tr>
<tr>
<td>family(binomial)</td>
<td>link(logit)</td>
</tr>
<tr>
<td>family(poisson)</td>
<td>link(log)</td>
</tr>
<tr>
<td>family(nbinomial)</td>
<td>link(log)</td>
</tr>
<tr>
<td>family(gamma)</td>
<td>link(power -1)</td>
</tr>
</tbody>
</table>

If you specify both `family()` and `link()`, not all combinations make sense. You may choose from the following combinations:

<table>
<thead>
<tr>
<th>identity</th>
<th>log</th>
<th>logit</th>
<th>probit</th>
<th>cloglog</th>
<th>power</th>
<th>opower</th>
<th>nbinomial</th>
<th>loglog</th>
<th>logc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverse Gaussian</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>binomial</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Poisson</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>negative binomial</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gamma</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Technical note

Some `family()` and `link()` combinations result in models already fit by Stata. These are

<table>
<thead>
<tr>
<th>family()</th>
<th>link()</th>
<th>Options</th>
<th>Equivalent Stata command</th>
</tr>
</thead>
<tbody>
<tr>
<td>gaussian</td>
<td>identity</td>
<td>nothing</td>
<td>irls vce(oim)</td>
</tr>
<tr>
<td>gaussian</td>
<td>identity</td>
<td>t(var) vce(hac nwest #) vfactor(#)</td>
<td>newey, t(var) lag(#) (see note 1)</td>
</tr>
<tr>
<td>binomial</td>
<td>cloglog</td>
<td>nothing</td>
<td>irls vce(oim)</td>
</tr>
<tr>
<td>binomial</td>
<td>probit</td>
<td>nothing</td>
<td>irls vce(oim)</td>
</tr>
<tr>
<td>binomial</td>
<td>logit</td>
<td>nothing</td>
<td>irls vce(oim)</td>
</tr>
<tr>
<td>poisson</td>
<td>log</td>
<td>nothing</td>
<td>irls vce(oim)</td>
</tr>
<tr>
<td>nbinomial</td>
<td>log</td>
<td>nothing</td>
<td>irls vce(oim)</td>
</tr>
<tr>
<td>gamma</td>
<td>log</td>
<td>scale(1)</td>
<td>irls vce(oim)</td>
</tr>
</tbody>
</table>

Notes:

1. The variance factor \( \#_v \) should be set to \( n/(n - k) \), where \( n \) is the number of observations and \( k \) the number of regressors. If the number of regressors is not specified, the estimated standard errors will, as a result, differ by this factor.

2. Because the link is not the canonical link for the binomial family, you must specify the `vce(oim)` option if using `irls` to get equivalent standard errors. If `irls` is used without `vce(oim)`, the regression coefficients will be the same but the standard errors will be only asymptotically equivalent. If no options are specified (`nothing`), `glm` will optimize using Newton–Raphson, making it equivalent to the other Stata command.

See [R] `cloglog` and [R] `probit` for more details about these commands.

3. Because the canonical link is being used, the standard errors will be equivalent whether the EIM or the OIM estimator of variance is used.
4. Family negative binomial, log-link models—also known as negative binomial regression models—are used for data with an overdispersed Poisson distribution. Although \texttt{glm} can be used to fit such models, using Stata’s maximum likelihood \texttt{nbreg} command is probably better. In the GLM approach, you specify \texttt{family(nbinomial \#_k)} and then search for a \#_k that results in the deviance-based dispersion being 1. You can also specify \texttt{family(nbinomial ml)} to estimate \#_k via maximum likelihood, which will report the same value returned from \texttt{nbreg}. However, \texttt{nbreg} also reports a confidence interval for it; see [R] \texttt{nbreg} and Rogers (1993). Of course, \texttt{glm} allows links other than log, and for those links, including the canonical \texttt{nbinomial} link, you will need to use \texttt{glm}.

5. \texttt{glm} can be used to estimate parameters from exponential regressions, but this method requires specifying \texttt{scale(1)}. However, censoring is not available. Censored exponential regression may be modeled using \texttt{glm} with \texttt{family(poisson)}. The log of the original response is entered into a Poisson model as an offset, whereas the new response is the censor variable. The result of such modeling is identical to the log relative hazard parameterization of \texttt{streg, dist(exp) nohr}. See [ST] \texttt{streg} for details about the \texttt{streg} command.

In general, where there is overlap between a capability of \texttt{glm} and that of some other Stata command, we recommend using the other Stata command. Our recommendation is not because of some inferiority of the GLM approach. Rather, those other commands, by being specialized, provide options and ancillary commands that are missing in the broader \texttt{glm} framework. Nevertheless, \texttt{glm} does produce the same answers where it should.

Special note. When equivalence is expected, for some datasets, you may still see very slight differences in the results, most often only in the later digits of the standard errors. When you compare \texttt{glm} output to an equivalent Stata command, these tiny discrepancies arise for many reasons:

a. \texttt{glm} uses a general methodology for starting values, whereas the equivalent Stata command may be more specialized in its treatment of starting values.

b. When using a canonical link, \texttt{glm}, \texttt{irls} should be equivalent to the maximum likelihood method of the equivalent Stata command, yet the convergence criterion is different (one is for deviance, the other for log likelihood). These discrepancies are easily resolved by adjusting one convergence criterion to correspond to the other.

c. When both \texttt{glm} and the equivalent Stata command use Newton–Raphson, small differences may still occur if the Stata command has a different default convergence criterion from that of \texttt{glm}. Adjusting the convergence criterion will resolve the difference. See [R] \texttt{ml} and [R] \texttt{Maximize} for more details.

Example 1

In example 1 of [R] \texttt{logistic}, we fit a model based on data from a study of risk factors associated with low birthweight (Hosmer, Lemeshow, and Sturdivant 2013, 24). We can replicate the estimation by using \texttt{glm}:
. use https://www.stata-press.com/data/r18/lbw  
(Hosmer & Lemeshow data)

. glm low age lwt i.race smoke ptl ht ui, family(binomial) link(logit)

Iteration 0:  Log likelihood =  -101.0213
Iteration 1:  Log likelihood =  -100.72519
Iteration 2:  Log likelihood =  -100.724
Iteration 3:  Log likelihood =  -100.724

Generalized linear models

| Coefficient std. err. | z     | P>|z|  | [95% conf. interval] |
|-----------------------|-------|-------|----------------------|
| low                   |       |       |                      |
| age                   | -.0271003 | .0364504 | -0.74   | 0.457 | -.0985418 | .0443412 |
| lwt                   | -.0151508 | .0069259 | -2.19   | 0.029 | -.0287253 | -.0015763 |
| race                  |       |       |                      |
| Black                 | 1.262647 | .5264101 | 2.40    | 0.016 | .2309024 | 2.294392 |
| Other                 | .8620792 | .4391532 | 1.96    | 0.050 | .0013548 | 1.722804 |
| smoke                 | .9233448 | .4008266 | 2.30    | 0.021 | .137739  | 1.708951 |
| ptl                   | .5418366 | .346249  | 1.56    | 0.118 | -.136799 | 1.220472 |
| ht                    | 1.832518 | .6916292 | 2.65    | 0.008 | .4769494 | 3.188086 |
| ui                    | .7585135 | .4593768 | 1.65    | 0.099 | -.1418484 | 1.658875 |
| _cons                 | .4612239 | 1.20459 | 0.38    | 0.702 | -1.899729 | 2.822176 |

glm, by default, presents coefficient estimates, whereas logistic presents the exponentiated coefficients—the odds ratios. glm’s eform option reports exponentiated coefficients, and glm, like Stata’s other estimation commands, replays results.
. glm, eform

Generalized linear models
Number of obs = 189
Optimization : ML
Residual df = 180
Scale parameter = 1
Deviance = 201.447991
(1/df) Deviance = 1.119156
Pearson = 182.0233425
(1/df) Pearson = 1.011241
Variance function: V(u) = u*(1-u)  [Bernoulli]
Link function : g(u) = ln(u/(1-u))  [Logit]
AIC = 1.1611
Log likelihood = -100.7239956  BIC = -742.0665

Note: _cons estimates baseline odds.

