spint Documentation

Release 1.0.6

pysal developers

Dec 30, 2019

CONTENTS:

1 Installing released version **3**

2 Installing development version **5**

3 API reference **7**

3.1 Gravity-type spatial interaction models 7

3.1.1 spint.gravity.BaseGravity 7

3.1.2 spint.gravity.Gravity 10

3.1.3 spint.gravity.Production 12

3.1.4 spint.gravity.Attraction 15

3.1.5 spint.gravity.Doubly 17

3.2 Tests for overdispersion 20

3.2.1 spint.dispersion.alpha_disp 20

3.2.2 spint.dispersion.phi_disp 21

4 References **23**

Index **25**

SPatial INTeraction models (SPINT)

spint supports python 3.5 and 3.6 only. Please make sure that you are operating in a python 3 environment.

CHAPTER

ONE

INSTALLING RELEASED VERSION

spint is available on the Python Package Index. Therefore, you can either install directly with *pip* from the command line:

```
pip install -U spint
```

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

TWO

INSTALLING DEVELOPMENT VERSION

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

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

You can also fork the pysal/spint repo and create a local clone of your fork. By making changes to your local clone and submitting a pull request to pysal/spint, you can contribute to the mgwr development.

CHAPTER

THREE

API REFERENCE

3.1 Gravity-type spatial interaction models

spint.gravity.BaseGravity(flows, cost[, ...])	Base class to set up gravity-type spatial interaction models and dispatch estimaton technqiues.
spint.gravity.Gravity(flows, o_vars, d_vars, ...)	Unconstrained (traditional gravity) gravity-type spatial interaction model
spint.gravity.Production(flows, origins, ...)	Production-constrained (origin-constrained) gravity-type spatial interaction model
spint.gravity.Attraction(flows, ... [, ...])	Attraction-constrained (destination-constrained) gravity-type spatial interaction model
spint.gravity.Doubly(flows, origins, ... [, ...])	Doubly-constrained gravity-type spatial interaction model

3.1.1 spint.gravity.BaseGravity

class spint.gravity.**BaseGravity**(*flows*, *cost*, *cost_func='pow'*, *o_vars=None*, *d_vars=None*, *origins=None*, *destinations=None*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)

Base class to set up gravity-type spatial interaction models and dispatch estimaton technqiues.

Parameters

flows [array of integers] n x 1; observed flows between O origins and D destinations

origins [array of strings] n x 1; unique identifiers of origins of n flows

destinations [array of strings] n x 1; unique identifiers of destinations of n flows

cost [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cost_func [string or function that has scalar input and output] functional form of the cost function; 'exp' | 'pow' | custom function

o_vars [array (optional)] n x p; p attributes for each origin of n flows; default is None

d_vars [array (optional)] n x p; p attributes for each destination of n flows; default is None

constant [boolean] True to include intercept in model; True by default

framework [string] estimation technique; currently only 'GLM' is avaialble

Quasi [boolean] True to estimate QuasiPoisson model; should result in same parameters as Poisson but with altered covariance; default to true which estimates Poisson model

SF [array] n x 1; eigenvector spatial filter to include in the model; default to None which does not include a filter; not yet implemented

CD [array] n x 1; competing destination term that accounts for the likelihood that alternative destinations are considered along with each destination under consideration for every OD pair; defaults to None which does not include a CD term; not yet implemented

Lag [W object] spatial weight for n observations (OD pairs) used to construct a spatial autoregressive model and estimator; defaults to None which does not include an autoregressive term; not yet implemented

Attributes

f [array] n x 1; observed flows; dependent variable; y

n [integer] number of observations

k [integer] number of parameters

c [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cf [function] cost function; used to transform cost variable

ov [array] n x p(o); p attributes for each origin of n flows

dv [array] n x p(d); p attributes for each destination of n flows

constant [boolean] True to include intercept in model; True by default

y [array] n x 1; dependent variable used in estimation including any transformations

X [array] n x k, design matrix used in estimation

params [array] n x k, k estimated beta coefficients; k = p(o) + p(d) + 1

yhat [array] n x 1, predicted value of y (i.e., fittedvalues)

cov_params [array] Variance covariance matrix (k x k) of betas

std_err [array] k x 1, standard errors of betas

pvalues [array] k x 1, two-tailed pvalues of parameters

tvalues [array] k x 1, the tvalues of the standard errors

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

resid_dev [array] n x 1, residual deviance of model

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

llnull [float] value of the loglikelihood function evaluated with only an intercept; see family.py for distribution-specific loglikelihoods

