zip — Zero-inflated Poisson regression

Description

zip fits a zero-inflated Poisson (ZIP) model to count data with excess zero counts. The ZIP model assumes that the excess zero counts come from a logit or probit model and the remaining counts come from a Poisson model.

Quick start

Zero-inflated Poisson model of \( y \) on \( x_1 \) and \( x_2 \) with inflation modeled using \( x_3 \)

\[
zip \ y \ x_1 \ x_2, \ inflate(x_3)
\]

Use a probit model instead of a logit model to predict excess zeros

\[
zip \ y \ x_1 \ x_2, \ inflate(x_3) \ probit
\]

Menu

Statistics > Count outcomes > Zero-inflated Poisson regression
Syntax

zip depvar [ indepvars ] [ if ] [ in ] [ weight ],
   inflate(varlist[, offset(varname)] | _cons) [ options ]

options                          Description

Model

* inflate()                      equation that determines whether the count is zero
   noconstant                    suppress constant term
   exposure(varname_e)           include ln(varname_e) in model with coefficient constrained to 1
   offset(varname_o)            include varname_o in model with coefficient constrained to 1
   constraints(constraints)     apply specified linear constraints
   probit                        use probit model to characterize excess zeros; default is logit

SE/Robust
   vce(vcetype)                  vcetype may be oim, robust, cluster clustvar, opg, bootstrap, or jackknife

Reporting
   level(#)                      set confidence level; default is level(95)
   irr                           report incidence-rate ratios
   noconsreport                  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
   maximize_options             control the maximization process; seldom used
   collinear                    keep collinear variables
   coeflegend                   display legend instead of statistics

* inflate(varlist[, offset(varname)] | _cons) is required.
indepvars and varlist may contain factor variables; see [U] 11.4.3 Factor variables.
bayes, bootstrap, by, collect, fp, jackknife, rolling, statsby, and svy are allowed; see [U] 11.1.10 Prefix commands. For more details, see [BAYES] bayes: zip.
Weights are not allowed with the bootstrap prefix; see [R] bootstrap.
vce() and weights are not allowed with the svy prefix; see [SVY] svy.
fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight.
collinear and coeflegend do not appear in the dialog box.
See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Options

inflate(varlist[, offset(varname)] | _cons) specifies the equation that determines whether the observed count is zero. Conceptually, omitting inflate() would be equivalent to fitting the model with poisson; see [R] poisson.
inflates() specifies the variables in the equation. You may optionally include an offset for this varlist.

inflate(_cons) specifies that the equation determining whether the count is zero contains only an intercept. To run a zero-inflated model of depvar with only an intercept in both equations, type zip depvar, inflate(_cons).

noconstant, exposure(varname_e), offset(varname_o), constraints(constraints); see [R] Estimation options.

probit requests that a probit, instead of logit, model be used to characterize the excess zeros in the data.

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.

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

irr reports estimated coefficients transformed to incidence-rate ratios, that is, $e^b$ rather than $b$. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. irr may be specified at estimation or when replaying previously estimated results.

nocnsreport; see [R] Estimation options.

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

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 type to technique(bhhh) resets the default vcetype to vce(opg).

The following options are available with zip but are not shown in the dialog box: collinear, coeflegend; see [R] Estimation options.

Remarks and examples stata.com

Zero-inflated Poisson (ZIP) models address the case when the data contain a higher fraction of zeros than is likely to be generated from a Poisson model. Having a large proportion of zero observations, in itself, does not necessarily mean that we have the excess zeros problem. For instance, a Poisson model with a mean value of 0.2 predicts that $P(Y = 0) = \exp(-0.2) \approx 0.82$. However, the range of possible outcomes is restricted because of the small variance, 0.2 (mean and variance are equal for a Poisson model), as shown by $P(Y > 3) \approx 0.00006$. Unlike Poisson models, ZIP models allow us to have a large fraction of zeros without restricting the range of outcomes.
ZIP models assume that an observation is 0 with a probability $p$ or is a realization of a Poisson random variable, which can also be 0, with a probability $1 - p$. For instance, you might count how many fish each visitor to a park catches. Many visitors may catch zero, because they do not fish (as opposed to being unsuccessful). Using a logit or probit model, you may model the probability $p$ of whether a person does not fish depending on several covariates related to fishing. Using a Poisson distribution, you may model how many fish a person catches depending on several covariates having to do with the success of catching fish (type of lure or bait, time of day, temperature, season, etc.). This is the type of data for which the zip command is useful.


Example 1: Fitting a ZIP model

We have fictional data on the number of fish caught (count) by visitors to a national park on a particular day. Some of the visitors do not fish, but we do not have the data on whether a person fished; we merely have data on how many fish were caught together with several covariates.

