A primer for working with the ***Sp***atial ***Int***eraction modeling (SpInt) module in the python spatial analysis library (PySAL)

Taylor Oshan[*]

November 6, 2016

1 Introduction

Spatial interaction modeling involves the analysis of flows from an origin to a destination, either over physical space (i.e., migration) or through abstract space (i.e., telecommunication). Despite being a fundamental technique to many geographic disciplines, there is relatively little software available to carry out spatial interaction modeling and the analysis of flow data, especially in the realm of free and open source software. Therefore, the purpose of this primer is to provide an overview of the recently developed spatial interaction modeling (SpInt) module[1] of the python spatial analysis library (PySAL)[2]. First, the current framework of the module will be highlighted. Next, the main functionality of the module will be illustrated using migration flows with a dataset previously used for spatial interaction modeling tutorials in the R programming environment (Dennett, 2012). Finally, some future additions are discussed.

2 The SpInt framework

2.1 Modeling framework

The core purpose of the SpInt module is to provide the functionality to calibrate spatial interaction models. Since the "family" of spatial interaction models put forth by Wilson (Wilson, 1971) are perhaps the most popular, they were chosen as the starting point of the module. Consider the basic gravity model (Fotheringham and O'Kelly, 1989),

$$T_{ij} = k\frac{V_i^{\mu} W_j^{\alpha}}{d_{ij}^{\beta}} \quad (1)$$

where

- T_{ij} = an $n \times m$ matrix of flows between n origins (subscripted by i) to m destinations (subscripted by j)

[*]Email: toshan@asu.edu
Affiliation: School of Geographical Sciences and Urban Planning, Arizona StateUniversity

[1]This primer is based on the 1.12.1 release of PySAL. For up-to-date code examples and namespaces, it is recommended to consult the documentation pertaining to the latest version of PySAL

[2]http://pysal.github.io/

- V = an $n \times p$ and vector of p origin attributes describing the emissiveness of i
- W = an $m \times p$ vector of p destination attributes describing the attractiveness of j
- d = an $n \times m$ matrix of the costs to overcome the physical separation between i and j (usually distance or time)
- k = a scaling factor to be estimated to ensure the total observed and predicted flows are consistent
- μ = a $p \times 1$ vector of parameters representing the effect of p origin attributes on flows
- α = a $p \times 1$ vector of parameters representing the effect of p destination attributes on flows
- β = a parameter representing the effect of movement costs on flows.

When data for T, V, W, and d are available we can estimate the model parameters (also called calibration), which summarize the effect that each model component contributes towards explaining the system of known flows (T). In contrast, known parameters can be used to predict unknown flows when there are deviations in model components (V, W, and d) or the set of locations in the system are altered.

Using an entropy-maximizing framework, Wilson derives a more informative and flexible "family" of four spatial interaction models (Wilson, 1971). This framework seeks to assign flows between a set of origins and destinations by finding the most probable configuration of flows out of all possible configurations, without making any additional assumptions. By using a common optimization problem and including information about the total inflows and outflows at each location (also called constraints), the following "family" of models can be obtained,

$$Unconstrained$$
$$Tij = V_i^{\mu} W_j^{\alpha} f(d_{ij}) \tag{2}$$

$$Production - constrained$$
$$T_{ij} = A_i O_i W_j^{\alpha} f(d_{ij}) \tag{3}$$
$$A_i = \sum_j W_j^{\alpha} f(d_{ij}) \tag{3a}$$

$$Attraction - constrained$$
$$T_{ij} = B_j D_j V_i^{\mu} f(d_{ij}) \tag{4}$$
$$B_j = \sum_i V_i^{\mu} f(d_{ij}) \tag{4a}$$

$$Doubly - constrained$$
$$T_{ij} = A_i B_j O_i D_j f(d_{ij}) \tag{5}$$
$$A_i = \sum_j W_j^{\alpha} B_j D_j f(d_{ij}) \tag{5a}$$
$$B_j = \sum_i V_i^{\mu} A_i O_i f(d_{ij}) \tag{5b}$$

where

- O_i = an $n \times 1$ vector of the total number of flows emanating from origin i
- D_j = an $m \times 1$ vector of the total number of flows terminating at destination j
- A_i = an $n \times 1$ vector of the origin balancing factors that ensures the total out-flows are preserved in the predicted flows
- B_j = an $m \times 1$ vector of the destination balancing factors that ensures the total in-flows are preserved in the predicted flows
- $f(d_{ij})$ = a function of cost or distance, referred to as the distance-decay function. Most commonly, this is an exponential or power function given by,

$$Power \qquad f(d_{ij}) = d_{ij}^{\beta} \tag{6}$$

$$Exponential \qquad f(d_{ij}) = exp(\beta * d_{ij}) \tag{7}$$

where β is expected to take a negative value. Different distance-decay functions assume different responses to the increasing costs associated with moving to more distant locations. Of note is that the unconstrained model with a power function distance-decay is equivalent to the basic gravity model in equation (2), except that the scaling factor, k, is not included. In fact, there is no scaling factor in any of the members of the family of maximum entropy models because there is a total trip constraint implied in their derivation and subsequently their calibration (Fotheringham and O'Kelly, 1989). Another aside is that in the doubly-constrained maximum entropy model, the values for A_i and B_j are dependent upon each other and may need to be computed iteratively depending on calibration technique. It is also usually assumed that all locations are both origins and destinations (i.e., $n = m$) for doubly-constrained models.