These results are the same as those reported in example 1 of \[R\] logistic.

Included in the output header are values for the Akaike (1973) information criterion (AIC) and the Bayesian information criterion (BIC) (Raftery 1995). Both are measures of model fit adjusted for the number of parameters that can be compared across models. In both cases, a smaller value generally indicates a better model fit. AIC is based on the log likelihood and thus is available only when Newton–Raphson optimization is used. BIC is based on the deviance and thus is always available.

Technical note

The values for AIC and BIC reported in the output after glm are different from those reported by estat ic:

. estat ic

Akaike’s information criterion and Bayesian information criterion

<table>
<thead>
<tr>
<th>Model</th>
<th>N</th>
<th>ll(null)</th>
<th>ll(model)</th>
<th>df</th>
<th>AIC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>189</td>
<td>-100.724</td>
<td>219.448</td>
<td>248.6237</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: BIC uses N = number of observations. See \[R\] IC note.

There are various definitions of these information criteria (IC) in the literature; glm and estat ic use different definitions. glm bases its computation of the BIC on deviance, whereas estat ic uses the likelihood. Both glm and estat ic use the likelihood to compute the AIC; however, the AIC from estat ic is equal to N, the number of observations, times the AIC from glm. Refer to Methods and formulas in this entry and \[R\] estat ic for the references and formulas used by glm and estat ic, respectively, to compute AIC and BIC. Inferences based on comparison of IC values reported by glm
for different GLM models will be equivalent to those based on comparison of IC values reported by estat ic after glm.

Example 2

We use data from an early insecticide experiment, given in Pregibon (1980). The variables are ldose, the log dose of insecticide; n, the number of flour beetles subjected to each dose; and r, the number killed.

```
use https://www.stata-press.com/data/r18/ldose
list, sep(4)
```

<table>
<thead>
<tr>
<th>ldose</th>
<th>n</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6907</td>
<td>59</td>
<td>6</td>
</tr>
<tr>
<td>1.7242</td>
<td>60</td>
<td>13</td>
</tr>
<tr>
<td>1.7552</td>
<td>62</td>
<td>18</td>
</tr>
<tr>
<td>1.7842</td>
<td>56</td>
<td>28</td>
</tr>
<tr>
<td>1.8113</td>
<td>63</td>
<td>52</td>
</tr>
<tr>
<td>1.8369</td>
<td>59</td>
<td>53</td>
</tr>
<tr>
<td>1.861</td>
<td>62</td>
<td>61</td>
</tr>
<tr>
<td>1.8839</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

The aim of the analysis is to estimate a dose–response relationship between \( p \), the proportion killed, and \( X \), the log dose.

As a first attempt, we will formulate the model as a linear logistic regression of \( p \) on ldose; that is, we will take the logit of \( p \) and represent the dose–response curve as a straight line in \( X \):

\[
\ln\left\{ \frac{p}{1-p} \right\} = \beta_0 + \beta_1 X
\]

Because the data are grouped, we cannot use Stata’s logistic command to fit the model. Instead, we will fit the model by using glm:

```
use https://www.stata-press.com/data/r18/ldose
list, sep(4)
```

```
. glm r ldose, family(binomial n) link(logit)
 Iteration 0: Log likelihood = -18.824848
 Iteration 1: Log likelihood = -18.715271
 Iteration 2: Log likelihood = -18.715123
 Iteration 3: Log likelihood = -18.715123
 Generalized linear models
 Number of obs = 8
 Optimization : ML Residual df = 6
 Scale parameter = 1
 Deviance = 11.23220702 (1/df) Deviance = 1.872035
 Pearson = 10.0267936 (1/df) Pearson = 1.671132
 Variance function: V(u) = u*(1-u/n) [Binomial]
 Link function : g(u) = ln(u/(n-u)) [Logit]
 Log likelihood = -18.71512262

<table>
<thead>
<tr>
<th></th>
<th>OIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient</td>
</tr>
<tr>
<td>ldose</td>
<td>34.27034</td>
</tr>
<tr>
<td>_cons</td>
<td>-60.71747</td>
</tr>
</tbody>
</table>
We specified `family(binomial n)`, meaning that variable `n` contains the denominator.

An alternative model, which gives asymmetric sigmoid curves for `p`, involves the complementary log–log, or cloglog, function:

\[
\ln\left\{-\ln\left(1 - p\right)\right\} = \beta_0 + \beta_1 X
\]

We fit this model by using `glm`:

```
glm r ldose, family(binomial n) link(cloglog)
```

```
Iteration 0: Log likelihood = -14.883594
Iteration 1: Log likelihood = -14.822264
Iteration 2: Log likelihood = -14.822228
Iteration 3: Log likelihood = -14.822228

Generalized linear models
Number of obs = 8
Optimization : ML
Residual df = 6
Scale parameter = 1
Deviance = 3.446418004 (1/df) Deviance = .574403
Pearson = 3.294675153 (1/df) Pearson = .5491125
Variance function: V(u) = u*(1-u/n)  [Binomial]
Link function : g(u) = ln(-ln(1-u/n))  [Complementary log-log]
AIC = 4.205557
Log likelihood = -14.82222811 BIC = -9.030231
```

<table>
<thead>
<tr>
<th></th>
<th>QIM</th>
<th>OIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>Coefficient</td>
</tr>
<tr>
<td>ldose</td>
<td>22.04118</td>
<td>1.793089</td>
</tr>
<tr>
<td>_cons</td>
<td>-39.57232</td>
<td>3.229047</td>
</tr>
</tbody>
</table>

The cloglog model is preferred; the deviance for the logistic model, 11.23, is much higher than the deviance for the cloglog model, 3.45. This change also is evident by comparing log likelihoods, or equivalently, AIC values.

This example also shows the advantage of the `glm` command—we can vary assumptions easily. Note the minor difference in what we typed to obtain the logistic and cloglog models:

```
glm r ldose, family(binomial n) link(logit)
glm r ldose, family(binomial n) link(cloglog)
```

If we were performing this work for ourselves, we would have typed the commands in a more abbreviated form:

```
glm r ldose, f(b n) l(l)
glm r ldose, f(b n) l(cl)
```

Technical note

Factor variables may be used with `glm`. Say that, in the example above, we had `ldose`, the log dose of insecticide; `n`, the number of flour beetles subjected to each dose; and `r`, the number killed—all as before—except that now we have results for three different kinds of beetles. Our hypothetical data include `beetle`, which contains the values 1 (“Destructive flour”), 2 (“Red flour”), and 3 (“Mealworm”).
. use https://www.stata-press.com/data/r18/beetle
. list, sep(0)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>beetle</td>
<td>ldose</td>
<td>n</td>
<td>r</td>
</tr>
<tr>
<td>1.</td>
<td>1.6907</td>
<td>59</td>
<td>6</td>
</tr>
<tr>
<td>2.</td>
<td>1.7242</td>
<td>60</td>
<td>13</td>
</tr>
<tr>
<td>3.</td>
<td>1.7552</td>
<td>62</td>
<td>18</td>
</tr>
<tr>
<td>4.</td>
<td>1.7842</td>
<td>56</td>
<td>28</td>
</tr>
<tr>
<td>5.</td>
<td>1.8113</td>
<td>63</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>1.861</td>
<td>64</td>
<td>23</td>
</tr>
<tr>
<td>24.</td>
<td>1.8839</td>
<td>58</td>
<td>22</td>
</tr>
</tbody>
</table>

(output omitted)

Let's assume that, at first, we wish merely to add a shift factor for the type of beetle. We could type

. glm r i.beetle ldose, family(bin n) link(cloglog)

Iteration 0:  Log likelihood =  -79.012269
Iteration 1:  Log likelihood =  -76.949510
Iteration 2:  Log likelihood =  -76.945645
Iteration 3:  Log likelihood =  -76.945645

Generalized linear models
Optimization :  ML
Deviance =  73.76505595 (1/df) Deviance =  3.688253
Pearson =  71.8901173 (1/df) Pearson =  3.594506
Variance function:  V(u) = u*(1-u/n)  [Binomial]
Link function :  g(u) = ln(-ln(1-u/n))  [Complementary log-log]
AIC =  6.74547
Log likelihood =  -76.94564525  BIC =  10.20398

<table>
<thead>
<tr>
<th></th>
<th>OIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient</td>
</tr>
<tr>
<td>beetle</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>-.0910396</td>
</tr>
<tr>
<td>Mealworm</td>
<td>-1.836058</td>
</tr>
<tr>
<td>ldose</td>
<td>19.41558</td>
</tr>
<tr>
<td>_cons</td>
<td>-34.84602</td>
</tr>
</tbody>
</table>
We find strong evidence that the insecticide works differently on the mealworm. We now check whether the curve is merely shifted or also differently sloped:

```stata
> glm r beetle##c.ldose, family(bin n) link(cloglog)
```

<table>
<thead>
<tr>
<th></th>
<th>OIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient</td>
</tr>
<tr>
<td>beetle</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>-.79933</td>
</tr>
<tr>
<td>Mealworm</td>
<td>17.78741</td>
</tr>
<tr>
<td>ldose</td>
<td>22.04118</td>
</tr>
<tr>
<td>beetle##c.ldose</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>.3838708</td>
</tr>
<tr>
<td>Mealworm</td>
<td>-10.726</td>
</tr>
<tr>
<td>_cons</td>
<td>-39.57232</td>
</tr>
</tbody>
</table>

We find that the (complementary log–log) dose–response curve for the mealworm has roughly half the slope of that for the destructive flour beetle.

See [U] 26 Working with categorical data and factor variables; what is said there concerning linear regression is applicable to any GLM model.
Variance estimators

`glm` offers many variance options and gives different types of standard errors when used in various combinations. We highlight some of them here, but for a full explanation, see Hardin and Hilbe (2018).

Example 3

Continuing with our flour beetle data, we rerun the most recently displayed model, this time requesting estimation via IRLS.

```stata
use https://www.stata-press.com/data/r18/beetle
 glm r beetle##c.ldose, f(bin n) l(cloglog) ltol(1e-13) irls
```

Iteration 1: Deviance = 54.41414
Iteration 2: Deviance = 50.19424
Iteration 3: Deviance = 50.16973
(output omitted)

<table>
<thead>
<tr>
<th></th>
<th>EIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coefficient</td>
</tr>
<tr>
<td>r</td>
<td>beetle</td>
</tr>
<tr>
<td></td>
<td>Red flour</td>
</tr>
<tr>
<td></td>
<td>Mealworm</td>
</tr>
<tr>
<td>ldose</td>
<td>22.04118</td>
</tr>
<tr>
<td></td>
<td>beetle#c.ldose</td>
</tr>
<tr>
<td></td>
<td>Red flour</td>
</tr>
<tr>
<td></td>
<td>Mealworm</td>
</tr>
<tr>
<td></td>
<td>_cons</td>
</tr>
</tbody>
</table>
```

Note our use of the `ltol()` option, which, although unrelated to our discussion on variance estimation, was used so that the regression coefficients would match those of the previous Newton–Raphson (NR) fit.

Because IRLS uses the EIM for optimization, the variance estimate is also based on EIM. If we want optimization via IRLS but the variance estimate based on OIM, we specify `glm, irls vce(oim)`:
. glm r beetle##c.ldose, f(b n) l(cl) ltol(1e-15) irls vce(oim) noheader nolog

<table>
<thead>
<tr>
<th>r</th>
<th>OIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient std. err. z P&gt;</td>
</tr>
<tr>
<td>beetle</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>-.79933 4.470882 -0.18 0.858 -9.562098 7.963438</td>
</tr>
<tr>
<td>Mealworm</td>
<td>17.78741 4.586429 3.88 0.000 8.798172 26.77664</td>
</tr>
<tr>
<td>ldose</td>
<td>22.04118 1.793089 12.29 0.000 18.52679 25.55557</td>
</tr>
<tr>
<td>beetle#c.ldose</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>.3838708 2.478477 0.15 0.877 -4.473855 5.241596</td>
</tr>
<tr>
<td>Mealworm</td>
<td>-10.726 2.526412 -4.25 0.000 -15.67768 -5.774321</td>
</tr>
<tr>
<td>_cons</td>
<td>-39.57232 3.229047 -12.26 0.000 -45.90114 -33.24351</td>
</tr>
</tbody>
</table>

This approach is identical to NR except for the convergence path. Because the cloglog link is not the canonical link for the binomial family, EIM and OIM produce different results. Both estimators, however, are asymptotically equivalent.

