
spglm Documentation

Release 1.0.5

pysal developers

Sep 27, 2018

Contents:

1	Installing released version	3
2	Installing development version	5
3	API reference	7
3.1	GLM	7
3.1.1	spglm.glm.GLM	7
3.1.2	spglm.glm.GLMResults	9
4	References	13

Sparse generalize linear models (spglm)

spglm supports python [3.5](#) and [3.6](#) only. Please make sure that you are operating in a python 3 environment.

CHAPTER 1

Installing released version

spglm is available on the [Python Package Index](#). Therefore, you can either install directly with *pip* from the command line:

```
pip install -U spglm
```

or download the source distribution (.tar.gz) and decompress it to your selected destination. Open a command shell and navigate to the decompressed folder. Type:

```
pip install .
```


CHAPTER 2

Installing development version

Potentially, you might want to use the newest features in the development version of `spglm` on github - [pysal/spglm](#) while have not been incorporated in the Pypi released version. You can achieve that by installing [pysal/spglm](#) by running the following from a command shell:

```
pip install https://github.com/pysal/spglm/archive/master.zip
```

You can also [fork](#) the [pysal/spglm](#) repo and create a local clone of your fork. By making changes to your local clone and submitting a pull request to [pysal/spglm](#), you can contribute to the `mgwr` development.

3.1 GLM

<code>spglm.glm.GLM(y, X[, family, offset, y_fix, ...])</code>	Generalised linear models.
<code>spglm.glm.GLMResults(model, params, mu, w)</code>	Results of estimated GLM and diagnostics.

3.1.1 spglm.glm.GLM

class `spglm.glm.GLM`(*y*, *X*, *family*=<spglm.family.Gaussian object>, *offset*=None, *y_fix*=None, *constant*=True)
 Generalised linear models. Can currently estimate Guassian, Poisson and Logisitic regression coefficients. GLM object prepares model input and fit method performs estimation which then returns a GLMResults object.

Parameters

- y** [array]
n*1, dependent variable.
- X** [array] n*k, independent variable, exlcuding the constant.
- family** [string] Model type: 'Gaussian', 'Poisson', 'Binomial'
- offset** [array] n*1, the offset variable at the ith location. For Poisson model this term is often the size of the population at risk or the expected size of the outcome in spatial epidemiology. Default is None where Ni becomes 1.0 for all locations.
- y_fix** [array] n*1, the fix intercept value of y

Examples

```
>>> import libpysal
>>> from spglm.glm import GLM
>>> from spglm.family import Gaussian
>>> db = libpysal.io.open(libpysal.examples.get_path('columbus.dbf'), 'r')
>>> y = np.array(db.by_col("HOVAL"))
>>> y = np.reshape(y, (49,1))
>>> X = []
>>> X.append(db.by_col("INC"))
>>> X.append(db.by_col("CRIME"))
>>> X = np.array(X).T
>>> model = GLM(y, X, family=Gaussian())
>>> results = model.fit()
>>> results.params
array([46.42818268,  0.62898397, -0.48488854])
```

Attributes

y [array]

n*1, dependent variable.

X [array] n*k, independent variable, including constant.

family [string] Model type: 'Gaussian', 'Poisson', 'logistic'

n [integer] Number of observations

k [integer] Number of independent variables

df_model [float] k-1, where k is the number of variables (including intercept)

df_residual [float] observations minus variables (n-k)

mean_y [float] Mean of y

std_y [float] Standard deviation of y

fit_params [dict] Parameters passed into fit method to define estimation routine.

Methods

<code>fit([ini_betas, tol, max_iter, solve])</code>	Method that fits a model with a particular estimation routine.
---	--

df_model	
df_resid	

`__init__(y, X, family=<spglm.family.Gaussian object>, offset=None, y_fix=None, constant=True)`
Initialize class

Methods

<code>__init__(y, X[, family, offset, y_fix, constant])</code>	Initialize class
<code>df_model()</code>	

Continued on next page

Table 3 – continued from previous page

<code>df_resid()</code>	
<code>fit([ini_betas, tol, max_iter, solve])</code>	Method that fits a model with a particular estimation routine.

Attributes

<code>mean_y</code>
<code>std_y</code>

3.1.2 spglm.glm.GLMResults

class `spglm.glm.GLMResults` (*model, params, mu, w*)

Results of estimated GLM and diagnostics.