Each member of the family of models provides a different system structure, which can be chosen depending on the available data or the specific research question at hand. The so-called unconstrained model does not conserve the total inflows or outflows during parameter estimation. The production-constrained and attraction-constrained models conserve either the number of total inflows or outflows, respectively, and are therefore useful for building models that allocate flows either to a set of origins or to a set of destinations. Finally, the doubly-constrained model conserves both the inflows and the outflows at each location during model calibration. The quantity of explanatory information provided by each model is given by the number of parameters it provides. As such, the unconstrained model provides the most information, followed by the two singly-constrained models, with the doubly-constrained model providing the least information. Conversely, the model's predictive power increases with higher quantities of built-in information (i.e. total in or out-flows) so that the doubly-constrained model usually provides the most accurate predictions, followed by the two singly-constrained models, and the unconstrained model supplying the weakest predictions (Fotheringham and O'Kelly, 1989).

2.2 Calibration framework

Spatial interaction models are often calibrated via linear programming, nonlinear optimization, or, increasingly more often, through linear regression. Given the flexibility and extendability of a regression framework it was chosen as the primary model calibration technique within the SpInt module. By taking the natural logarithm of both sides of a spatial interaction model, say the basic gravity model, is is possible to obtain the so-called log-linear or log-normal spatial interaction model,

$$\begin{aligned}&Power - function\\ &\ln T_{ij} = k + \mu \ln V_i + \alpha \ln W_j - \beta \ln d_{ij} + \epsilon \qquad (8a)\end{aligned}$$

$$\begin{aligned}&Exponential - function\\ &\ln T_{ij} = k + \mu \ln V_i + \alpha \ln W_j - \beta d_{ij} + \epsilon \qquad (8b)\end{aligned}$$

where ϵ is a a normally distributed error term with a mean of 0. The only difference between equations (8a) and (8b) is the functional distance-decay specification, which results from plugging either equation (6) for a power function or (7) for an exponential function into equation (2) before linearizing it. The only practical difference here is that the distance is logged in equation (8a) whereas in equation (8b) it is not. Constrained spatial interaction models can be achieved by including fixed effects for the origins (production-constrained), fixed effects for the destinations (attraction-constrained) or both (doubly-constrained). However, there are several limitations of the log-normal gravity model, which include,

1. flows are often counts of people or objects and should be modeled as discrete entities;
2. flows are often not normally distributed;
3. downward biased flow predictions due to producing estimates for the logarithm of flows instead of actual flows;
4. zero flows are problematic since the logarithm of zero is undefined.

Therefore, the Poisson log-linear regression specification for the family of spatial interaction models was proposed (Flowerdew and Aitkin, 1982; Flowerdew and Lovett, 1988). This specification assumes that the number of flows between i and j is drawn from a Poisson distribution with mean, $\lambda_{ij} = T_{ij}$, where λ_{ij} is assumed to be logarithmically linked to the linear combination of variables,

$$\ln \lambda_{ij} = k + \mu \ln V_i + \alpha \ln W_j - \beta \ln d_{ij}) \quad (9a)$$

and exponentiating both sides of the equation yields the unconstrained Poisson log-linear gravity model,

$$T_{ij} = \exp(k + \mu \ln V_i + \alpha \ln W_j - \beta \ln d_{ij}) \quad (9b)$$

where equations (9a) and (9b) refer to the unconstrained model with a power function distance-decay. As previously mentioned, using fixed effects for the balancing factors in equations (3-5), the constrained variants of the family of spatial interaction models can be specified as,

$$Production - constrained$$
$$T_{ij} = \exp(k + \mu_i + \alpha \ln W_j - \beta \ln d_{ij}) \qquad (10)$$

$$Attraction - constrained$$
$$T_{ij} = \exp(k + \mu \ln V_i + \alpha_j - \beta \ln d_{ij}) \qquad (11)$$

$$Doubly - constrained$$
$$T_{ij} = \exp(k + \mu_i + \alpha_j - \beta \ln d_{ij}) \qquad (12)$$

where μ_i are origin fixed effects and α_i are destination fixed effects that achieve the same results as including balancing factors (Tiefelsdorf and Boots, 1995). Notice that k is the estimated intercept and must be included in these log-linear models (equation 8-12) to ensure the total number of flows is conserved, despite not being included in the maximum entropy models where such conservation is implied. Similar to equation (8b), the exponential function distance-decay can be specified in equation (9b, 10-12) by omitting the logarithm associated with d_{ij}. Using Poisson regression is more representative of flows and satisfies limitations (1-2) and it also alleviates limitations (3-4) since we no longer need to take the logarithm of T_{ij}. Using fixed effects within Poisson regression to calibrate the doubly-constrained model also avoids the need for iterative computation of the balancing factors that exists in other calibration methods (Fotheringham and O'Kelly, 1989).

