Title

stata.com

glm — Generalized linear models

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
Acknowledgments | References | Also see

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, family(binomial)
```

Logit model of `y` events occurring in 15 trials as a function of `x`

```
glm y x, 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, family(binomial n) link(probit)
```

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

```
glm y x, 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, family(gamma) link(log) vce(bootstrap)
```

Menu

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

Syntax

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

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

familyname	Description
gaussian	Gaussian (normal)
igaussian	inverse Gaussian
binomial [*varname*$_N$ | #$_N$]	Bernoulli/binomial
poisson	Poisson
nbinomial [#$_k$ | ml]	negative binomial
gamma	gamma

linkname	Description
identity	identity
log	log
logit	logit
probit	probit
cloglog	clog-log
power #	power
opower #	odds power
nbinomial	negative binomial
loglog	log-log
logc	log-complement

indepvars may contain factor variables; see [U] **11.4.3 Factor variables**.

depvar and *indepvars* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.

bayes, bootstrap, by, fmm, fp, jackknife, mfp, mi estimate, nestreg, rolling, statsby, stepwise, and svy are allowed; see [U] **11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: glm** and [FMM] **fmm: glm**.

vce(bootstrap), vce(jackknife), and vce(jackknife1) are not allowed with the mi estimate prefix; see [MI] **mi estimate**.

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

aweights are not allowed with the jackknife prefix; see [R] **jackknife**.

vce(), vfactor(), disp(), scale(), irls, fisher(), noheader, notable, nodisplay, and weights are not allowed with the svy prefix; see [SVY] **svy**.

fweights, aweights, iweights, and pweights are allowed; see [U] **11.1.6 weight**.

noheader, notable, nodisplay, collinear, and coeflegend do not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

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)).

Model 2

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.

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

SE/Robust

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

In addition to the standard *vcetype*s, `glm` allows the following alternatives:

`vce(eim)` specifies that the EIM estimate of variance be used.

`vce(jackknife1)` specifies that the one-step jackknife estimate of variance be used.

`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 `glm`. *kernel* is a user-written program or one of

`nwest` | `gallant` | `anderson`

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

`vce(unbiased)` specifies that the unbiased sandwich estimate of variance be used.

`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.

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

`scale(x2` | `dev` | `#)` overrides the default scale parameter. This option is allowed only with Hessian (information matrix) variance estimates.

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

`scale(x2)` specifies that the scale parameter be set to the Pearson χ^2 (or generalized χ^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.

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

`scale(#)` sets the scale parameter to `#`. For example, using `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 `streg, dist(exp) nohr` command (see [ST] **streg**) if all the observations are uncensored.

Reporting

`level(#)`; see [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 [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

Maximization

`ml` requests that optimization be carried out using Stata's `ml` commands and is the default.

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

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize**. These options are seldom used.

Setting the optimization method to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

If option `irls` is specified, only *maximize_options* `iterate()`, `nolog`, `trace`, and `ltolerance()` are allowed. With `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.

`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 `irls`).

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

The following options are available with `glm` but are not shown in the dialog box:

`noheader` suppresses the header information from the output. The coefficient table is still displayed.

`notable` suppresses the table of coefficients from the output. The header information is still displayed.

`nodisplay` suppresses the output. The iteration log is still displayed.

`collinear`, `coeflegend`; see [R] **Estimation options**. `collinear` is not allowed with `irls`.

Remarks and examples

stata.com

Remarks are presented under the following headings:

General use
Variance estimators
User-defined functions

General use

`glm` fits generalized linear models of y with covariates $\mathbf{x}$:

$$g\{E(y)\} = \mathbf{x}\boldsymbol{\beta}, \qquad y \sim F$$

$g(\,)$ 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

$$E(y) = \mathbf{x}\boldsymbol{\beta}, \qquad y \sim \text{Normal}$$