AIC [float] Akaike information criterion

D2 [float] percentage of explained deviance

adj_D2 [float] adjusted percentage of explained deviance

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

adj_pseudoR2 [float] adjusted McFadden's pseudo R2

SRMSE [float] standardized root mean square error

SSI [float] Sorensen similarity index

results [object] full results from estimated model. May contain addtional diagnostics

Example

———-

>>> import numpy as np

>>> import libpysal

>>> from spint.gravity import BaseGravity

>>> db = libpysal.io.open(libpysal.examples.get_path('nyc_bikes_ct.csv'))

>>> cost = np.array(db.by_col('tripduration')).reshape((-1,1))

>>> flows = np.array(db.by_col('count')).reshape((-1,1))

>>> model = BaseGravity(flows, cost)

>>> model.params

array([17.84839637, -1.68325787])

Methods

`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.

SRMSE	
SSI	
reshape	

__init__(*self*, *flows*, *cost*, *cost_func='pow'*, *o_vars=None*, *d_vars=None*, *origins=None*, *destinations=None*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)

Initialize self. See help(type(self)) for accurate signature.

Methods

`SRMSE`(self)	
`SSI`(self)	
__init__(self, flows, cost[, cost_func, ...])	Initialize self.
`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
`reshape`(self, array)	

3.1.2 spint.gravity.Gravity

class spint.gravity.**Gravity**(*flows*, *o_vars*, *d_vars*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)

Unconstrained (traditional gravity) gravity-type spatial interaction model

Parameters

flows [array of integers] n x 1; observed flows between O origins and D destinations

cost [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cost_func [string or function that has scalar input and output] functional form of the cost function; 'exp' | 'pow' | custom function

o_vars [array (optional)] n x p; p attributes for each origin of n flows; default is None

d_vars [array (optional)] n x p; p attributes for each destination of n flows; default is None

constant [boolean] True to include intercept in model; True by default

framework [string] estimation technique; currently only 'GLM' is avaialble

Quasi [boolean] True to estimate QuasiPoisson model; should result in same parameters as Poisson but with altered covariance; default to true which estimates Poisson model

SF [array] n x 1; eigenvector spatial filter to include in the model; default to None which does not include a filter; not yet implemented

CD [array] n x 1; competing destination term that accounts for the likelihood that alternative destinations are considered along with each destination under consideration for every OD pair; defaults to None which does not include a CD term; not yet implemented

Lag [W object] spatial weight for n observations (OD pairs) used to construct a spatial autoregressive model and estimator; defaults to None which does not include an autoregressive term; not yet implemented

Attributes

f [array] n x 1; observed flows; dependent variable; y

n [integer] number of observations

k [integer] number of parameters

c [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cf [function] cost function; used to transform cost variable

ov [array] n x p(o); p attributes for each origin of n flows

dv [array] n x p(d); p attributes for each destination of n flows

constant [boolean] True to include intercept in model; True by default

y [array] n x 1; dependent variable used in estimation including any transformations

X [array] n x k, design matrix used in estimation

params [array] n x k, k estimated beta coefficients; k = p(o) + p(d) + 1

yhat [array] n x 1, predicted value of y (i.e., fittedvalues)

cov_params [array] Variance covariance matrix (kxk) of betas

std_err [array] k x 1, standard errors of betas

pvalues [array] k x 1, two-tailed pvalues of parameters

tvalues [array] k x 1, the tvalues of the standard errors

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

resid_dev [array] n x 1, residual deviance of model

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

llnull [float] value of the loglikelihood function evaluated with only an intercept; see family.py for distribution-specific loglikelihoods

AIC [float] Akaike information criterion

D2 [float] percentage of explained deviance

adj_D2 [float] adjusted percentage of explained deviance

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

adj_pseudoR2 [float] adjusted McFadden's pseudo R2

SRMSE [float] standardized root mean square error

SSI [float] Sorensen similarity index

results [object] Full results from estimated model. May contain addtional diagnostics