Calibration of Poisson regression can be carried out within a generalized linear modeling framework (GLM) using iteratively weighted least squares (IWSL), which converges to the maximum likelihood estimates for the parameter estimates (Nelder and Wedderburn, 1972). To maintain computational efficiency with increasingly larger spatial interaction datasets, SpInt is built upon a custom GLM/IWLS routine that leverages sparse data structures for the production-constrained, attraction-constrained, and doubly-constrained models. As the number of locations in these models increases, so the does the number of binary indicator variables needed to construct the fixed effects that enforce the constraints. Therefore, larger spatial interaction datasets become increasingly sparse and the utilization of sparse data structures takes advantage of this feature. As a metric, constrained models with $n = m = 3,000$ locations, which implies $n*m = n^2 = 9,000,000$ observed flows when each location is an origin and destination, can be calibrated within minutes on a standard macbook pro notebook.

2.3 Model fit statistics

In order to evaluate the fit of spatial interaction models, it has been recommended that a variety of statistics be used (Knudsen and Fotheringham, 1986), which is the approach taken in SpInt. For the log-normal regression specification, it is popular to utilize the coefficient of determination (R^2), though this statistic is not available within the GLM framework used by SpInt. In replacement of the R^2 statistic, the SpInt framework provides a pseudo R^2 based on the likelihood function (McFadden, 1974),

$$R^2_{pseudo} = 1 - \frac{\ln \hat{L}(M_{full})}{\ln \hat{L}(M_{Intercept})} \qquad (13)$$

where $\hat{L}$ is the likelihood of an estimated model, M_{full} is the model including all explanatory variables of interest, and $M_{Intercept}$ is the model with only an intercept (i.e., no covariates). Like the R^2 statistic, the pseudo version is at a maximum at a value of 1 with higher values denoting better model fit. To account for model complexity, there is also an adjusted version of this statistic,

$$R^2_{adj-pseudo} = 1 - \frac{\ln \hat{L}(M_{full}) - K}{\ln \hat{L}(M_{Intercept})} \quad (14)$$

where K is the number of regressors. If model fit does not sufficiently improve, then it is possible for this measure to decrease as variables are added, signaling that the additional variables do not contribute towards a better model fit. Henceforth, these pseudo R^2 statistics are referred to solely as R^2 and adjusted R^2. Another model fit statistic available in the SpInt module that also accounts for model complexity is the the Akaike information criterion (AIC),

$$AIC = -2 \ln \hat{L}(M_{full}) + 2K \quad (15)$$

where lower AIC values indicate a better model fit (Akaike, 1974). This statistic is grounded in information theory, whereby the AIC is an asymptotic estimate of the information that is lost by using the full model to represent a given theoretical process.

The R^2 and AIC are designed for model selection, which means they should not be used to compare between different spatial systems. One solution to this issue is the standardized root mean square error (SRMSE),

$$SRMSE = \frac{\frac{\sqrt{\sum_i \sum_j (T_{ij} - \hat{T}_{ij})^2}}{n*m}}{\frac{\sum_i \sum_j T_{ij}}{n*m}} \quad (16)$$

where the numerator is the root mean square error of the observed flows, T_{ij}, and the flows predicted by the model, $\hat{T}_{ij}$, and the denominator is the mean of the observed flows and is responsible for standardization of the statistic. Here, $n * m$ is the number of origin-destination pairs that constitute the system of flows. A SRMSE value of 0 indicates perfect model fit, while higher values indicate decreasing model fit; however, the upper limit of the statistic is not necessarily 1 and will depend on the distribution of the observed values (Knudsen and Fotheringham, 1986).

One final fit statistic, a modified Sorensen similarity index (SSI), is included within the SpInt module because it has become increasingly popular in some spatial interaction literature that deals with non-parametric models (Lenormand et al., 2012; Masucci et al., 2012; Yan et al., 2013). Using the same symbol definition from the SRMSE, the SSI is defined as,

$$SSI = \frac{1}{(n*m)} \sum_i \sum_j \frac{2min(T_{ij}, \hat{T}_{ij})}{T_{ij} + \hat{T}_{ij}} \quad (17)$$

which is bounded between values of 0 and 1 with values closer to 1 indicating a better model fit.