or linear regression. If $g(\,)$ is the logit function and y is distributed as Bernoulli, we have

$$\text{logit}\{E(y)\} = \mathbf{x}\boldsymbol{\beta}, \qquad y \sim \text{Bernoulli}$$

or logistic regression. If $g(\,)$ is the natural log function and y is distributed as Poisson, we have

$$\ln\{E(y)\} = \mathbf{x}\boldsymbol{\beta}, \qquad y \sim \text{Poisson}$$

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

Link function	`glm` option
identity	`link(identity)`
log	`link(log)`
logit	`link(logit)`
probit	`link(probit)`
complementary log-log	`link(cloglog)`
odds power	`link(opower #)`
power	`link(power #)`
negative binomial	`link(nbinomial)`
log-log	`link(loglog)`
log-complement	`link(logc)`

Define $\mu = E(y)$ and $\eta = g(\mu)$, meaning that $g(\cdot)$ maps $E(y)$ to $\eta = \mathbf{x}\boldsymbol{\beta} + \text{offset}$.

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 $\eta = \ln\{\mu/(1-\mu)\}$, the natural log of the odds.

`probit` is defined as $\eta = \Phi^{-1}(\mu)$, where $\Phi^{-1}(\,)$ is the inverse Gaussian cumulative.

`cloglog` is defined as $\eta = \ln\{-\ln(1-\mu)\}$.

`opower` is defined as $\eta = \left[\{\mu/(1-\mu)\}^n - 1\right]/n$, the power of the odds. The function is generalized so that `link(opower 0)` is equivalent to `link(logit)`, the natural log of the odds.

`power` is defined as $\eta = \mu^n$. Specifying `link(power 1)` is equivalent to specifying `link(identity)`. The power function is generalized so that $\mu^0 \equiv \ln(\mu)$. Thus, `link(power 0)` is equivalent to `link(log)`. Negative powers are, of course, allowed.

`nbinomial` is defined as $\eta = \ln\{\mu/(\mu+k)\}$, where $k = 1$ if `family(nbinomial)` is specified, $k = \#_k$ if `family(nbinomial` $\#_k$`)` is specified, and k is estimated via maximum likelihood if `family(nbinomial ml)` is specified.

`loglog` is defined as $\eta = -\ln\{-\ln(\mu)\}$.

`logc` is defined as $\eta = \ln(1-\mu)$.

The available distributional families are

Family	`glm` option
Gaussian (normal)	`family(gaussian)`
inverse Gaussian	`family(igaussian)`
Bernoulli/binomial	`family(binomial)`
Poisson	`family(poisson)`
negative binomial	`family(nbinomial)`
gamma	`family(gamma)`

`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` *varname*$_N$`)`. 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, *varname*$_N$ 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
`family(gaussian)`	`link(identity)`
`family(igaussian)`	`link(power -2)`
`family(binomial)`	`link(logit)`
`family(poisson)`	`link(log)`
`family(nbinomial)`	`link(log)`
`family(gamma)`	`link(power -1)`

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

	identity	log	logit	probit	cloglog	power	opower	nbinomial	loglog	logc
Gaussian	x	x				x				
inverse Gaussian	x	x				x				
binomial	x	x	x	x	x	x	x		x	x
Poisson	x	x				x				
negative binomial	x	x				x		x		
gamma	x	x				x				

❑ Technical note

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

`family()`	`link()`	Options	Equivalent Stata command
`gaussian`	`identity`	*nothing* | `irls` | `irls vce(oim)`	`regress`
`gaussian`	`identity`	`t(`*var*`) vce(hac nwest #) vfactor(`$\#_v$`)`	`newey, t(`*var*`) lag(#)` (see note 1)
`binomial`	`cloglog`	*nothing* | `irls vce(oim)`	`cloglog` (see note 2)
`binomial`	`probit`	*nothing* | `irls vce(oim)`	`probit` (see note 2)
`binomial`	`logit`	*nothing* | `irls` | `irls vce(oim)`	`logit` or `logistic` (see note 3)
`poisson`	`log`	*nothing* | `irls` | `irls vce(oim)`	`poisson` (see note 3)
`nbinomial`	`log`	*nothing* | `irls vce(oim)`	`nbreg` (see note 4)
`gamma`	`log`	`scale(1)`	`streg, dist(exp) nohr` (see note 5)

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 `glm` can be used to fit such models, using Stata's maximum likelihood `nbreg` command is probably better. In the GLM approach, you specify `family(nbinomial #k)` and then search for a $\#_k$ that results in the deviance-based dispersion being 1. You can also specify `family(nbinomial ml)` to estimate $\#_k$ via maximum likelihood, which will report the same value returned from `nbreg`. However, `nbreg` also reports a confidence interval for it; see [R] **nbreg** and Rogers (1993). Of course, `glm` allows links other than `log`, and for those links, including the canonical `nbinomial` link, you will need to use `glm`.

5. `glm` can be used to estimate parameters from exponential regressions, but this method requires specifying `scale(1)`. However, censoring is not available. Censored exponential regression may be modeled using `glm` with `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 `streg, dist(exp) nohr`. See [ST] **streg** for details about the `streg` command.

In general, where there is overlap between a capability of `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 `glm` framework. Nevertheless, `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 `glm` output to an equivalent Stata command, these tiny discrepancies arise for many reasons:

a. `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, `glm, 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 `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 `glm`. Adjusting the convergence criterion will resolve the difference. See [R] **ml** and [R] **Maximize** for more details.

❑

▷ Example 1

In example 1 of [R] **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 `glm`:

```
. use https://www.stata-press.com/data/r16/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                         Number of obs   =        189
Optimization     : ML                             Residual df     =        180
                                                  Scale parameter =          1
Deviance         =  201.4479911                   (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
```

low	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
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         =  201.4479911                   (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
```

low	Odds Ratio	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
age	.9732636	.0354759	-0.74	0.457	.9061578	1.045339
lwt	.9849634	.0068217	-2.19	0.029	.9716834	.9984249
race						
black	3.534767	1.860737	2.40	0.016	1.259736	9.918406
other	2.368079	1.039949	1.96	0.050	1.001356	5.600207
smoke	2.517698	1.00916	2.30	0.021	1.147676	5.523162
ptl	1.719161	.5952579	1.56	0.118	.8721455	3.388787
ht	6.249602	4.322408	2.65	0.008	1.611152	24.24199
ui	2.1351	.9808153	1.65	0.099	.8677528	5.2534
_cons	1.586014	1.910496	0.38	0.702	.1496092	16.8134

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
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
.	189	.	-100.724	9	219.448	248.6237

Note: BIC uses N = number of observations. See **[R] BIC 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/r16/ldose
. list, sep(4)
```

	ldose	n	r
1.	1.6907	59	6
2.	1.7242	60	13
3.	1.7552	62	18
4.	1.7842	56	28
5.	1.8113	63	52
6.	1.8369	59	53
7.	1.861	62	61
8.	1.8839	60	60

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:

$$\ln\{p/(1-p)\} = \beta_0 + \beta_1 X$$

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`:

```
. 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]
                                                  AIC             =   5.178781
Log likelihood   = -18.71512262                   BIC             =  -1.244442
```

r	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ldose	34.27034	2.912141	11.77	0.000	28.56265	39.97803
_cons	-60.71747	5.180713	-11.72	0.000	-70.87149	-50.56346

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 clog-log, function:

$$\ln\{-\ln(1-p)\} = \beta_0 + \beta_1 X$$

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
------------------------------------------------------------------------------
             |                 OIM
           r |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       ldose |    22.04118   1.793089    12.29   0.000     18.52679    25.55557
       _cons |   -39.57232   3.229047   -12.26   0.000    -45.90114   -33.24351
------------------------------------------------------------------------------
```