Going back to NR, we can also specify vce(robust) to get the Huber/White/sandwich estimator of variance:

. glm r beetle##c.ldose, f(b n) l(cl) vce(robust) noheader nolog

<table>
<thead>
<tr>
<th>r</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient std. err. z P&gt;</td>
</tr>
<tr>
<td>beetle</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>-.79933 5.733049 -0.14 0.889 -12.0359 10.43724</td>
</tr>
<tr>
<td>Mealworm</td>
<td>17.78741 5.158477 3.45 0.001 7.676977 27.89784</td>
</tr>
<tr>
<td>ldose</td>
<td>22.04118 .8998551 24.49 0.000 20.27749 23.80486</td>
</tr>
<tr>
<td>beetle#c.ldose</td>
<td></td>
</tr>
<tr>
<td>Red flour</td>
<td>.3838708 3.174427 0.12 0.904 -5.837892 6.606633</td>
</tr>
<tr>
<td>Mealworm</td>
<td>-10.726 2.800606 -3.83 0.000 -16.21508 -5.236912</td>
</tr>
<tr>
<td>_cons</td>
<td>-39.57232 1.621306 -24.41 0.000 -42.75003 -36.39462</td>
</tr>
</tbody>
</table>

The sandwich estimator gets its name from the form of the calculation—it is the multiplication of three matrices, with the outer two matrices (the “bread”) set to the OIM variance matrix. When irls is used along with vce(robust), the EIM variance matrix is instead used as the bread. Using a result from McCullagh and Nelder (1989), Newson (1999) points out that the EIM and OIM variance matrices are equivalent under the canonical link. Thus if irls is specified with the canonical link, the resulting variance is labeled “Robust”. When the noncanonical link for the family is used, which is the case in the example below, the EIM and OIM variance matrices differ, so the resulting variance is labeled “Semirobust”.

```shell
Robust
```
```stata
. glm r beetle##c.ldose, f(b n) l(cl) irls ltol(1e-15) vce(robust) noheader > nolog
```

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient</td>
<td>std. err.</td>
<td>z</td>
<td>P&gt;</td>
<td>z</td>
</tr>
<tr>
<td><strong>beetle</strong></td>
<td><strong>Red flour</strong></td>
<td>-.79933</td>
<td>6.288963</td>
<td>-0.13</td>
<td>0.899</td>
</tr>
<tr>
<td><strong>Mealworm</strong></td>
<td>17.78741</td>
<td>5.255307</td>
<td>3.38</td>
<td>0.001</td>
<td>7.487194</td>
</tr>
<tr>
<td><strong>ldose</strong></td>
<td>22.04118</td>
<td>.9061566</td>
<td>24.32</td>
<td>0.000</td>
<td>20.26514</td>
</tr>
<tr>
<td><strong>beetle#c.ldose</strong></td>
<td><strong>Red flour</strong></td>
<td>.3838708</td>
<td>3.489723</td>
<td>0.11</td>
<td>0.912</td>
</tr>
<tr>
<td><strong>Mealworm</strong></td>
<td>-10.726</td>
<td>2.855897</td>
<td>-3.76</td>
<td>0.000</td>
<td>-16.32345</td>
</tr>
<tr>
<td><strong>_cons</strong></td>
<td>-39.57232</td>
<td>1.632544</td>
<td>-24.24</td>
<td>0.000</td>
<td>-42.77205</td>
</tr>
</tbody>
</table>

The outer product of the gradient (OPG) estimate of variance is one that avoids the calculation of second derivatives. It is equivalent to the “middle” part of the sandwich estimate of variance and can be specified by using `glm, vce(opg)`, regardless of whether NR or IRLS optimization is used.

```stata
. glm r beetle##c.ldose, f(b n) l(cl) vce(opg) noheader nolog
```

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient</td>
<td>std. err.</td>
<td>z</td>
<td>P&gt;</td>
<td>z</td>
</tr>
<tr>
<td><strong>beetle</strong></td>
<td><strong>Red flour</strong></td>
<td>-.79933</td>
<td>6.664045</td>
<td>-0.12</td>
<td>0.905</td>
</tr>
<tr>
<td><strong>Mealworm</strong></td>
<td>17.78741</td>
<td>6.838505</td>
<td>2.60</td>
<td>0.009</td>
<td>4.384183</td>
</tr>
<tr>
<td><strong>ldose</strong></td>
<td>22.04118</td>
<td>3.572983</td>
<td>6.17</td>
<td>0.000</td>
<td>15.03826</td>
</tr>
<tr>
<td><strong>beetle#c.ldose</strong></td>
<td><strong>Red flour</strong></td>
<td>.3838708</td>
<td>3.700192</td>
<td>0.10</td>
<td>0.917</td>
</tr>
<tr>
<td><strong>Mealworm</strong></td>
<td>-10.726</td>
<td>3.796448</td>
<td>-2.83</td>
<td>0.005</td>
<td>-18.1669</td>
</tr>
<tr>
<td><strong>_cons</strong></td>
<td>-39.57232</td>
<td>6.433101</td>
<td>-6.15</td>
<td>0.000</td>
<td>-52.18097</td>
</tr>
</tbody>
</table>

The OPG estimate of variance is a component of the BHHH (Berndt et al. 1974) optimization technique. This method of optimization is also available with `glm` with the `technique()` option; however, the `technique()` option is not allowed with the `irls` option.

### Example 4

The Newey–West (1987) estimator of variance is a sandwich estimator with the “middle” of the sandwich modified to account for possible autocorrelation between the observations. These estimators are a generalization of those given by the Stata command `newey` for linear regression. See [TS] `newey` for more details.

For example, consider the dataset given in [TS] `newey`, which has time-series measurements on `usr` and `idle`. We want to perform a linear regression with Newey–West standard errors.
Examining Methods and formulas of [TS] newey, we see that the variance estimate is multiplied by a correction factor of \( \frac{n}{(n-k)} \), where \( k \) is the number of regressors. \texttt{glm}, \texttt{vce(hac ...)} does not make this correction, so to get the same standard errors, we must use the \texttt{vfactor()} option within \texttt{glm} to make the correction manually.

. display 30/28
1.0714286

.tsset time
Time variable: time, 1 to 30
Delta: 1 unit

.glm usr idle, vce(hac nwest 3) vfactor(1.0714286)
Iteration 0: Log likelihood = -71.74339627

Generalized linear models
Number of obs = 30
Optimization : ML
Residual df = 28
Scale parameter = 7.493297
(1/df) Deviance = 7.493297
(1/df) Pearson = 7.493297
Variance function: V(u) = 1 [Gaussian]
Link function : g(u) = u [Identity]
HAC kernel (lags): Newey-West (3)
AIC = 4.916226
BIC = 114.5788

| HAC | Coefficient | std. err. | z | P>|z| | [95% conf. interval] |
|-----|-------------|-----------|---|------|---------------------|
| idle| -.2281501   | .0690928  | -3.30 | 0.001 | -.3635694 to -.0927307 |
| _cons| 23.13483    | 6.327033  | 3.66 | 0.000 | 10.73407 to 35.53558 |

The \texttt{glm} command above reproduces the results given in [TS] newey. We may now generalize this output to models other than simple linear regression and to different kernel weights.
. glm usr idle, fam(gamma) link(log) vce(hac gallant 3)

Iteration 0: Log likelihood = -61.76593
Iteration 1: Log likelihood = -60.963233
Iteration 2: Log likelihood = -60.95097
Iteration 3: Log likelihood = -60.950965

Generalized linear models
Optimization : ML
Scale parameter = .431296
Deviance = 9.908506707
Pearson = 12.07628677
Variance function: V(u) = u^2 [Gamma]
Link function : g(u) = ln(u) [Log]
HAC kernel (lags): Gallant (3)

AIC = 4.196731
Log likelihood = -60.95096484
BIC = -85.32502

<table>
<thead>
<tr>
<th></th>
<th>HAC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient std. err.</td>
</tr>
<tr>
<td>idle</td>
<td>-.0796609 .0184647 -4.31 0.000 -.115851 -.0434708</td>
</tr>
<tr>
<td>_cons</td>
<td>7.771011 1.510198 5.15 0.000 4.811078 10.73094</td>
</tr>
</tbody>
</table>

.glm also offers variance estimators based on the bootstrap (resampling your data with replacement) and the jackknife (refitting the model with each observation left out in succession). Also included is the one-step jackknife estimate, which, instead of performing full reestimation when each observation is omitted, calculates a one-step NR estimate, with the full data regression coefficients as starting values.