3 An illustrative example: migration in Austria

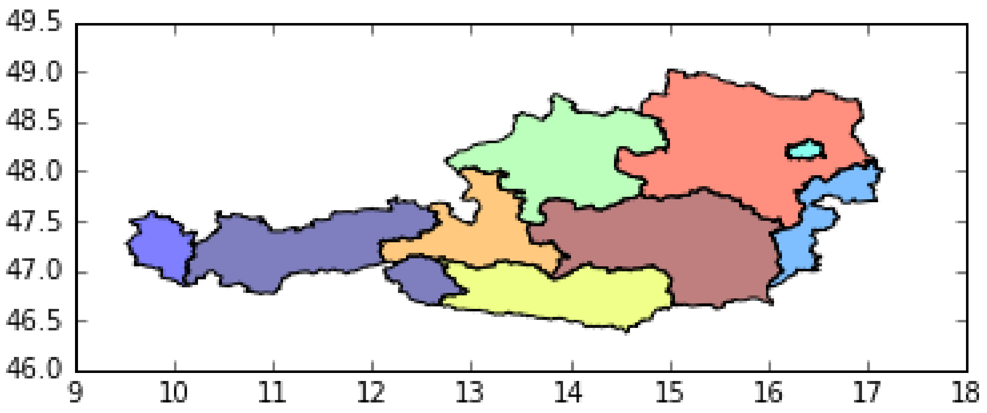
3.1 The data

Despite being a small toy dataset, the following example is utilized for consistency since it was previously used to demonstrate spatial interaction modeling in the R programming language (Dennett, 2012). The data are migration flows between Austrian NUTS level 2 municipalities in 2006. In order to use a regression-based calibration, the data has to be transformed from the matrices and vectors described in equations (1-5) to a table where each row represents a single origin-destination dyad, (i, j) and any variables associated with locations i and j. Details on how to do this are outlined further in (LeSage and Pace, 2008), though this has already been done in the example data. Let's have a look!

```
In [19]: import pandas as pd
         import geopandas as gp
         %pylab inline
         austria_shp = gp.read_file('austria.shp')
         austria_shp.plot()
         austria = pd.read_csv('austria.csv')
         austria.head()
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
Out[19]:    Unnamed: 0 Origin Destination  Data    Oi     Dj           Dij
         0           0   AT11        AT11     0  4016   5146  1.000000e-300
         1           1   AT11        AT12  1131  4016  25741   1.030018e+02
         2           2   AT11        AT13  1887  4016  26980   8.420467e+01
         3           3   AT11        AT21    69  4016   4117   2.208119e+02
         4           4   AT11        AT22   738  4016   8634   1.320075e+02
```



The **Origin** and **Destination** columns refer to the labels for origin locations, i, and the labels for destination locations, j, the **Data** column is the number of flows between i and j, the **Oi** and **Dj** columns are the number of total out-flows at i and total in-flows at j, respectively, and the **Dij** column is the Euclidian distance between the centroids of i and j. In this case we use the total out-flow and total in-flow as variables to describe how emissive an origin is and how attractive a destination is. If we want a more informative and interesting model we can replace these with application specific variables that pertain to different hypotheses. Next, lets format the data into arrays.

```
In [4]: austria = austria[austria['Origin'] != austria['Destination']]
        flows = austria['Data'].values
        Oi = austria['Oi'].values
        Dj = austria['Dj'].values
        Dij = austria['Dij'].values
        Origin = austria['Origin'].values
        Destination = austria['Destination'].values
```

The Oi and Dj vectors need not be $n^2 \times 1$ arrays. In fact, they can be $n^2 \times k$ where k is the number of variables that are being used to describe either origin or destination attributes associated with flows. It should also be noted that intra-zonal flows have been excluded (the first line of code above). This is sometimes done because intra-zonal flows are large compared to inter-zonal flows and would therefore heavily influence the model or because it is not possible to adequately define a distance associated with intra-zonal flows. Some solutions to these issues have been proposed (Kordi et al., 2012; Tsutsumi and Tamesue, 2011), though for simplicity, intra-zonal were removed for this example.

3.2 Calibrating the models

Now, lets load the main SpInt functionality and calibrate some models. The "family" of spatial interaction models are found within the **gravity** namespace of the SpInt module and the estimated parameters can be accessed via the **params** attribute of a successfully instantiated spatial interaction model.

```
In [5]: from pysal.contrib.spint.gravity import Gravity
        from pysal.contrib.spint.gravity import Production
        from pysal.contrib.spint.gravity import Attraction
        from pysal.contrib.spint.gravity import Doubly
```

Unconstrained (basic gravity) model

```
In [6]: gravity = Gravity(flows, Oi, Dj, Dij, 'exp')
        print gravity.params
```

```
[ -8.01822841e+00   8.69316127e-01   8.91445153e-01  -6.22938370e-03]
```

Production-constrained model

```
In [7]: production = Production(flows, Origin, Dj, Dij, 'exp')
        print production.params[-2:]
```

```
[ 0.90285448 -0.0072617 ]
```