The complementary log-log model is preferred; the deviance for the logistic model, 11.23, is much higher than the deviance for the clog-log 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 clog-log 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/r16/beetle
. list, sep(0)
```

	beetle	ldose	n	r
1.	Destructive flour	1.6907	59	6
2.	Destructive flour	1.7242	60	13
3.	Destructive flour	1.7552	62	18
4.	Destructive flour	1.7842	56	28
5.	Destructive flour	1.8113	63	52
	(*output omitted*)			
23.	Mealworm	1.861	64	23
24.	Mealworm	1.8839	58	22

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 =  -76.94951
Iteration 2:   log likelihood = -76.945645
Iteration 3:   log likelihood = -76.945645
Generalized linear models                         Number of obs   =         24
Optimization     : ML                             Residual df     =         20
                                                  Scale parameter =          1
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
```

r	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
beetle						
Red flour	-.0910396	.1076132	-0.85	0.398	-.3019576	.1198783
Mealworm	-1.836058	.1307125	-14.05	0.000	-2.09225	-1.579867
ldose	19.41558	.9954265	19.50	0.000	17.46458	21.36658
_cons	-34.84602	1.79333	-19.43	0.000	-38.36089	-31.33116

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:

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

Iteration 0:   log likelihood = -67.270188
Iteration 1:   log likelihood = -65.149316
Iteration 2:   log likelihood = -65.147978
Iteration 3:   log likelihood = -65.147978

Generalized linear models                         Number of obs   =         24
Optimization     : ML                             Residual df     =         18
                                                  Scale parameter =          1
Deviance         =  50.16972096                   (1/df) Deviance =   2.787207
Pearson          =  49.28422567                   (1/df) Pearson  =   2.738013

Variance function: V(u) = u*(1-u/n)               [Binomial]
Link function    : g(u) = ln(-ln(1-u/n))          [Complementary log-log]

                                                  AIC             =   5.928998
Log likelihood   = -65.14797776                   BIC             =  -7.035248

------------------------------------------------------------------------------
             |                 OIM
           r |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      beetle |
  Red flour  |    -.79933   4.470882    -0.18   0.858    -9.562098    7.963438
   Mealworm  |   17.78741   4.586429     3.88   0.000     8.798172    26.77664
             |
       ldose |   22.04118   1.793089    12.29   0.000     18.52679    25.55557
             |
beetle#c.ldose |
  Red flour  |   .3838708   2.478477     0.15   0.877    -4.473855    5.241596
   Mealworm  |    -10.726   2.526412    -4.25   0.000    -15.67768   -5.774321
             |
       _cons |  -39.57232   3.229047   -12.26   0.000    -45.90114   -33.24351
------------------------------------------------------------------------------
```

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.

```
. use https://www.stata-press.com/data/r16/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)
Generalized linear models                         Number of obs   =         24
Optimization     : MQL Fisher scoring             Residual df     =         18
                   (IRLS EIM)                     Scale parameter =          1
Deviance         =  50.16972096                   (1/df) Deviance =   2.787207
Pearson          =  49.28422528                   (1/df) Pearson  =   2.738013
Variance function: V(u) = u*(1-u/n)               [Binomial]
Link function    : g(u) = ln(-ln(1-u/n))          [Complementary log-log]
                                                  BIC             =  -7.035248
```

r	EIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
beetle						
Red flour	-.79933	4.586649	-0.17	0.862	-9.788997	8.190337
Mealworm	17.78741	4.624834	3.85	0.000	8.7229	26.85192
ldose	22.04118	1.799356	12.25	0.000	18.5145	25.56785
beetle# c.ldose						
Red flour	.3838708	2.544068	0.15	0.880	-4.602411	5.370152
Mealworm	-10.726	2.548176	-4.21	0.000	-15.72033	-5.731665
_cons	-39.57232	3.240274	-12.21	0.000	-45.92314	-33.2215

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
```

r	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
beetle						
Red flour	-.79933	4.470882	-0.18	0.858	-9.562098	7.963438
Mealworm	17.78741	4.586429	3.88	0.000	8.798172	26.77664
ldose	22.04118	1.793089	12.29	0.000	18.52679	25.55557
beetle#c.ldose						
Red flour	.3838708	2.478477	0.15	0.877	-4.473855	5.241596
Mealworm	-10.726	2.526412	-4.25	0.000	-15.67768	-5.774321
_cons	-39.57232	3.229047	-12.26	0.000	-45.90114	-33.24351