Parameters

model [GLM object]

Pointer to GLM object with estimation parameters.

params [array] $k \times 1$, estimated coefficients

mu [array] $n \times 1$, predicted y values.

w [array] $n \times 1$, final weight used for iwl

Examples

```
>>> import libpysal
>>> from spglm.glm import GLM, GLMResults
>>> from spglm.family import Gaussian
>>> db = libpysal.io.open(libpysal.examples.get_path('columbus.dbf'), 'r')
>>> y = np.array(db.by_col("HOVAL"))
>>> y = np.reshape(y, (49,1))
>>> X = []
>>> X.append(db.by_col("INC"))
>>> X.append(db.by_col("CRIME"))
>>> X = np.array(X).T
>>> model = GLM(y, X, family=Gaussian())
>>> results1 = model.fit()
>>> results1.aic
408.73548964604873
>>> model = results1.model
>>> params = results1.params.flatten()
>>> predy = results1.mu
>>> w = results1.w
>>> results2 = GLMResults(model, params, predy, w)
>>> results2.aic
408.73548964604873
```

Attributes

model [GLM Object]

Points to GLM object for which parameters have been estimated.

y [array] $n \times 1$, dependent variable.

x [array] $n \times k$, independent variable, including constant.

family [string] Model type: 'Gaussian', 'Poisson', 'Logistic'

n [integer] Number of observations

k [integer] Number of independent variables

df_model [float] $k-1$, where k is the number of variables (including intercept)

df_residual [float] observations minus variables ($n-k$)

fit_params [dict] parameters passed into fit method to define estimation routine.

scale [float] sigma squared used for subsequent computations.

params [array] $n \times k$, estimated beta coefficients

w [array] $n \times 1$, final weight values of x

mu [array] $n \times 1$, predicted value of y (i.e., fitted values)

cov_params [array] Variance covariance matrix ($k \times k$) of betas

bse [array] $k \times 1$, standard errors of betas

pvalues [array] $k \times 1$, two-tailed pvalues of parameters

tvalues [array] $k \times 1$, the tvalues of the standard errors

null [array] $n \times 1$, predicted values of y for null model

deviance [float] value of the deviance function evaluated at params; see family.py for distribution-specific deviance

null_deviance [float] value of the deviance function for the model fit with a constant as the only regressor

llf [float] value of the loglikelihood function evaluated at params; see family.py for distribution-specific loglikelihoods

llnull [float] value of log-likelihood function evaluated at null

aic [float] AIC

bic [float] BIC

D2 [float] percent deviance explained

adj_D2 [float] adjusted percent deviance explained

pseudo_R2 [float] McFadden's pseudo R2 (coefficient of determination)

adj_pseudoR2 [float] adjusted McFadden's pseudo R2

tr_S : trace of the hat matrix **S** **resid_response** : array
response residuals; defined as $y - \mu$

resid_pearson [array] Pearson residuals; defined as $(y - \mu) / \sqrt{\text{VAR}(\mu)}$ where VAR is the distribution specific variance function; see family.py and varfuncs.py for more information.

resid_working [array] Working residuals; the working residuals are defined as $\text{resid_response} / \text{link}'(\mu)$; see links.py for the derivatives of the link functions.

resid_anscombe [array] Anscombe residuals; see family.py for distribution-specific Anscombe residuals.

resid_deviance [array] deviance residuals; see family.py for distribution-specific deviance residuals.

pearson_chi2 [float] chi-Squared statistic is defined as the sum of the squares of the Pearson residuals

normalized_cov_params [array] $k \times k$, approximates $[X.T * X]^{-1}$

Methods

<code>conf_int([alpha, cols, method])</code>	Returns the confidence interval of the fitted parameters.
<code>cov_params([r_matrix, column, scale, cov_p, ...])</code>	Returns the variance/covariance matrix.
<code>tvalues()</code>	Return the t-statistic for a given parameter estimate.

D2	
adj_D2	
adj_pseudoR2	
aic	
bic	
bse	
deviance	
df_model	
df_resid	
initialize	
llf	
llnull	
normalized_cov_params	
null	
null_deviance	
pearson_chi2	
pseudoR2	
pvalues	
resid_anscombe	
resid_deviance	
resid_pearson	
resid_response	
resid_working	
scale	
tr_S	

`__init__(model, params, mu, w)`
Initialize self. See help(type(self)) for accurate signature.

Methods

D2()	
<code>__init__</code> (model, params, mu, w)	Initialize self.
adj_D2()	
adj_pseudoR2()	
aic()	
bic()	
bse()	
conf_int([alpha, cols, method])	Returns the confidence interval of the fitted parameters.
cov_params([r_matrix, column, scale, cov_p, ...])	Returns the variance/covariance matrix.
deviance()	
df_model()	
df_resid()	
initialize(model, params, **kwd)	
llf()	
llnull()	
normalized_cov_params()	
null()	
null_deviance()	
pearson_chi2()	
pseudoR2()	
pvalues()	
resid_anscombe()	
resid_deviance()	
resid_pearson()	
resid_response()	
resid_working()	
scale()	
tr_S()	
tvalues()	Return the t-statistic for a given parameter estimate.

Attributes

use_t

CHAPTER 4

References

Symbols

`__init__()` (spglm.glm.GLM method), [8](#)

`__init__()` (spglm.glm.GLMResults method), [11](#)

G

GLM (class in spglm.glm), [7](#)

GLMResults (class in spglm.glm), [9](#)