Attraction-constrained model

```
In [8]: attraction = Attraction(flows, Destination, Oi, Dij, 'exp')
        print attraction.params[-2:]
```

```
[ 0.90037216 -0.00695034]
```

Doubly-constrained model

```
In [9]: doubly = Doubly(flows, Origin, Destination, Dij, 'exp')
        print doubly.params[-1:]

[-0.00791533]
```

Note that for the above examples the print statement for the constrained models params attribute is limited to print only the main model variables (i.e., not fixed effects), though it is still possible to access the fixed effect parameters too.

```
In [10]: print production.params

[-1.16851884  1.68980685  2.15135947  0.59917703  0.88336198  1.20669895
  0.68945769  1.15434225  1.01013674  0.90285448 -0.0072617 ]
```

The first parameter is always the overall intercept with the subsequent 8 parameters representing the fixed effects in this case. You might ask, "why not 9 fixed effects for the 9 different municipalities?". Due to the coding scheme used in SpInt, and many popular statistical programming languages, you would use $n - 1$ binary indicator variables in the design matrix to include the fixed effects for all 9 municipalities in the model. While the non-zero entries in these columns of the design matrix indicate which rows are associated with which municipality, where a row has all zero entries then implicitly refers to the nth municipality that has been left out. In Spint, this is always the first origin or destination for the production-constrained and attraction-constrained models. For the doubly-constrained model, both the first origin and the first destination are left out (Tiefelsdorf and Boots, 1995). In terms of interpreting the parameters, these dropped locations are assumed to be 0.

You can also access typical model diagnostics, such as standard errors (**std_err**), t-values (**tvalues**), p-values (**pvalues**), and confidence intervals (**cont_int**).

3.3 Interpreting the parameters

First, it will be demonstrated how to interpret the coefficients associated with the main model variables from a general Poisson regression. However, because the spatial interaction model is a log-linear Poisson regression (i.e., we take the log of the explanatory variables) the same interpretation often cannot be applied because we are working in logarithmic space. Therefore, it will also be demonstrated how to interpret the parameters when they are associated with a logged explanatory variable.

Recall from the previous section that the exponential distance-decay specification results in a model that does not take the logarithm of d_{ij}. Therefore, we can use an unconstrained gravity model with an exponential distance-decay specification to demonstrate a typical interpretation of coefficients from a Poisson regression.

```
In [11]: gravity = Gravity(flows, Oi, Dj, Dij, 'exp')
         print gravity.params

[ -8.01822841e+00   8.69316127e-01   8.91445153e-01  -6.22938370e-03]
```

-6.22938370e-03 is the coefficient for the distance variable in the above example. In Poisson regression, the coefficients are typically interpreted as the proportionate change in the predicted response, here T_{ij}, if we increase an explanatory variable by 1 unit (Cameron and Trivedi, 2013). Technically, this is expressed as,

$$\tilde{T}_{ij} = T_{ij} * \exp(\beta) \quad (18)$$

where $\tilde{T}_{ij}$ is the new value of T_{ij} and β is a coefficient, here the one typically associated with distance in a Poisson log-linear spatial interaction model with an exponential function distance-decay. For this example, this means from a 1 unit increase in distance, holding all other factors constant, if our model predicted 2,500 flows, then we can expect the number of flows to decrease to approximately 2,484.475. We can also identify the percent change expected from a one unit increase in distance using,

$$\Delta_{\%} = (1 - exp(\beta)) * 100.0 \quad (19)$$

which serves as an alternative interpretation of β. In this case, we could say that from a 1 unit increase in distance we could expect the number of predicted flows to decrease by approximately 0.621%.

However, neither equation (19) or (20) is applicable when the coefficient is associated with a logged explanatory variable. This is important for Poisson log-linear spatial interaction models because this applies to the origin and destination variables when using an exponential function of distance-decay and to the origin, destination, and distance variables when using a power function of distance-decay. In these cases, the interpretation of the coefficients becomes the percent change in the predicted response, here T_{ij}, if we increase the associated explanatory variable by 1% (Cameron and Trivedi, 2013). For example, 8.91445153e-01 is the coefficient associated with destination total in-flows (i.e., attractiveness) in the above example. Then if we increase the in-flows to location j by 1%, say from $25,000$ to $25,250$, and holding all other factors constant, we can expect the number of flows from i to j (i.e., T_{ij}) to increase from $2,000$ to $2,020$.[3]

Finally, the fixed effects in the constrained models can be interpreted such that the mean predicted flows, T_{ij}, are e^{μ_i} (e^{α_j}) times larger if they originate (terminate) from location i (location j) (Cameron and Trivedi, 2013), where e^{μ_i} is equivalent notation for $\exp(\mu_i)$.