This approach is identical to NR except for the convergence path. Because the clog-log 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
```

r	Robust Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
beetle						
Red flour	-.79933	5.733049	-0.14	0.889	-12.0359	10.43724
Mealworm	17.78741	5.158477	3.45	0.001	7.676977	27.89784
ldose	22.04118	.8998551	24.49	0.000	20.27749	23.80486
beetle#c.ldose						
Red flour	.3838708	3.174427	0.12	0.904	-5.837892	6.605633
Mealworm	-10.726	2.800606	-3.83	0.000	-16.21508	-5.236912
_cons	-39.57232	1.621306	-24.41	0.000	-42.75003	-36.39462

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".

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

r	Coef.	Semirobust Std. Err.	z	P>|z|	[95% Conf.	Interval]
beetle						
Red flour	-.79933	6.288963	-0.13	0.899	-13.12547	11.52681
Mealworm	17.78741	5.255307	3.38	0.001	7.487194	28.08762
ldose	22.04118	.9061566	24.32	0.000	20.26514	23.81721
beetle#c.ldose						
Red flour	.3838708	3.489723	0.11	0.912	-6.455861	7.223603
Mealworm	-10.726	2.855897	-3.76	0.000	-16.32345	-5.128542
_cons	-39.57232	1.632544	-24.24	0.000	-42.77205	-36.3726

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.

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

r	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
beetle						
Red flour	-.79933	6.664045	-0.12	0.905	-13.86062	12.26196
Mealworm	17.78741	6.838505	2.60	0.009	4.384183	31.19063
ldose	22.04118	3.572983	6.17	0.000	15.03826	29.0441
beetle#c.ldose						
Red flour	.3838708	3.700192	0.10	0.917	-6.868372	7.636114
Mealworm	-10.726	3.796448	-2.83	0.005	-18.1669	-3.285097
_cons	-39.57232	6.433101	-6.15	0.000	-52.18097	-26.96368

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.

```
. use https://www.stata-press.com/data/r16/idle2
. list usr idle time
```

	usr	idle	time
1.	0	100	1
2.	0	100	2
3.	0	97	3
4.	1	98	4
5.	2	94	5
	(*output omitted*)		
29.	1	98	29
30.	1	98	30

Examining *Methods and formulas* of [TS] **newey**, we see that the variance estimate is multiplied by a correction factor of $n/(n-k)$, where k is the number of regressors. `glm, vce(hac ...)` does not make this correction, so to get the same standard errors, we must use the `vfactor()` option within `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 = -71.743396
Generalized linear models                         Number of obs   =         30
Optimization     : ML                             Residual df     =         28
                                                  Scale parameter =   7.493297
Deviance         =  209.8123165                   (1/df) Deviance =   7.493297
Pearson          =  209.8123165                   (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
Log likelihood   = -71.74339627                   BIC             =   114.5788
```

usr	HAC Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
idle	-.2281501	.0690928	-3.30	0.001	-.3635694	-.0927307
_cons	23.13483	6.327033	3.66	0.000	10.73407	35.53558

