# SpIntPrimer

October 15, 2016

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

### 1.1 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 geogaphic disciplines, there is relatively little software available to carry out spatial interaction modeling and the analysis of flow data. Therefore, the purpose of this primer is to provide an overview of the recently develped spatial interaction modeling (SpInt) module of the python spatial analysis library (PySAL). First, the current framework of the module will be highlighted. Next, the main functionality of the module will be illustrated with a common regional science example, migration flows, with a dataset previously used for spatial interaction modeling tutorials in the R programming environment. Finally, some auxiliary functionality and future additions are discussed.

### 1.2 The SpInt Framework

#### 1.2.1 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 & 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 $j$)
- $V$ = 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 flows is consistent
- $\mu$ = a $p \times 1$ vector of parameters representing the effect of $p$ origin attributes on flows
- $\alpha$ = a $p \times 1$ vector of parameters representing the effect of $p$ destination attributes on flows
- $\beta$ = an exponential 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:

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

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

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

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

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 thevorigin 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 an exponential or power function given by,

$$\begin{aligned} &Power \\ &f(d_{ij}) = d_{ij}^{\beta} \end{aligned} \quad (6)$$

$$\begin{aligned} &Exponential \\ &f(d_{ij}) = exp(\beta * d_{ij}) \end{aligned} \quad (7)$$

where $\beta$ is expected to take a negative value. Different distance-decay functions assume different responses to higher costs associated with moving to more distant locations. Of note is that the unconstrained model with of 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 constrained implied in their derivation and subsequently incorporated into their calibration (Fotheringham & O'Kelly , 1989). Another aside is that in the doubly-constrained maximum entoropy model the values for $A_i$ and $B_j$ are dependent upon each other and may need to be computed iteratively depending on calibration technqiue. It is also usually assumed that $n = m$ for doubly-constrained models.

The family provides different model strucutre depedning on the available data or the 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 & O'Kelly, 1989).

### 1.2.2 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 technqiue 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:

$$\ln T_{ij} = k + \mu \ln V_i + \alpha \ln W_j - \beta \ln d_{ij} + \epsilon \quad (8)$$

where $\epsilon$ is a a normally distributed error term with a mean of 0. Constrained spatial interaction models can be achieved by including a fixed effects for the origins (production-constrained), a fixed effect 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. Bias
4. zero flows

Therefore, Flowerdew (1984) proposes the Poisson log-linear regression specification for the family of spatial interaction models. 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 $\lambda_{ij}$ is assumed to be logarithmically linked to the linear combination of variables,

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

and exponentiating both sides of the equation yields the following family of Poisson log-linear spatial interaction models:

$$Unconstrained$$
$$T_{ij} = \exp(k + \mu V_i + \alpha W_j - \beta d_{ij}) \qquad (10)$$

$$Production - constrained$$
$$T_{ij} = \exp(k + \mu_i + \alpha W_j - \beta d_{ij}) \qquad (11)$$

$$Attraction - constrained$$
$$T_{ij} = \exp(k + \mu V_i + \alpha_j - \beta d_{ij}) \qquad (12)$$

$$Doubly - constrained$$
$$T_{ij} = \exp(k + \mu_i + \alpha_j - \beta d_{ij}) \qquad (13)$$

where $\mu_i$ are origin fixed effects and $\alpha_i$ are destination fixed effects that achieve the same constraints as the balancing factors in equations (2-5). Using Poisson regression automatically alleviates limiations (1) and (2) and since we no longer need to take the logarithm of $T_{ij}$, issue (4) is also absolved. 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.

Calibration of Poisson regression can be carried out within a generalized linear modeling framework (GLM) using iteratively weighted least sqaures (IWSL), which converges to the maximum likelihood estimates for the parameter estimates (Nelder & 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 take advantage of this feature. As a metric, constrained models with $n = 3,000$ locations, which implies $n^2 = 9,000,000$ observations can still be calibrated within minutes on a standard macbook pro notebook.