Example

———-

>>> import numpy as np

>>> import libpysal

>>> from spint.gravity import Gravity

>>> db = libpysal.io.open(libpysal.examples.get_path('nyc_bikes_ct.csv'))

>>> cost = np.array(db.by_col('tripduration')).reshape((-1,1))

>>> flows = np.array(db.by_col('count')).reshape((-1,1))

>>> o_cap = np.array(db.by_col('o_cap')).reshape((-1,1))

>>> d_cap = np.array(db.by_col('d_cap')).reshape((-1,1))

>>> model = Gravity(flows, o_cap, d_cap, cost, 'exp')

>>> model.params

array([3.80050153e+00, 5.54103854e-01, 3.94282921e-01, -2.27091686e-03])

Methods

fit(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
local(self, loc_index, locs)	Calibrate local models for subsets of data from a single location to all other locations

SRMSE	
SSI	
reshape	

__init__(*self*, *flows*, *o_vars*, *d_vars*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)
Initialize self. See help(type(self)) for accurate signature.

Methods

SRMSE(self)	
SSI(self)	
__init__(self, flows, o_vars, d_vars, cost, ...)	Initialize self.
fit(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
local(self, loc_index, locs)	Calibrate local models for subsets of data from a single location to all other locations
reshape(self, array)	

3.1.3 spint.gravity.Production

class spint.gravity.**Production**(*flows*, *origins*, *d_vars*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)
Production-constrained (origin-constrained) gravity-type spatial interaction model

Parameters

flows [array of integers] n x 1; observed flows between O origins and D destinations

origins [array of strings] n x 1; unique identifiers of origins of n flows; when there are many origins it will be faster to use integers rather than strings for id labels.

cost [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cost_func [string or function that has scalar input and output] functional form of the cost function; 'exp' | 'pow' | custom function

d_vars [array (optional)] n x p; p attributes for each destination of n flows; default is None

constant [boolean] True to include intercept in model; True by default

framework [string] estimation technique; currently only 'GLM' is avaialble

Quasi [boolean] True to estimate QuasiPoisson model; should result in same parameters as Poisson but with altered covariance; default to true which estimates Poisson model

SF [array] n x 1; eigenvector spatial filter to include in the model; default to None which does not include a filter; not yet implemented

CD [array] n x 1; competing destination term that accounts for the likelihood that alternative destinations are considered along with each destination under consideration for every OD pair; defaults to None which does not include a CD term; not yet implemented

Lag [W object] spatial weight for n observations (OD pairs) used to construct a spatial autoregressive model and estimator; defaults to None which does not include an autoregressive term; not yet implemented

Attributes

f [array] n x 1; observed flows; dependent variable; y

n [integer] number of observations

k [integer] number of parameters

c [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cf [function] cost function; used to transform cost variable

o [array] n x 1; index of origin id's

dv [array] n x p; p attributes for each destination of n flows

constant [boolean] True to include intercept in model; True by default

y [array] n x 1; dependent variable used in estimation including any transformations

X [array] n x k, design matrix used in estimation

params [array] n x k, k estimated beta coefficients; k = # of origins + p + 1

yhat [array] n x 1, predicted value of y (i.e., fittedvalues)

cov_params [array] Variance covariance matrix (kxk) of betas

std_err [array] k x 1, standard errors of betas

pvalues [array] k x 1, two-tailed pvalues of parameters

tvalues [array] k x 1, the tvalues of the standard errors

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

resid_dev [array] n x 1, residual deviance of model

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

llnull [float] value of the loglikelihood function evaluated with only an intercept; see family.py for distribution-specific loglikelihoods

AIC [float] Akaike information criterion

D2 [float] percentage of explained deviance

adj_D2 [float] adjusted percentage of explained deviance

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

adj_pseudoR2 [float] adjusted McFadden's pseudo R2

SRMSE [float] standardized root mean square error

SSI [float] Sorensen similarity index

results [object] Full results from estimated model. May contain addtional diagnostics

Example

——-

>>> import numpy as np

>>> import libpysal

>>> from spint.gravity import Production

>>> db = libpysal.io.open(libpysal.examples.get_path('nyc_bikes_ct.csv'))