The `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                         Number of obs   =         30
Optimization     : ML                             Residual df     =         28
                                                  Scale parameter =    .431296
Deviance         =  9.908506707                   (1/df) Deviance =   .3538752
Pearson          =  12.07628677                   (1/df) Pearson  =    .431296
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
------------------------------------------------------------------------------
             |                 HAC
         usr |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        idle |  -.0796609   .0184647    -4.31   0.000    -.115851   -.0434708
       _cons |   7.771011   1.510198     5.15   0.000    4.811078    10.73094
------------------------------------------------------------------------------
```

`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                         Number of obs   =         30
Optimization     : ML                             Residual df     =         28
                                                  Scale parameter =    .431296
Deviance         =  9.908506707                   (1/df) Deviance =   .3538752
Pearson          =  12.07628677                   (1/df) Pearson  =    .431296
Variance function: V(u) = u^2                     [Gamma]
Link function    : g(u) = ln(u)                   [Log]
                                                  AIC             =   4.196731
Log likelihood   = -60.95096484                   BIC             =  -85.32502
------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Normal-based
         usr |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        idle |  -.0796609    .016657    -4.78   0.000   -.1123081   -.0470137
       _cons |   7.771011   1.378037     5.64   0.000    5.070108    10.47192
------------------------------------------------------------------------------
```

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 progname
        version 16.1
        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 {
                /* set η=g(μ) */
                /* Intermediate calculations go here */
                generate double 'eta' = ...
                exit
        }
        if 'todo' == 1 {
                /* set μ=g^-1(η) */
                /* Intermediate calculations go here */
                generate double 'mu' = ...
                exit
        }
        if 'todo' == 2 {
                /* set return = ∂μ/∂η */
                /* Intermediate calculations go here */
                generate double 'return' = ...
                exit
        }
        if 'todo' == 3 {
                /* set return = ∂^2μ/∂η^2 */
                /* 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 progname
        version 16.1
        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 μ"
                exit
        }
        if `todo' == 0 {
                /*  set η to initial value.  */
                /*  Intermediate calculations go here  */
                generate double `eta' = ...
                exit
        }
        if `todo' == 1 {
                /*  set return = V(μ)  */
                /*  Intermediate calculations go here  */
                generate double `return' = ...
                exit
        }
        if `todo' == 2 {
                /*  set return = ∂V(μ)/∂μ  */
                /*  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 progname, rclass
        version 16.1
        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

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

▷ 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 ith observation for the Poisson family is given by

$$d_i^2 = \begin{cases} 2\widehat{\mu}_i & \text{if } y_i = 0 \\ 2\{y_i \ln(y_i/\widehat{\mu}_i) - (y_i - \widehat{\mu}_i)\} & \text{otherwise} \end{cases}$$

We now have enough information to write our own Poisson-log `glm` module. We create the file `mylog.ado`, which contains

```
program mylog
        version 16.1
        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 16.1
        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 {                            // Initialization of η; see note 2
                gen double `eta' = ln(`mu')
                exit
        }
```

```
        if `todo' == 1 {
                gen double `return' =  `mu'          //  V(μ) = μ
                exit
        }
        if `todo' == 2 {                             //  ∂V(μ)/∂μ
                gen byte `return' = 1
                exit
        }
        if `todo' == 3 {                             //  squared deviance, defined above
                local y "$SGLM_y"
                if "`y'" == "" {
                        local y "`e(depvar)'"
                }
                gen double `return' = cond(`y'==0, 2*`mu', /*
                          */ 2*(`y'*ln(`y'/`mu')-(`y'-`mu')))
                exit
        }
        if `todo' == 4 {                             //  Anscombe residual; see note 3
                local y "$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 {                             //  log likelihood; see note 4
                local y "$SGLM_y"
                if "`y'" == "" {
                        local y "`e(depvar)'"
                }
                gen double `return' = -`mu'+`y'*ln(`mu')-lngamma(`y'+1)
                exit
        }
        if `todo' == 6 {                             //  adjustment to residual; see note 5
                gen double `return' = 1/(6*sqrt(`mu'))
                exit
        }
        di as error "Unknown call to glm variance function"
        error 198
end
```

Notes:

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

2. Here the initial value of η 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 `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 `y` is set either to `SGLM_y` if it is in current estimation or to `e(depvar)` if this function is being accessed by `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/r16/airline
. list airline injuries n XYZowned
```

	airline	injuries	n	XYZowned
1.	1	11	0.0950	1
2.	2	7	0.1920	0
3.	3	7	0.0750	0
4.	4	19	0.2078	0
5.	5	9	0.1382	0
6.	6	4	0.0540	1
7.	7	3	0.1292	0
8.	8	1	0.0503	0
9.	9	3	0.0629	1

```
. generate lnN=ln(n)
. glm injuries XYZowned lnN, f(mypois) 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	OIM Coef.	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`.

```
program mylog2                                                  // <-- changed
        version 16.1
        args todo eta mu return
        if `todo' == -1 {
                global SGLM_lt "My Log, Version 2"              // <-- changed
                if "$SGLM_m" == "1" {                           // <-- changed
                        global SGLM_lf "ln(u)"                  // <-- changed
                }                                               // <-- changed
                else    global SGLM_lf "ln(u/$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"
        exit 198
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/r16/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 Coef.	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(phi)`	scale parameter
`e(aic)`	model AIC
`e(bic)`	model BIC
`e(ll)`	log likelihood, if NR
`e(N_clust)`	number of clusters
`e(chi2)`	χ^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(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
`e(wtype)`	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 χ^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

e(properties)	b V
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient 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.

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(disp)	dispersion parameter
e(bic)	model BIC
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

e(wtype)	weight type
e(wexp)	weight expression
e(clustvar)	name of cluster variable
e(offset)	linear offset variable
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(properties)	b V
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

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

The canonical reference on GLM is McCullagh and Nelder (1989). The term "generalized linear model" is from Nelder and Wedderburn (1972). Many people use the acronym GLIM for GLM models because of the classic GLM software tool GLIM, by Baker and Nelder (1985). See Dobson and Barnett (2018) for a concise introduction and overview. See Rabe-Hesketh and Everitt (2007) for more examples of GLM using Stata. Hoffmann (2004) focuses on applying generalized linear models, using real-world datasets, along with interpreting computer output, which for the most part is obtained using Stata.

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 v_j for the jth 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 `fweights` are specified and $N = M$ otherwise. Each IRLS step is performed by `regress` using w_j as the weights.

Let d_j^2 denote the squared deviance residual for the jth observation:

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

For the Bernoulli family (binomial with denominator 1),

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

For the binomial family with denominator m_j,

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

For the Poisson family,

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

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

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

For the negative binomial,

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

Let $\phi = 1$ if the scale parameter is set to one; otherwise, define $\phi = \widehat{\phi}_0(n-k)/n$, where $\widehat{\phi}_0$ 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 jth observation:

For the Gaussian family,

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

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

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

For the Poisson family,

$$\ln L_j = \phi\,[y_j\ln(\widehat{\mu}_j) - \widehat{\mu}_j - \ln\{\Gamma(y_j+1)\}]$$

For the gamma family, $\ln L_j = -y_j/\widehat{\mu}_j + \ln(1/\widehat{\mu}_j)$.

For the inverse Gaussian,

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

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

$$\begin{aligned}\ln L_j =& \phi\,[\ln\{\Gamma(m+y_j)\} - \ln\{\Gamma(y_j+1)\} - \ln\{\Gamma(m)\}\\ & -m\ln(1+\widehat{\mu}_j/m) + y_j\ln\{\widehat{\mu}_j/(\widehat{\mu}_j+m)\}]\end{aligned}$$