Variable count exhibits an excess of zero observations (142 of 250 observations), beyond what would be expected from a Poisson model. We suspect that the number of zeros may be inflated because many visitors are not fishing. That is, a zero observation may be the result of a visitor who was unfortunate and caught no fish but may also be because the visitor did not fish. A standard Poisson model (see [R] poisson) treats these two types of zero observations as a homogeneous group, which typically leads to biased statistical results. We would like to distinguish between the two types of zeros and possibly draw inference for them separately (see example 1 in [R] zip postestimation).

The zip command allows us to model the two types of zeros. First, using the required option inflate(), we model whether a visitor fishes as a function of the number of children accompanying him or her (child) and whether he or she is camping (camper). Next, we assume the response variable, count, depends on whether the visitor used a live bait (livebait) and the number of persons (persons), which includes the visitor and any adults or children accompanying him. Note that persons is always greater than child.
. use https://www.stata-press.com/data/r17/fish
(Fictional fishing data)
. zip count persons livebait, inflate(child camper)

Fitting constant-only model:
Iteration 0:  log likelihood = -1347.807
Iteration 1:  log likelihood = -1305.3245
Iteration 2:  log likelihood = -1104.3005
Iteration 3:  log likelihood = -1103.9426
Iteration 4:  log likelihood = -1103.9425

Fitting full model:
Iteration 0:  log likelihood = -1103.9425
Iteration 1:  log likelihood = -896.2346
Iteration 2:  log likelihood = -851.61723
Iteration 3:  log likelihood = -850.70435
Iteration 4:  log likelihood = -850.70142
Iteration 5:  log likelihood = -850.70142

Zero-inflated Poisson regression
Number of obs = 250
Inflation model: logit
Nonzero obs = 108
Zero obs = 142
LR chi2(2) = 506.48
Prob > chi2 = 0.0000

Log likelihood = -850.7014

| count      | Coefficient | Std. err. | z    | P>|z|   | [95% conf. interval] |
|------------|-------------|-----------|------|-------|---------------------|
| count      |             |           |      |       |                     |
| persons    | .8068853    | .0453288  | 17.80| 0.000 | .7180424 .8957281   |
| livebait   | 1.757289    | .2446082  | 7.18 | 0.000 | 1.277866 2.236713   |
| _cons      | -2.178472   | .2860289  | -7.62| 0.000 | -2.739078 -1.617865 |

| inflate    |             |           |      |       |                     |
| child      | 1.602571    | .2797719  | 5.73 | 0.000 | 1.054228 2.150913   |
| camper     | -1.015698   | .365259   | -2.78| 0.005 | -1.731593 -.2998038 |
| _cons      | -.4922872   | .3114562  | -1.58| 0.114 | -1.10273 .1181558  |

Coefficients in the upper half of the table correspond to the Poisson model for individuals who fished. For instance, among visitors who fished, using a live bait increases the expected number of caught fish by a factor of \( \exp(1.7572) \approx 5.8 \), holding other covariates constant.

Example 2: Comparing model fit

When you have count data, you may want to test whether a conventional count data model or a zero-inflated count data model is preferable. The classical likelihood-ratio test cannot be used here because the models are not nested. But we can use information criteria such as the AIC and BIC to check whether the standard or zero-inflated model is more appropriate.

Continuing with our fishing example, let’s check whether the standard Poisson or ZIP model is more appropriate for our data. First, we store the estimation results from the previous ZIP model by typing

. estimates store zip
Next, we fit the Poisson model corresponding to the main equation of the ZIP model and store its results as `pois`:

```
. poisson count persons livebait
(output omitted)
. estimates store pois
```

We use `estimates stats` to display the AIC and BIC values for the two models.

```
. estimates stats pois zip
```

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>pois</td>
<td>250</td>
<td>-1647.716</td>
<td>-1312.178</td>
<td>3</td>
<td>2630.356</td>
<td>2640.92</td>
</tr>
<tr>
<td>zip</td>
<td>250</td>
<td>-1103.942</td>
<td>-850.7014</td>
<td>6</td>
<td>1713.403</td>
<td>1734.532</td>
</tr>
</tbody>
</table>

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

The ZIP model has smaller AIC and BIC values; we thus conclude that it fits our data better than the standard Poisson model.