. set seed 1
. glm usr idle, fam(gamma) link(log) vce(bootstrap, reps(100) nodots)

Generalized linear models
Optimization : ML
Scale parameter = .431296
Deviance = 9.908506707
Pearson = 12.07628677
Variance function: V(u) = u^2 [Gamma]
Link function : g(u) = ln(u) [Log]

Log likelihood = -60.95096484
BIC = -85.32502

<table>
<thead>
<tr>
<th></th>
<th>Observed Bootstrap Normal-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coefficient std. err.</td>
</tr>
<tr>
<td></td>
<td>[95% conf. interval]</td>
</tr>
<tr>
<td>idle</td>
<td>-.0796609 .016657 -4.78 0.000 -.1123081 -.0470137</td>
</tr>
<tr>
<td>_cons</td>
<td>7.771011 1.378037 5.64 0.000 5.070108 10.47192</td>
</tr>
</tbody>
</table>

See Hardin and Hilbe (2018) for a full discussion of the variance options that go with glm and, in particular, of how the different variance estimators are modified when vce(cluster clustvar) is specified. Finally, not all variance options are supported with all types of weights. See help glm for a current table of the variance options that are supported with the different weights.
User-defined functions

`glm` may be called with a community-contributed link function, variance (family) function, Newey–West kernel-weight function, or any combination of the three.

Syntax of link functions

```
program proglename
    version 18.0
    args todo eta mu return
    if 'todo' == -1 {
        /* Set global macros for output */
        global SGLM_lt "title for link function"
        global SGLM lf "subtitle showing link definition"
        exit
    }
    if 'todo' == 0 {
        /* Intermediate calculations go here */
        generate double 'eta' = ...
        exit
    }
    if 'todo' == 1 {
        /* Intermediate calculations go here */
        generate double 'mu' = ...
        exit
    }
    if 'todo' == 2 {
        /*Intermediate calculations go here */
        generate double 'return' = ...
        exit
    }
    if 'todo' == 3 {
        /*Intermediate calculations go here */
        generate double 'return' = ...
        exit
    }
    display as error "Unknown call to glm link function"
    exit 198
end
```
Syntax of variance functions

program proname
    version 18.0
    args todo eta mu return
    if ‘todo’ == -1 {
        /* Set global macros for output */
        /* Also check that depvar is in proper range */
        /* Note: For this call, eta contains indicator for whether each obs. is in est. sample */
        global SGLM_vt "title for variance function"
        global SGLM_vf "subtitle showing function definition"
        global SGLM_mu "program to call to enforce boundary conditions on \( \mu \)"
        exit
    }
    if ‘todo’ == 0 {
        /* set \( \eta \) to initial value. */
        /* Intermediate calculations go here */
        generate double ‘eta’ = ...
        exit
    }
    if ‘todo’ == 1 {
        /* set return = \( V(\mu) \) */
        /* Intermediate calculations go here */
        generate double ‘return’ = ...
        exit
    }
    if ‘todo’ == 2 {
        /* set return = \( \partial V(\mu)/\partial \mu \) */
        /* Intermediate calculations go here */
        generate double ‘return’ = ...
        exit
    }
    if ‘todo’ == 3 {
        /* set return = squared deviance (per observation) */
        /* Intermediate calculations go here */
        generate double ‘return’ = ...
        exit
    }
    if ‘todo’ == 4 {
        /* set return = Anscombe residual */
        /* Intermediate calculations go here */
        generate double ‘return’ = ...
        exit
    }
    if ‘todo’ == 5 {
        /* set return = log likelihood */
        /* Intermediate calculations go here */
        generate double ‘return’ = ...
        exit
    }
    if ‘todo’ == 6 {
        /* set return = adjustment for deviance residuals */
        /* Intermediate calculations go here */
        generate double ‘return’ = ...
        exit
    }
    display as error "Unknown call to glm variance function"
    exit 198
end
Syntax of Newey–West kernel-weight functions

```
program programe, rclass
    version 18.0
    args G j
    /* G is the maximum lag */
    /* j is the current lag */
    /* Intermediate calculations go here */
    return scalar wt = computed weight
    return local setype "Newey-West"
    return local sewtype "name of kernel"
end
```

Global macros available for community-contributed programs

<table>
<thead>
<tr>
<th>Global macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGLM_V</td>
<td>program name of variance (family) evaluator</td>
</tr>
<tr>
<td>SGLM_L</td>
<td>program name of link evaluator</td>
</tr>
<tr>
<td>SGLM_y</td>
<td>dependent variable name</td>
</tr>
<tr>
<td>SGLM_m</td>
<td>binomial denominator</td>
</tr>
<tr>
<td>SGLM_a</td>
<td>negative binomial k</td>
</tr>
<tr>
<td>SGLM_p</td>
<td>power if power() or opower() is used, or an argument from a user-specified link function</td>
</tr>
<tr>
<td>SGLM_s1</td>
<td>indicator; set to one if scale is equal to one</td>
</tr>
<tr>
<td>SGLM_ph</td>
<td>value of scale parameter</td>
</tr>
</tbody>
</table>

Example 5

Suppose that we wish to perform Poisson regression with a log-link function. Although this regression is already possible with standard `glm`, we will write our own version for illustrative purposes.

Because we want a log link, \( \eta = g(\mu) = \ln(\mu) \), and for a Poisson family the variance function is \( V(\mu) = \mu \).