## 1.3 An Illustrative example: migration in Austria

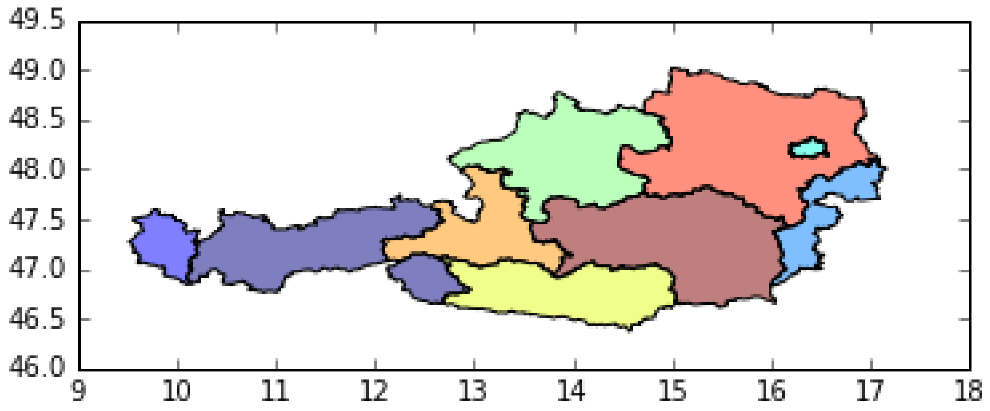
### 1.3.1 The data

The following example purposefully utilizes data that has was previously used to demonstrate spatial interaction modeling the R programming language for validation and consistency. The data is 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 either location. Details on how to do this are outlined further in (LeSage & Pace, 2008), though this has already been done in the example data. Let's have a look!

```
In [74]: import pandas as pd
         import geopandas as gp
         %pylab inline
         austria_shp = gp.read_file('austria.shp')
         austria_shp.plot()
         austria = pd.read_csv('http://dl.dropbox.com/u/8649795/AT_Austria.csv')
         austria.drop(['Oi2007', 'Dj2007', 'OrigAT11', 'OrigAT12', 'OrigAT13', 'Ori
         austria.head()
```

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

```
Out[74]:   Origin Destination  Data    Oi     Dj           Dij
         0   AT11        AT11     0  4016   5146  1.000000e-300
         1   AT11        AT12  1131  4016  25741   1.030018e+02
         2   AT11        AT13  1887  4016  26980   8.420467e+01
         3   AT11        AT21    69  4016   4117   2.208119e+02
         4   AT11        AT22   738  4016   8634   1.320075e+02
```



The **Origin** and **Destination** columns refer to the origin, $i$, and destination, $j$, location labels, the **Data** column is the number of flows, the **Oi** and **Dj** columns are the number of total out-flows and total in-flows, respectively, and the **Dij** column is the distance between $i$ 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 model we can replace these with application specific variables that pertain to different hypotheses. Next, lets format the data into arrays.

```
In [42]: 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 $n^2 \times 1$ vectors. In fact, they can be $n^2 \times k$ where $k$ is the number of variables that are being used to describe origin or desitnation factors associated with flows.

### 1.3.2 Calibrating the models

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

```
In [64]: from pysal.contrib.spint.gravity import Gravity, Production, Attraction, D
```

Unconstrained (basic gravity) model

```
In [65]: gravity = Gravity(flows, Oi, Dj, Dij, 'exp')
         print gravity.params

[ 0.44314667  0.51739961 -0.00979932]
```

Production-constrained model

```
In [66]: production = Production(flows, Origin, Dj, Dij, 'exp')
         print production.params[-2:]

[ 0.90285448 -0.0072617 ]
```

Attraction-constrained model

```
In [54]: attraction = Attraction(flows, Destination, Oi, Dij, 'exp')
         print attraction.params[-2:]

[ 0.90037216 -0.00695034]
```

Doubly-constrained model

```
In [55]: doubly = Doubly(flows, Origin, Destination, Dij, 'exp')
         print doubly.params[-1:]
```

```
[-0.00791533]
```

Note that for the constrained models we have limited the params attribute to print only those associated variables (i.e., not fixed effects), though it is still possible to access the fixed effect parameters too.

```
In [67]: print production.params
```

```
[-1.16851884  0.52128801  0.98284063 -0.56934181 -0.28515686  0.0381801
 -0.47906115 -0.0141766  -0.1583821   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 miunicipality, where a row has all zero entries then 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-cosntrained model, both the first origin and the first destination are left out. In terms of interpetting the parameters, these dropped locations are assumed to be 0. Since the fixed effects parameters are interpretted as deviations from the overall intercept, this essentially means the intercept acts as the fixed effect for the first location. Therefore, we could also say first 9 parameters are the origin fixed effect parameters and that the last two parameters are for destination attractiveness and distance, respectively.

### 1.3.3 Interpretting the parameters

### 1.3.4 Assessing model fit

```
In [63]: print gravity.pseudoR2
         print production.pseudoR2
         print attraction.pseudoR2
         print doubly.pseudoR2
         print ''
         print gravity.D2
         print production.D2
         print attraction.D2
         print doubly.D2
         print ""
         print gravity.SSI
         print production.SSI
```

```
        print attraction.SSI
        print doubly.SSI
        print ""
        print gravity.SRMSE
        print production.SRMSE
        print attraction.SRMSE
        print doubly.SRMSE
```

```
0.812858632041
0.910155702595
0.909354566583
0.943539912198

0.834893355219
0.913167247331
0.912363460094
0.946661920896

0.68757357636
0.740913740348
0.752155260541
0.811852110904

1.01757072219
0.464519773826
0.58404798182
0.37928618533
```

#### 1.3.5 Local models

#### 1.3.6 Testing for overdispersion

### 1.4 Additional functionality

#### 1.4.1 Existing features

#### 1.4.2 Future additions

Finally, there are several paradigms for incorporating spatial effects into spatial interaction models (competing destinations, spatial autoregressive, eigenvector spatial filter)

```
In [ ]:
```