3.4 Assessing model fit

We can compare the different model fit statistics across the four types of spatial interaction models for this example. Let's process the statistics into a tidy table and have a look.

```
In [12]: R2, adjR2, SSI, SRMSE, AIC = [], [], [], [], []
         model_name = ['grav', 'prod', 'att', 'doub']
         col_names = ['R2', 'adjR2', 'AIC', 'SRMSE', 'SSI']
         models = [gravity, production, attraction, doubly]
```

[3]This example is only illustrative. Of course, if we increased the total in-flows, this would imply that we are also increasing the total out-flows from somewhere else and therefore the system could not truly be held constant. However, substantive modeling calls for origin and destination variables that are not derived from the interaction matrix. Therefore, this is not an issue in practice when there is the assumption of independence between flows. It should also be noted that it has been hypothesized that flows may not be independent and therefore more accurate estimates can be obtained using more advanced methods (LeSage and Pace, 2008; Chun, 2008).

```
        for model in models:
            R2.append(model.pseudoR2)
            adjR2.append(model.adj_pseudoR2)
            SSI.append(model.SSI)
            SRMSE.append(model.SRMSE)
            AIC.append(model.AIC)

        cols = {'model_name': model_name,
                'R2': R2,
                'adjR2': adjR2,
                'SSI': SSI,
                'SRMSE': SRMSE,
                'AIC': AIC }

        data = pd.DataFrame(cols).set_index('model_name')
        data[col_names]
```

```
Out[12]:                  R2     adjR2           AIC     SRMSE       SSI
        model_name
        grav          0.885764  0.885718  20122.074349  0.607776  0.727358
        prod          0.910156  0.910031  15841.253799  0.464520  0.740914
        att           0.909355  0.909230  15982.313101  0.584048  0.752155
        doub          0.943540  0.943335   9977.159141  0.379286  0.811852
```

From this table we can see that all of the fit statistics indicate a better model fit as constraints are introduced. That is, the weakest model fit is consistently related to the gravity model, with similarly increased model fit for the production-constrained and attraction-constrained models, and finally, the best model fit is associated with the doubly constrained model. We can also see that the R^2 and adjusted R^2 are very close, since these models have a very similar number of explanatory variables, thereby resulting in little or no penalization for model complexity.

We can also take a look at whether the power or exponential distance-decay specification results in a better model fit. For simplicity, lets just take a look at the SRMSE for a doubly constrained model.

```
In [13]: print 'SRMSE for exp distance-decay: ', doubly.SRMSE
         pow_doubly = Doubly(flows, Origin, Destination, Dij, 'pow')
         print 'SRMSE for exp distance-decay: ', pow_doubly.SRMSE

SRMSE for exp distance-decay:  0.37928618533
SRMSE for exp distance-decay:  0.277703139642
```

For this example, it looks like the power distance-decay specification results in a better model fit.

3.5 Local models

The SpInt module also makes it possible to calibrate "local" models, which subset the data by specific origins or destinations in order o investigate how spatial interaction processes vary over space (Fotheringham and Brunsdon, 1999). Below is an example of how to get local parameters and local diagnostics for a gravity model subset by its origins. The result is a dictionary of lists where the keys are the different sets of local values including parameters, hypothesis testing diagnostics, and the previously reviewed fit statistics.

```
In [14]: gravity = Gravity(flows, Oi, Dj, Dij, 'pow')
         local_gravity = gravity.local(Origin, np.unique(Origin))
```

Lets take a look at the local distance-decay parameters. The origin, destination and distance-decay parameters are indexed sequentially through the design matrix starting with 0 as you move through the origin attributes, through to the destination attributes, and finally the distance-decay attribute. Therefore, for n variables, the distance-decay parameters are always the $n-1^{th}$ parameter, in this case of 3 variables: **param2**.