The Poisson density is given by

\[
f(y_i) = \frac{e^{-\exp(\mu_i)} e^{\mu_i y_i}}{y_i!}
\]

resulting in a log likelihood of

\[
L = \sum_{i=1}^{n} \{-e^{\mu_i} + \mu_i y_i - \ln(y_i!)}\]

The squared deviance of the \( i \)th observation for the Poisson family is given by

\[
d^2_i = \begin{cases} 
2\mu_i & \text{if } y_i = 0 \\
2\{y_i \ln(y_i/\hat{\mu}_i) - (y_i - \hat{\mu}_i)\} & \text{otherwise}
\end{cases}
\]
We now have enough information to write our own Poisson-log glm module. We create the file mylog.ado, which contains

```
program mylog
  version 18.0
  args todo eta mu return
  if 'todo' == -1 {
    global SGLM_lt "My Log"       // Titles for output
    global SGLM_lf "ln(u)"
    exit
  }
  if 'todo' == 0 {
    gen double 'eta' = ln('mu')    // η = ln(μ)
    exit
  }
  if 'todo' == 1 {
    gen double 'mu' = exp('eta')   // μ = exp(η)
    exit
  }
  if 'todo' == 2 {
    gen double 'return' = 'mu'     // ∂μ/∂η = exp(η) = μ
    exit
  }
  if 'todo' == 3 {
    gen double 'return' = 'mu'     // ∂²μ/∂η² = exp(η) = μ
    exit
  }
  di as error "Unknown call to glm link function"
  exit 198
end
```

and we create the file mypois.ado, which contains

```
program mypois
  version 18.0
  args todo eta mu return
  if 'todo' == -1 {
    local y "$SGLM_y"
    local touse "'eta'"       // ‘eta’ marks estimation sample here
    capture assert `y'>=0 if `touse'       // check range of y
    if _rc {
      di as error "dependent variable ‘y’ has negative values"
      exit 499
    }
    global SGLM_vt "My Poisson"       // Titles for output
    global SGLM_vf "u"
    global SGLM_mu "glim_mu 0 ."     // see note 1
    exit
  }
  if ‘todo’ == 0 {
    gen double ‘eta’ = ln(‘mu’)      // Initialization of η; see note 2
    exit
  }
```

```
if 'todo' == 1 {  
    gen double 'return' = 'mu'  
    exit  
}
if 'todo' == 2 {  
    gen byte 'return' = 1  
    exit  
}
if 'todo' == 3 {  
    local y "$SGLM_y"
    if "'y'" == "" {  
        local y "'e(depvar)'"
    }
    gen double 'return' = cond('y'==0, 2*('mu'), /*
        */ 2*('y'*ln('y'/'mu')-('y'-'mu')))  
    exit  
}
if 'todo' == 4 {  
    local y "$SGLM_y"
    if "'y'" == "" {  
        local y "'e(depvar)'"
    }
    gen double 'return' = 1.5*('y'^(2/3)-'mu'^(2/3)) / 'mu'^(1/6)  
    exit  
}
if 'todo' == 5 {  
    local y "$SGLM_y"
    if "'y'" == "" {  
        local y "'e(depvar)'"
    }
    gen double 'return' = -'mu'+'y'*ln('mu')-lngamma('y'+1)  
    exit  
}
if 'todo' == 6 {  
    gen double 'return' = 1/(6*sqrt('mu'))  
    exit  
}
end

di as error "Unknown call to glm variance function"
error 198

Notes:

1. \texttt{glim\_mu} is a Stata program that will, at each iteration, bring \( \hat{\mu} \) back into its plausible range, should it stray out of it. Here \texttt{glim\_mu} is called with the arguments zero and missing, meaning that zero is the lower bound of \( \hat{\mu} \) and there exists no upper bound—such is the case for Poisson models.

2. Here the initial value of \( \eta \) is easy because we intend to fit this model with our user-defined log link. In general, however, the initialization may need to vary according to the link to obtain convergence. If so, the global macro \texttt{SGLM\_L} is used to determine which link is being utilized.

3. The Anscombe formula is given here because we know it. If we were not interested in Anscombe residuals, we could merely set \('return'\) to missing. Also, the local macro \texttt{y} is set either to \texttt{SGLM\_y} if it is in current estimation or to \texttt{e(depvar)} if this function is being accessed by \texttt{predict}.

4. If we were not interested in ML estimation, we could omit this code entirely and just leave an exit statement in its place. Similarly, if we were not interested in deviance or IRLS optimization, we could set \('return'\) in the deviance portion of the code \(('todo'==3)\) to missing.
5. This code defines the term to be added to the predicted residuals if the adjusted option is specified. Again, if we were not interested, we could set ‘return’ to missing.

We can now test our Poisson-log module by running it on the airline data presented in [R] poisson.

```
. use https://www.stata-press.com/data/r18/airline
. list airline injuries n XYZowned

<table>
<thead>
<tr>
<th>airline</th>
<th>injuries</th>
<th>n</th>
<th>XYZowned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.0950</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0.1920</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7</td>
<td>0.0750</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>19</td>
<td>0.2078</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>9</td>
<td>0.1382</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4</td>
<td>0.0540</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3</td>
<td>0.1292</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1</td>
<td>0.0503</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>3</td>
<td>0.0629</td>
</tr>
</tbody>
</table>
```

. generate lnN=ln(n)
. glm injuries XYZowned lnN, f(mypoisy) l(mylog) scale(1)

```
Iteration 0: Log likelihood = -22.557572
Iteration 1: Log likelihood = -22.332861
Iteration 2: Log likelihood = -22.332276
Iteration 3: Log likelihood = -22.332276

Generalized linear models
Number of obs = 9
Optimization : ML
Residual df = 6
Scale parameter = 1
Deviance = 12.70432823 (1/df) Deviance = 2.117388
Pearson = 12.7695081 (1/df) Pearson = 2.128251
Variance function: V(u) = u [My Poisson]
Link function : g(u) = ln(u) [My Log]
AIC = 5.629395
Log likelihood = -22.33227605 BIC = -.4790192

| injuries | Coefficient | std. err. | z     | P>|z| | [95% conf. interval] |
|----------|-------------|-----------|-------|-----|---------------------|
| XYZowned | .6840668    | .3895877  | 1.76  | 0.079 | -.0795111 1.447645 |
| lnN      | 1.424169    | .3725155  | 3.82  | 0.000 | .6940517 2.154286 |
| _cons    | 4.863891    | .7090501  | 6.86  | 0.000 | 3.474178 6.253603 |
```

(Standard errors scaled using dispersion equal to square root of 1.)

These are precisely the results given in [R] poisson and are those that would have been given had we run glm, family(poisson) link(log). The only minor adjustment we needed to make was to specify the scale(1) option. If scale() is left unspecified, glm assumes scale(1) for discrete distributions and scale(x2) for continuous ones. By default, glm assumes that any user-defined family is continuous because it has no way of checking. Thus, we needed to specify scale(1) because our model is discrete.

Because we were careful in defining the squared deviance, we could have fit this model with IRLS. Because log is the canonical link for the Poisson family, we would not only get the same regression coefficients but also the same standard errors.
Example 6

Suppose now that we wish to use our log link (mylog.ado) with glm's binomial family. This task requires some modification because our current function is not equipped to deal with the binomial denominator, which we are allowed to specify. This denominator is accessible to our link function through the global macro SGLM_m. We now make the modifications and store them in mylog2.ado.