---

**Stored results**

`zip` stores the following in `e()`:

Scalars

<table>
<thead>
<tr>
<th>Scalar</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e(N)</td>
<td>number of observations</td>
</tr>
<tr>
<td>e(N_zero)</td>
<td>number of zero observations</td>
</tr>
<tr>
<td>e(k)</td>
<td>number of parameters</td>
</tr>
<tr>
<td>e(k_eq)</td>
<td>number of equations in <code>e(b)</code></td>
</tr>
<tr>
<td>e(k_eq_model)</td>
<td>number of equations in overall model test</td>
</tr>
<tr>
<td>e(k_dv)</td>
<td>number of dependent variables</td>
</tr>
<tr>
<td>e(df_m)</td>
<td>model degrees of freedom</td>
</tr>
<tr>
<td>e(ll)</td>
<td>log likelihood</td>
</tr>
<tr>
<td>e(ll_0)</td>
<td>log likelihood, constant-only model</td>
</tr>
<tr>
<td>e(N_clust)</td>
<td>number of clusters</td>
</tr>
<tr>
<td>e(chi2)</td>
<td>( \chi^2 )</td>
</tr>
<tr>
<td>e(p)</td>
<td>( p )-value for model test</td>
</tr>
<tr>
<td>e(rank)</td>
<td>rank of <code>e(V)</code></td>
</tr>
<tr>
<td>e(ic)</td>
<td>number of iterations</td>
</tr>
<tr>
<td>e(rc)</td>
<td>return code</td>
</tr>
<tr>
<td>e(converged)</td>
<td>1 if converged, 0 otherwise</td>
</tr>
</tbody>
</table>

Macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e(cmd)</td>
<td><code>zip</code></td>
</tr>
<tr>
<td>e(cmdline)</td>
<td>command as typed</td>
</tr>
<tr>
<td>e(depvar)</td>
<td>name of dependent variable</td>
</tr>
<tr>
<td>e(inflate)</td>
<td><code>logit</code> or <code>probit</code></td>
</tr>
<tr>
<td>e(wtype)</td>
<td>weight type</td>
</tr>
<tr>
<td>e(wexp)</td>
<td>weight expression</td>
</tr>
<tr>
<td>e(title)</td>
<td>title in estimation output</td>
</tr>
<tr>
<td>e(clustvar)</td>
<td>name of cluster variable</td>
</tr>
<tr>
<td>e(offset1)</td>
<td>offset</td>
</tr>
<tr>
<td>e(offset2)</td>
<td>offset for <code>inflate()</code></td>
</tr>
<tr>
<td>e(chi2type)</td>
<td>Wald or LR: type of model ( \chi^2 ) test</td>
</tr>
<tr>
<td>e(vce)</td>
<td><code>vcetype</code> specified in <code>vce()</code></td>
</tr>
<tr>
<td>e(vcetype)</td>
<td>title used to label Std. err.</td>
</tr>
<tr>
<td>e(opt)</td>
<td>type of optimization</td>
</tr>
</tbody>
</table>
Consider the formulation of a zero-inflated model as presented in Lambert (1992). Define

\[ \xi_j^\beta = x_j^T \beta + \text{offset}_j^\beta \]
\[ \xi_j^\gamma = z_j^T \gamma + \text{offset}_j^\gamma \]
\[ \lambda_j = \exp(\xi_j^\beta) \]
\[ F_j = F(\xi_j^\gamma) \]

where \( F(\cdot) \) is the inverse of the logit function or, if the \texttt{probit} option was specified, the inverse of the probit function (or the standard normal cumulative distribution function). All subjects are assumed to be independent with the \( j \)th response determined as follows:

\[ Y_j = 0 \quad \text{with probability} \quad F_j \]
\[ Y_j \sim \text{Poisson}(\lambda_j) \quad \text{with probability} \quad 1 - F_j \]

In other words,

\[ \Pr(Y_j = 0|x_j, z_j) = F_j + (1 - F_j) \exp(-\lambda_j) \]
\[ \Pr(Y_j = n|x_j, z_j) = (1 - F_j) \exp(-\lambda_j) \frac{\lambda_j^n}{n!} \quad \text{for} \quad n = 1, 2, \ldots \]
The `zip` command maximizes the log-likelihood $\ln L$, defined by

$$
\ln L = \sum_{j \in S} w_j \ln \{F_j + (1 - F_j) \exp(-\lambda_j)\} \\
+ \sum_{j \notin S} w_j \{\ln(1 - F_j) - \lambda_j + \xi_j y_j - \ln(y_j!}\}
$$

where $w_j$ are the weights and $S$ is the set of observations for which the observed outcome $y_j = 0$.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster clustvar)`, respectively. See [P] `_robust`, particularly Maximum likelihood estimators and Methods and formulas.

`zip` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] Variance estimation.

References


Also see

[R] `zip postestimation` — Postestimation tools for `zip`

[R] `zinb` — Zero-inflated negative binomial regression

[R] `nbreg` — Negative binomial regression

[R] `poisson` — Poisson regression

[R] `tnbreg` — Truncated negative binomial regression

[R] `tpoisson` — Truncated Poisson regression

[BAYES] `bayes: zip` — Bayesian zero-inflated Poisson regression

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

[XT] `xtpoisson` — Fixed-effects, random-effects, and population-averaged Poisson models

[U] 20 Estimation and postestimation commands