>>> cost = np.array(db.by_col('tripduration')).reshape((-1,1))

>>> flows = np.array(db.by_col('count')).reshape((-1,1))

>>> o = np.array(db.by_col('o_tract')).reshape((-1,1))

>>> d_cap = np.array(db.by_col('d_cap')).reshape((-1,1))

>>> model = Production(flows, o, d_cap, cost, 'exp')

>>> model.params[-4:]

array([1.34721352, 0.96357345, 0.85535775, -0.00227444])

Methods

`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
`local`(self[, locs])	Calibrate local models for subsets of data from a single location to all other locations

SRMSE	
SSI	
reshape	

__init__(*self*, *flows*, *origins*, *d_vars*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)
Initialize self. See help(type(self)) for accurate signature.

Methods

`SRMSE`(self)	
`SSI`(self)	
`__init__`(self, flows, origins, d_vars, cost, ...)	Initialize self.
`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
`local`(self[, locs])	Calibrate local models for subsets of data from a single location to all other locations
`reshape`(self, array)	

3.1.4 spint.gravity.Attraction

class spint.gravity.**Attraction**(*flows*, *destinations*, *o_vars*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)

Attraction-constrained (destination-constrained) gravity-type spatial interaction model

Parameters

flows [array of integers] n x 1; observed flows between O origins and D destinations

destinations [array of strings] n x 1; unique identifiers of destinations of n flows; when there are many destinations it will be faster to use integers over strings for id labels.

cost [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cost_func [string or function that has scalar input and output] functional form of the cost function; 'exp' | 'pow' | custom function

o_vars [array (optional)] n x p; p attributes for each origin of n flows; default is None

constant [boolean] True to include intercept in model; True by default

y [array] n x 1; dependent variable used in estimation including any transformations

X [array] n x k, design matrix used in estimation

framework [string] estimation technique; currently only 'GLM' is avaialble

Quasi [boolean] True to estimate QuasiPoisson model; should result in same parameters as Poisson but with altered covariance; default to true which estimates Poisson model

SF [array] n x 1; eigenvector spatial filter to include in the model; default to None which does not include a filter; not yet implemented

CD [array] n x 1; competing destination term that accounts for the likelihood that alternative destinations are considered along with each destination under consideration for every OD pair; defaults to None which does not include a CD term; not yet implemented

Lag [W object] spatial weight for n observations (OD pairs) used to construct a spatial autoregressive model and estimator; defaults to None which does not include an autoregressive term; not yet implemented

Attributes

f [array] n x 1; observed flows; dependent variable; y

n [integer] number of observations

k [integer] number of parameters

c [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cf [function] cost function; used to transform cost variable

d [array] n x 1; index of destination id's

ov [array] n x p; p attributes for each origin of n flows

constant [boolean] True to include intercept in model; True by default

params [array] n x k, k estimated beta coefficients; k = # of destinations + p + 1

yhat [array] n x 1, predicted value of y (i.e., fittedvalues)

cov_params [array] Variance covariance matrix (kxk) of betas

std_err [array] k x 1, standard errors of betas

pvalues [array] k x 1, two-tailed pvalues of parameters

tvalues [array] k x 1, the tvalues of the standard errors

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

resid_dev [array] n x 1, residual deviance of model

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

llnull [float] value of the loglikelihood function evaluated with only an intercept; see family.py for distribution-specific loglikelihoods

AIC [float] Akaike information criterion

D2 [float] percentage of explained deviance

adj_D2 [float] adjusted percentage of explained deviance

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

adj_pseudoR2 [float] adjusted McFadden's pseudo R2

SRMSE [float] standardized root mean square error

SSI [float] Sorensen similarity index

results [object] Full results from estimated model. May contain addtional diagnostics

Example

———-

```
>>> import numpy as np
>>> import libpysal
>>> from spint.gravity import Attraction
>>> db = libpysal.io.open(libpysal.examples.get_path('nyc_bikes_ct.csv'))
>>> cost = np.array(db.by_col('tripduration')).reshape((-1,1))
>>> flows = np.array(db.by_col('count')).reshape((-1,1))
>>> d = np.array(db.by_col('d_tract')).reshape((-1,1))
>>> o_cap = np.array(db.by_col('o_cap')).reshape((-1,1))
>>> model = Attraction(flows, d, o_cap, cost, 'exp')
>>> model.params[-4:]
array([ 1.21962276, 0.87634028, 0.88290909, -0.00229081])
```

Methods

`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
`local`(self[, locs])	Calibrate local models for subsets of data from a single location to all other locations

SRMSE	
SSI	
reshape	

__init__(*self*, *flows*, *destinations*, *o_vars*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)
Initialize self. See help(type(self)) for accurate signature.