```stata
program mylog2 // <-- changed
  version 18.0
  args todo eta mu return
  if 'todo' == -1 {
    global SGLM_lt "My Log, Version 2" // <-- changed
    if "$SGLM_m" == "1" { // <-- changed
      global SGLM_lf "ln(u)" // <-- changed
    }
    else global SGLM_lf "ln(u/$SGLM_m)" // <-- changed
    exit
  }
  if 'todo' == 0 {
    gen double 'eta' = ln('mu'/$SGLM_m) // <-- changed
    exit
  }
  if 'todo' == 1 {
    gen double 'mu' = $SGLM_m*exp('eta') // <-- changed
    exit
  }
  if 'todo' == 2 {
    gen double 'return' = 'mu'
    exit
  }
  if 'todo' == 3 {
    gen double 'return' = 'mu'
    exit
  }
  di as error "Unknown call to glm link function"
end
```

We can now run our new log link with glm’s binomial family. Using the flour beetle data from earlier, we have
. use https://www.stata-press.com/data/r18/beetle, clear
. glm r ldose, f(bin n) l(mylog2) irls
Iteration 1:  Deviance =  2212.108
Iteration 2:  Deviance =  452.9352
Iteration 3:  Deviance =  429.95
Iteration 4:  Deviance =  429.2745
Iteration 5:  Deviance =  429.2192
Iteration 6:  Deviance =  429.2082
Iteration 7:  Deviance =  429.2061
Iteration 8:  Deviance =  429.2057
Iteration 9:  Deviance =  429.2056
Iteration 10:  Deviance =  429.2056
Iteration 11:  Deviance =  429.2056
Iteration 12:  Deviance =  429.2056

Generalized linear models  Number of obs = 24
Optimization : MQL Fisher scoring  Residual df = 22
              (IRLS EIM)  Scale parameter = 1
Deviance = 429.205599  (1/df) Deviance = 19.50935
Pearson = 413.088142  (1/df) Pearson = 18.77673
Variance function: V(u) = u*(1-u/n)  [Binomial]
Link function : g(u) = ln(u/n)  [My Log, Version 2]
BIC = 359.2884

| r     | EIM            | Coefficient | std. err. | z  | P>|z| | [95% conf. interval] |
|-------|----------------|-------------|-----------|----|-----|-------------------|
| ldose |                | 8.478908    | .4702808  | 18.03 | 0.000 | 7.557175 - 9.400642 |
| _cons |                | -16.11006   | .8723167  | -18.47 | 0.000 | -17.81977 -14.40035 |

For a more detailed discussion on user-defined functions, and for an example of a user-defined Newey–West kernel weight, see Hardin and Hilbe (2018).

John Ashworth Nelder (1924–2010) was born in Somerset, England. He studied mathematics and statistics at Cambridge and worked as a statistician at the National Vegetable Research Station and then Rothamsted Experimental Station. In retirement, he was actively affiliated with Imperial College London. Nelder was especially well known for his contributions to the theory of linear models and to statistical computing. He was the principal architect of generalized and hierarchical generalized linear models and of the programs GenStat and GLIM.

Robert William Maclagan Wedderburn (1947–1975) was born in Edinburgh and studied mathematics and statistics at Cambridge. At Rothamsted Experimental Station, he developed the theory of generalized linear models with Nelder and originated the concept of quasilikelihood. He died of anaphylactic shock from an insect bite on a canal holiday.
Stored results

glm, ml stores the following in e():

Scalars

- `e(N)` number of observations
- `e(k)` number of parameters
- `e(k_eq)` number of equations in `e(b)`
- `e(k_eq_model)` number of equations in overall model test
- `e(k_dv)` number of dependent variables
- `e(df_m)` model degrees of freedom
- `e(df)` residual degrees of freedom
- `e(phil)` scale parameter
- `e(aic)` model AIC
- `e(bic)` model BIC
- `e(ll)` log likelihood, if NR
- `e(N_clust)` number of clusters
- `e(chi2)` $\chi^2$
- `e(p)` $p$-value for model test
- `e(deviance)` deviance
- `e(deviance_s)` scaled deviance
- `e(deviance_p)` Pearson deviance
- `e(deviance_ps)` scaled Pearson deviance
- `e(dispers)` dispersion
- `e(dispers_s)` scaled dispersion
- `e(dispers_p)` Pearson dispersion
- `e(dispers_ps)` scaled Pearson dispersion
- `e(nbml)` 1 if negative binomial parameter estimated via ML, 0 otherwise
- `e(vf)` factor set by `vfactor()`, 1 if not set
- `e(power)` power set by `link(power #)` or `link(opower #)`
- `e(rank)` rank of `e(V)`
- `e(ic)` number of iterations
- `e(rc)` return code
- `e(converged)` 1 if converged, 0 otherwise

Macros

- `e(cmd)` `glm`
- `e(cmdline)` command as typed
- `e(depvar)` name of dependent variable
- `e(varfunc)` program to calculate variance function
- `e(varfuncf)` variance title
- `e(link)` program to calculate link function
- `e(linkt)` link title
- `e(linkf)` link function
- `e(m)` number of binomial trials
- `e(vtype)` weight type
- `e(wexp)` weight expression
- `e(title)` title in estimation output
- `e(clustvar)` name of cluster variable
- `e(offset)` linear offset variable
- `e(chi2type)` Wald; type of model $\chi^2$ test
- `e(cons)` noconstant, if specified
- `e(hac_kernel)` HAC kernel
- `e(hac_lag)` HAC lag
- `e(vce)` vcetype specified in `vce()`
- `e(vcetype)` title used to label Std. err.
- `e(opt)` `ml` or `irls`
- `e(opt1)` optimization title, line 1
- `e(opt2)` optimization title, line 2
- `e(which)` max or min; whether optimizer is to perform maximization or minimization
- `e(ml_method)` type of ml method
- `e(user)` name of likelihood-evaluator program
- `e(technique)` maximization technique
In addition to the above, the following is stored in `r()`:

Matrices

- `r(table)` — matrix containing the coefficients with their standard errors, test statistics, \( p \)-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any `r`-class command is run after the estimation command.

### glm, irls stores the following in `e()`:

**Scalars**

- `e(N)` — number of observations
- `e(k)` — number of parameters
- `e(k_eq_model)` — number of equations in overall model test
- `e(df_m)` — model degrees of freedom
- `e(df)` — residual degrees of freedom
- `e(phi)` — scale parameter
- `e(N_clust)` — number of clusters
- `e(deviance)` — deviance
- `e(deviance_s)` — scaled deviance
- `e(deviance_p)` — Pearson deviance
- `e(deviance_ps)` — scaled Pearson deviance
- `e(dispers)` — dispersion
- `e(dispers_s)` — scaled dispersion
- `e(dispers_p)` — Pearson dispersion
- `e(dispers_ps)` — scaled Pearson dispersion
- `e(nbml)` — 1 if negative binomial parameter estimated via ML, 0 otherwise
- `e(vf)` — factor set by `vfactor()`, 1 if not set
- `e(power)` — power set by `link(power #)` or `link(opower #)`
- `e(rank)` — rank of `e(V)`
- `e(rc)` — return code

### Macros

- `e(cmd)` — `glm`