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

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

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

Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, ed. B. N. Petrov and F. Csaki, 267–281. Budapest: Akailseoniai–Kiudo.

Anscombe, F. J. 1953. Contribution of discussion paper by H. Hotelling "New light on the correlation coefficient and its transforms". *Journal of the Royal Statistical Society, Series B* 15: 229–230.

Baker, R. J., and J. A. Nelder. 1985. *The Generalized Linear Interactive Modelling System, Release 3.77*. Oxford: Numerical Algorithms Group.

Basu, A. 2005. Extended generalized linear models: Simultaneous estimation of flexible link and variance functions. *Stata Journal* 5: 501–516.

Berndt, E. K., B. H. Hall, R. E. Hall, and J. A. Hausman. 1974. Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement* 3/4: 653–665.

Cummings, P. 2009. Methods for estimating adjusted risk ratios. *Stata Journal* 9: 175–196.

Discacciati, A., and M. Bottai. 2017. Instantaneous geometric rates via generalized linear models. *Stata Journal* 17: 358–371.

Dobson, A. J., and A. G. Barnett. 2018. *An Introduction to Generalized Linear Models*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.

Hardin, J. W., and J. M. Hilbe. 2018. *Generalized Linear Models and Extensions*. 4th ed. College Station, TX: Stata Press.

Hilbe, J. M. 1993. sg16: Generalized linear models. *Stata Technical Bulletin* 11: 20–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 149–159. College Station, TX: Stata Press.

——. 2009. *Logistic Regression Models*. Boca Raton, FL: Chapman & Hill/CRC.

——. 2014. *Modeling Count Data*. New York: Cambridge University Press.

Hoffmann, J. P. 2004. *Generalized Linear Models: An Applied Approach*. Boston: Pearson.

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman & Hall/CRC.

Nelder, J. A. 1975. Robert William MacLagan Wedderburn, 1947–1975. *Journal of the Royal Statistical Society, Series A* 138: 587.

Nelder, J. A., and R. W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society, Series A* 135: 370–384.

Newey, W. K., and K. D. West. 1987. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica* 55: 703–708.

Newson, R. B. 1999. sg114: rglm—Robust variance estimates for generalized linear models. *Stata Technical Bulletin* 50: 27–33. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 181–190. College Station, TX: Stata Press.

——. 2004. Generalized power calculations for generalized linear models and more. *Stata Journal* 4: 379–401.

Orsini, N., R. Bellocco, and P. C. Sjölander. 2013. Doubly robust estimation in generalized linear models. *Stata Journal* 13: 185–205.

Parner, E. T., and P. K. Andersen. 2010. Regression analysis of censored data using pseudo-observations. *Stata Journal* 10: 408–422.

Pregibon, D. 1980. Goodness of link tests for generalized linear models. *Applied Statistics* 29: 15–24.

Rabe-Hesketh, S., and B. S. Everitt. 2007. *A Handbook of Statistical Analyses Using Stata*. 4th ed. Boca Raton, FL: Chapman & Hall/CRC.

Rabe-Hesketh, S., A. Pickles, and C. Taylor. 2000. sg129: Generalized linear latent and mixed models. *Stata Technical Bulletin* 53: 47–57. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 293–307. College Station, TX: Stata Press.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2002. Reliable estimation of generalized linear mixed models using adaptive quadrature. *Stata Journal* 2: 1–21.

Raftery, A. E. 1995. Bayesian model selection in social research. In Vol. 25 of *Sociological Methodology*, ed. P. V. Marsden, 111–163. Oxford: Blackwell.

Rogers, W. H. 1993. sg16.4: Comparison of nbreg and glm for negative binomial. *Stata Technical Bulletin* 16: 7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 82–84. College Station, TX: Stata Press.

Royston, P. 1994. sg22: Generalized linear models: Revision of glm. *Stata Technical Bulletin* 18: 6–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 112–121. College Station, TX: Stata Press.

Sasieni, P. D. 2012. Age–period–cohort models in Stata. *Stata Journal* 12: 45–60.

Schonlau, M. 2005. Boosted regression (boosting): An introductory tutorial and a Stata plugin. *Stata Journal* 5: 330–354.

Senn, S. J. 2003. A conversation with John Nelder. *Statistical Science* 18: 118–131.

Williams, R. 2010. Fitting heterogeneous choice models with oglm. *Stata Journal* 10: 540–567.

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 model

[MI] **Estimation** — Estimation commands for use with mi estimate

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtgee** — Fit population-averaged panel-data models by using GEE

[U] 20 Estimation and postestimation commands