Methods

`SRMSE`(self)	
`SSI`(self)	
__init__(self, flows, destinations, o_vars, ...)	Initialize self.
`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
`local`(self[, locs])	Calibrate local models for subsets of data from a single location to all other locations
`reshape`(self, array)	

3.1.5 spint.gravity.Doubly

class spint.gravity.**Doubly**(*flows*, *origins*, *destinations*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)
Doubly-constrained gravity-type spatial interaction model

Parameters

flows [array of integers] n x 1; observed flows between O origins and D destinations

origins [array of strings] n x 1; unique identifiers of origins of n flows; when there are many origins it will be faster to use integers rather than strings for id labels.

destinations [array of strings] n x 1; unique identifiers of destinations of n flows; when there are many destinations it will be faster to use integers rather than strings for id labels

cost [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cost_func [string or function that has scalar input and output] functional form of the cost function; 'exp' | 'pow' | custom function

constant [boolean] True to include intercept in model; True by default

y [array] n x 1; dependent variable used in estimation including any transformations

X [array] n x k, design matrix used in estimation

framework [string] estimation technique; currently only 'GLM' is avaialble

Quasi [boolean] True to estimate QuasiPoisson model; should result in same parameters as Poisson but with altered covariance; default to true which estimates Poisson model

SF [array] n x 1; eigenvector spatial filter to include in the model; default to None which does not include a filter; not yet implemented

CD [array] n x 1; competing destination term that accounts for the likelihood that alternative destinations are considered along with each destination under consideration for every OD pair; defaults to None which does not include a CD term; not yet implemented

Lag [W object] spatial weight for n observations (OD pairs) used to construct a spatial autoregressive model and estimator; defaults to None which does not include an autoregressive term; not yet implemented

Attributes

f [array] n x 1; observed flows; dependent variable; y

n [integer] number of observations

k [integer] number of parameters

c [array] n x 1; cost to overcome separation between each origin and destination associated with a flow; typically distance or time

cf [function] cost function; used to transform cost variable

o [array] n x 1; index of origin id's

d [array] n x 1; index of destination id's

constant [boolean] True to include intercept in model; True by default

params [array] n x k, estimated beta coefficients; k = # of origins + # of destinations; the first x-1 values pertain to the x destinations (leaving out the first destination to avoid perfect collinearity; no fixed effect), the next x values pertain to the x origins, and the final value is the distance decay coefficient

yhat [array] n x 1, predicted value of y (i.e., fittedvalues)

cov_params [array] Variance covariance matrix (kxk) of betas

std_err [array] k x 1, standard errors of betas

pvalues [array] k x 1, two-tailed pvalues of parameters

tvalues [array] k x 1, the tvalues of the standard errors

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

resid_dev [array] n x 1, residual deviance of model

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

llnull [float] value of the loglikelihood function evaluated with only an intercept; see family.py for distribution-specific loglikelihoods

AIC [float] Akaike information criterion

D2 [float] percentage of explained deviance

adj_D2 [float] adjusted percentage of explained deviance

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

adj_pseudoR2 [float] adjusted McFadden's pseudo R2

SRMSE [float] standardized root mean square error

SSI [float] Sorensen similarity index

results [object] Full results from estimated model. May contain addtional diagnostics

Example

———-

>>> import numpy as np

>>> import libpysal

>>> from spint.gravity import Doubly

>>> db = libpysal.io.open(libpysal.examples.get_path('nyc_bikes_ct.csv'))

>>> cost = np.array(db.by_col('tripduration')).reshape((-1,1))

>>> flows = np.array(db.by_col('count')).reshape((-1,1))

>>> d = np.array(db.by_col('d_tract')).reshape((-1,1))

>>> o = np.array(db.by_col('o_tract')).reshape((-1,1))

>>> model = Doubly(flows, o, d, cost, 'exp')

>>> model.params[-1:]

array([-0.00232112])

Methods

`fit`(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
`local`(self[, locs])	**Not inmplemented for doubly-constrained models** Not possible due to insufficient degrees of freedom.