- `e(cmdline)` — command as typed
- `e(depvar)` — name of dependent variable
- `e(varfunc)` — program to calculate variance function
- `e(varfunct)` — variance title
- `e(varfuncf)` — variance function
- `e(link)` — program to calculate link function
- `e(linkt)` — link title
- `e(linkf)` — link function
- `e(m)` — number of binomial trials
glm — Generalized linear models

Matrices
e(b) coefficient vector
e(V) variance–covariance matrix of the estimators
e(V_modelbased) model-based variance

Functions
e(sample) marks estimation sample

In addition to the above, the following is stored in r():

Matrices
r(table) matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas


This discussion highlights the details of parameter estimation and predicted statistics. For a more detailed treatment, and for information on variance estimation, see Hardin and Hilbe (2018). glm supports estimation with survey data. For details on VCEs with survey data, see [SVY] Variance estimation.

glm obtains results by IRLS, as described in McCullagh and Nelder (1989), or by maximum likelihood using Newton–Raphson. The implementation here, however, allows user-specified weights, which we denote as \( w_j \) for the \( j \)th observation. Let \( M \) be the number of “observations” ignoring weights. Define

\[
w_j = \begin{cases} 
1 & \text{if no weights are specified} \\
 v_j & \text{if fweights or iweights are specified} \\
 Mv_j/(\sum_k v_k) & \text{if aweights or pweights are specified}
\end{cases}
\]
The number of observations is then $N = \sum_{j} w_j$ if weights are specified and $N = M$ otherwise. Each IRLS step is performed by \texttt{regress} using $w_j$ as the weights.

Let $d_j^2$ denote the squared deviance residual for the $j$th observation:

For the Gaussian family, $d_j^2 = (y_j - \hat{\mu}_j)^2$.

For the Bernoulli family (binomial with denominator 1),

$$d_j^2 = \begin{cases} -2\ln(1 - \hat{\mu}_j) & \text{if } y_j = 0 \\ -2\ln(\hat{\mu}_j) & \text{otherwise} \end{cases}$$

For the binomial family with denominator $m_j$,

$$d_j^2 = \begin{cases} 2y_j\ln(y_j/\hat{\mu}_j) + 2(m_j - y_j)\ln\{m_j/(m_j - \hat{\mu}_j)\} & \text{if } 0 < y_j < m_j \\ 2m_j\ln\{m_j/(m_j - \hat{\mu}_j)\} & \text{if } y_j = 0 \\ 2y_j\ln(y_j/\hat{\mu}_j) & \text{if } y_j = m_j \end{cases}$$

For the Poisson family,

$$d_j^2 = \begin{cases} 2\hat{\mu}_j & \text{if } y_j = 0 \\ 2\{y_j\ln(y_j/\hat{\mu}_j) - (y_j - \hat{\mu}_j)\} & \text{otherwise} \end{cases}$$

For the gamma family, $d_j^2 = -2\{\ln(y_j/\hat{\mu}_j) - (y_j - \hat{\mu}_j)/\hat{\mu}_j\}$.

For the inverse Gaussian, $d_j^2 = (y_j - \hat{\mu}_j)^2/(\hat{\mu}_j^2y_j)$.

For the negative binomial,

$$d_j^2 = \begin{cases} 2\ln(1 + k\hat{\mu}_j)/k & \text{if } y_j = 0 \\ 2y_j\ln(y_j/\hat{\mu}_j) - 2\{(1 + ky_j)/k\ln\{(1 + ky_j)/(1 + k\hat{\mu}_j)\} & \text{otherwise} \end{cases}$$

Let $\phi = 1$ if the scale parameter is set to one; otherwise, define $\phi = \hat{\phi}_0(n - k)/n$, where $\hat{\phi}_0$ is the estimated scale parameter and $k$ is the number of covariates in the model (including intercept).

Let $\ln L_j$ denote the log likelihood for the $j$th observation:

For the Gaussian family,

$$\ln L_j = -\frac{1}{2} \left[ \frac{(y_j - \hat{\mu}_j)^2}{\phi} \right] + \ln(2\pi\phi)$$

For the binomial family with denominator $m_j$ (Bernoulli if all $m_j = 1$),

$$\ln L_j = \phi \times \begin{cases} \ln\{\Gamma(m_j + 1)\} - \ln\{\Gamma(y_j + 1)\} - \ln\{\Gamma(m_j - y_j + 1)\} & \text{if } 0 < y_j < m_j \\ (m_j - y_j)\ln(1 - \hat{\mu}_j/m_j) + y_j\ln(\hat{\mu}_j/m_j) & \text{if } y_j = 0 \\ m_j\ln(1 - \hat{\mu}_j/m_j) & \text{if } y_j = m_j \end{cases}$$

For the Poisson family,

$$\ln L_j = \phi \left[ y_j\ln(\hat{\mu}_j) - \hat{\mu}_j - \ln\{\Gamma(y_j + 1)\} \right]$$
For the gamma family, $\ln L_j = -y_j/\hat{\mu}_j + \ln(1/\hat{\mu}_j)$.

For the inverse Gaussian,

$$\ln L_j = -\frac{1}{2} \left\{ \frac{(y_j - \hat{\mu}_j)^2}{y_j \hat{\mu}_j^2} + 3 \ln(y_j) + \ln(2\pi) \right\}$$

For the negative binomial (let $m = 1/k$),

$$\ln L_j = \phi \left[ \ln\{\Gamma(m + y_j)\} - \ln\{\Gamma(y_j + 1)\} - \ln\{\Gamma(m)\} - m \ln(1 + \hat{\mu}_j/m) + y_j \ln\{\hat{\mu}_j/(\hat{\mu}_j + m)\} \right]$$

The overall deviance reported by glm is $D^2 = \sum_j w_j d_j^2$. The dispersion of the deviance is $D^2$ divided by the residual degrees of freedom.

The Akaike information criterion (AIC) and Bayesian information criterion (BIC) are given by

$$\text{AIC} = -2 \ln L + 2k \frac{N}{N - k} \ln(N)$$
$$\text{BIC} = D^2 - (N - k) \ln(N)$$

where $\ln L = \sum_j w_j \ln L_j$ is the overall log likelihood.

The Pearson deviance reported by glm is $\sum_j w_j r_j^2$. The corresponding Pearson dispersion is the Pearson deviance divided by the residual degrees of freedom. glm also calculates the scaled versions of all of these quantities by dividing by the estimated scale parameter.

Acknowledgments

glm was written by James Hardin of the Arnold School of Public Health at the University of South Carolina and Joseph Hilbe (1944–2017) of Arizona State University, the coauthors of the Stata Press book Generalized Linear Models and Extensions. The previous version of this routine was written by Patrick Royston of the MRC Clinical Trials Unit, London, and coauthor of the Stata Press book Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model. The original version of this routine was published in Royston (1994). Royston’s work, in turn, was based on a prior implementation by Joseph Hilbe, first published in Hilbe (1993). Roger Newson wrote an early implementation (Newson 1999) of robust variance estimates for GLM. Parts of this entry are excerpts from Hardin and Hilbe (2018).

References


Also see

[R] glm postestimation — Postestimation tools for glm
[R] cloglog — Complementary log–log regression
[R] logistic — Logistic regression, reporting odds ratios
[R] nbreg — Negative binomial regression
[R] poisson — Poisson regression
[R] regress — Linear regression
[BAYES] bayes: glm — Bayesian generalized linear models
[FMM] fmm: glm — Finite mixtures of generalized linear regression models
[ME] meglm — Multilevel mixed-effects generalized linear models
[MI] Estimation — Estimation commands for use with mi estimate
[SVY] svy estimation — Estimation commands for survey data
[XT] xtggee — GEE population-averaged panel-data models
[U] 20 Estimation and postestimation commands