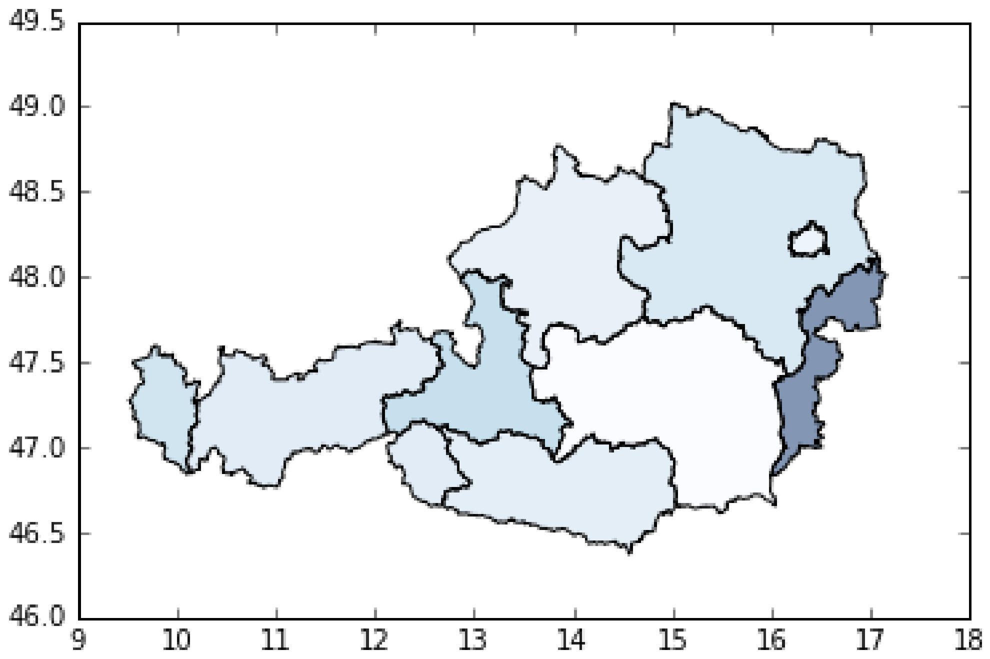
```
In [15]: print np.round(local_gravity['param2'], 4)

[-3.4028 -1.3583 -0.8307 -1.1492 -0.4781 -1.0095 -1.6758 -1.2156 -1.5397]
```

We can also take a look at the local R^2.

```
In [16]: print np.round(local_gravity['pseudoR2'], 4)

[ 0.9665  0.9894  0.9893  0.5205  0.676   0.7298  0.6333  0.432   0.515 ]
```

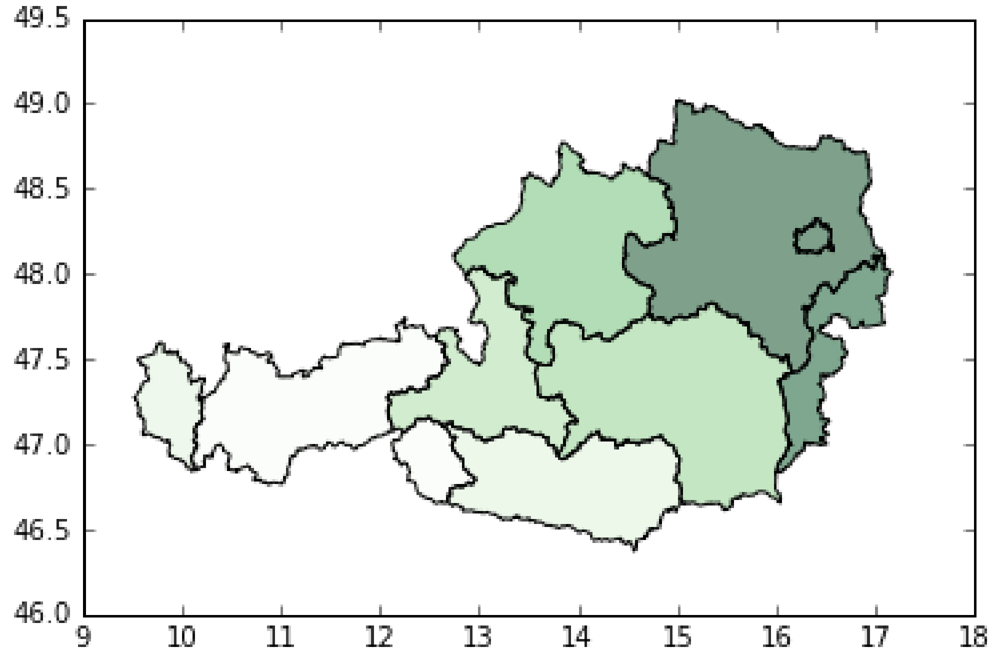
Both the local distance-decay and the R^2 show some variation. We can explore this spatially, by mapping the local values. First, lets join the local values to a shapefile and then plot the local distance-decay parameters

```
In [17]: #Join local values to census tracts
         local_vals = pd.DataFrame({'betas': local_gravity['param2'],
                                    'Dest':np.unique(Origin),
                                    'pseudoR2': local_gravity['pseudoR2']})
         local_vals = pd.merge(local_vals, austria_shp[['NUTS_ID', 'geometry']],
                               left_on='Dest', right_on='NUTS_ID')
         local_vals = gp.GeoDataFrame(local_vals)

         #Plot betas - use inverse so the most negative values are "higher"
         fig = plt.figure()
         ax = fig.add_subplot(111)
         local_vals['inv_betas'] = (local_vals['betas']*-1)
         local_vals.plot('inv_betas', cmap='Blues', ax=ax)

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x115da9fd0>
```



Next, lets map the local R^2 values. Above we can see a much stronger distance-decay for the most westerly municipality. Below we can see that the model fit is stronger in the north-west and decreases in the south-east. Using these patterns, we could then further postulate why they arise or how we might be able to improve model fit.

```
In [18]: fig = plt.figure()
         ax = fig.add_subplot(111)
         local_vals.plot('pseudoR2', cmap='Greens', ax=ax)

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x116f3b9d0>
```