SRMSE	
SSI	
reshape	

__init__(*self*, *flows*, *origins*, *destinations*, *cost*, *cost_func*, *constant=True*, *framework='GLM'*, *SF=None*, *CD=None*, *Lag=None*, *Quasi=False*)
Initialize self. See help(type(self)) for accurate signature.

Methods

SRMSE(self)	
SSI(self)	
__init__(self, flows, origins, destinations, ...)	Initialize self.
fit(self[, framework, Quasi])	Method that fits a particular count model usign the appropriate estimation technique.
local(self[, locs])	**Not inmplemented for doubly-constrained models** Not possible due to insufficient degrees of freedom.
reshape(self, array)	

3.2 Tests for overdispersion

spint.dispersion.alpha_disp(model[, alt_var])	Test the hypothesis that var[y] = mu (equidispersion) against the alternative hypothesis that var[y] = mu + alpha * alt_var(mu) where mu is the expected value of y, alpha is an estimated coefficient, and alt_var() specifies an alternative variance as a function of mu.
spint.dispersion.phi_disp(model)	Test the hypothesis that var[y] = mu (equidispersion) against the alternative hypothesis (quasi-Poisson) that var[y] = phi * mu where mu is the expected value of y and phi is an estimated overdispersion coefficient which is equivalent to 1+alpha in the alternative alpha dispersion test.

3.2.1 spint.dispersion.alpha_disp

spint.dispersion.**alpha_disp**(*model*, *alt_var=<function <lambda> at 0x7f46fa7e2620>*)

Test the hypothesis that var[y] = mu (equidispersion) against the alternative hypothesis that var[y] = mu + alpha * alt_var(mu) where mu is the expected value of y, alpha is an estimated coefficient, and alt_var() specifies an alternative variance as a function of mu. alt_var=lambda x:x corresponds to an alternative hypothesis of a negative binomimal model with a linear variance function and alt_var=lambda x:x**2 correspinds to an alternative hypothesis of a negative binomial model with a quadratic varaince function.

alpha > 0: overdispersion alpha = 1: equidispersion alpha < 0: underdispersion

Parameters

model [Model results class] function can only be called on a sucessfully fitted model which has a valid response variable, y, and a valid predicted response variable, yhat.

alt_var [function] specifies an alternative varaince as a function of mu. Function must take a single scalar as input and return a single scalar as output

Returns

———-

array [[alpha coefficient, tvalue of alpha, pvalue of alpha]]

3.2.2 spint.dispersion.phi_disp

spint.dispersion.**phi_disp**(*model*)

Test the hypothesis that var[y] = mu (equidispersion) against the alternative hypothesis (quasi-Poisson) that var[y] = phi * mu where mu is the expected value of y and phi is an estimated overdispersion coefficient which is equivalent to 1+alpha in the alternative alpha dispersion test.

phi > 0: overdispersion phi = 1: equidispersion phi < 0: underdispersion

Parameters

model [Model results class] function can only be called on a sucessfully fitted model which has a valid response variable, y, and a valid predicted response variable, yhat.

alt_var [function] specifies an alternative varaince as a function of mu. Function must take a single scalar as input and return a single scalar as output

Returns

———-

array [[alpha coefficient, tvalue of alpha, pvalue of alpha]]

CHAPTER

FOUR

REFERENCES

INDEX

Symbols

__init__() (*spint.gravity.Attraction method*), 17
__init__() (*spint.gravity.BaseGravity method*), 9
__init__() (*spint.gravity.Doubly method*), 19
__init__() (*spint.gravity.Gravity method*), 12
__init__() (*spint.gravity.Production method*), 14

A

alpha_disp() (*in module spint.dispersion*), 20
Attraction (*class in spint.gravity*), 15

B

BaseGravity (*class in spint.gravity*), 7

D

Doubly (*class in spint.gravity*), 17

G

Gravity (*class in spint.gravity*), 10

P

phi_disp() (*in module spint.dispersion*), 21
Production (*class in spint.gravity*), 12