Further functionality

In addition to all of the features presented here, there are several other tools that exist in SpInt or could be added. First, there are dispersion tests available in the **dispersion** namespace of the SpInt module, which can be used to test whether or not the Poisson equidispersion assumption is met. That is, that the conditional mean and variance are equivalent, which can be unrealistic in many scenarios. If these tests indicate overdispersion or underdispersion, then it might be appropriate to use a Quasi-Poisson model, which relaxes the equidispersion assumption of the Poisson model. The resulting parameter estimates are equivalent to the Poisson model, but the standard errors are typically larger whenever equidispersion does not hold (Wedderburn, 1974). The Quasi-Poisson model specification can be carried out by setting **Quasi=True** in any of the spatial interaction models introduced here. Alternatively, it might be more appropriate to change the underlying probability model from Poisson to that of negative binomial or a zero-inflated model. However, this has not yet been implemented in SpInt and therefore remains as future work.

Another area of potential expansion is to accommodate several paradigms for incorporating spatial effects into spatial interaction models, such as competing destinations (Fotheringham, 1983), a spatial lag autoregressive model (LeSage and Pace, 2008), or an eigenvector spatial filter model (Chun, 2008). These paradigms require code that computes additional variables, more complex calibration techniques, and specialized representations of spatial relationships. Some solutions to the latter are available in the **spintW** namespace of the **weights** module of PySAL. While there is still much work to be done to develop a more robust set of open source spatial interaction modeling tools, SpInt provides a starting point for which to build upon.

References

Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723.

Cameron, A. C. and Trivedi, P. K. (2013). *Regression Analysis of Count Data*. Cambridge University Press, New York.

Chun, Y. (2008). Modeling network autocorrelation within migration flows by eigenvector spatial filtering. *Journal of Geographical Systems*, 10(4):317–344.

Dennett, A. (2012). Estimating flows between geographical locations:'get me started in'spatial interaction modelling. Working Paper 184, Citeseer, UCL.

Flowerdew, R. and Aitkin, M. (1982). A Method of Fitting the Gravity Model Based on the Poisson Distribution. *Journal of Regional Science*, 22(2):191–202.

Flowerdew, R. and Lovett, A. (1988). Fitting Constrained Poisson Regression Models to Interurban Migration Flows. *Geographical Analysis*, 20(4):297–307.

Fotheringham, A. S. (1983). A new set of spatial-interaction models: the theory of competing destinations. *Environment and Planning A*, 15(1):15–36.

Fotheringham, A. S. and Brunsdon, C. (1999). Local Forms of Spatial Analysis. *Geographical Analysis*, 31(4):340–358.

Fotheringham, A. S. and O'Kelly, M. E. (1989). *Spatial Interaction Models:Formulations and Applications*. Kluwer Academic Publishers, London.

Knudsen, D. and Fotheringham, A. (1986). Matrix comparison, Goodness-of-fit, and spatial interaction modeling. *International Regional Science Review*, 10:127–147.

Kordi, M., Kaiser, C., and Fotheringham, A. S. (2012). A possible solution for the centroid-to-centroid and intra-zonal trip length problems. In *International Conference on Geographic Information Science, Avignon*.

Lenormand, M., Huet, S., Gargiulo, F., and Deffuant, G. (2012). A Universal Model of Commuting Networks. *PLoS ONE*, 7(10):e45985.

LeSage, J. P. and Pace, R. K. (2008). Spatial Econometric Modeling Of Origin-Destination Flows. *Journal of Regional Science*, 48(5):941–967.

Masucci, A. P., Serras, J., Johansson, A., and Batty, M. (2012). Gravity vs radiation model: on the importance of scale and heterogeneity in commuting flows. *arXiv:1206.5735 [physics]*. arXiv: 1206.5735.

McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, pages 105–142. Academic Press, New York.

Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384.

Tiefelsdorf, M. and Boots, B. (1995). The specification of constrained interaction models using the SPSS loglinear procedure. *Geographical Systems*, 2:21–38.

Tsutsumi, M. and Tamesue, K. (2011). Intraregional Flow Problem in Spatial Econometric Model for Origin-destination Flows. *Procedia - Social and Behavioral Sciences*, 21:184–192.

Wedderburn, R. W. M. (1974). Quasi-Likelihood Functions, Generalized Linear Models, and the Gauss-Newton Method. *Biometrika*, 61(3):439–447.

Wilson, A. G. (1971). A family of spatial interaction models, and associated developments. *Environment and Planning A*, 3:1–32.

Yan, X.-Y., Zhao, C., Fan, Y., Di, Z., and Wang, W.-X. (2013). Universal Predictability of Mobility Patterns in Cities. *arXiv:1307.7502 [physics]*. arXiv: 1307.7502.