

# Simulated\_data\_example\_Poisson-MGWR

May 18, 2020

## Notebook Outline:

- Section 0.0.1
- Section 0.0.2
  - Section 0.0.2
  - Section 0.0.2
- Section 0.0.3
  - Section 0.0.3
  - Section 0.0.4
- Section 0.0.4
  - Section 0.0.4
  - Section 0.0.4
- Section 0.0.5

Branch - gsco19

PR - <https://github.com/pysal/mgwr/pull/60>

## 0.0.1 Set up Cells

```
In [1]: import sys
        #change path here to point to your folder
        sys.path.append("C:/Users/msachde1/Downloads/Research/Development/mgwr")

In [2]: import warnings
        warnings.filterwarnings("ignore")
        import pandas as pd
        import numpy as np

        from mgwr.gwr import GWR
        from spglm.family import Gaussian, Binomial, Poisson
        from mgwr.gwr import MGWR
        from mgwr.sel_bw import Sel_BW
        import multiprocessing as mp
        pool = mp.Pool()
```

```

from scipy import linalg
import numpy.linalg as la
from scipy import sparse as sp
from scipy.sparse import linalg as spla
from spreg.utils import spdots, spmultiply
from scipy import special
import libpysal as ps
import seaborn as sns
import matplotlib.pyplot as plt
from copy import deepcopy
import copy
from collections import namedtuple
import spglm

```

## 0.0.2 Create Simulated Dataset

### Forming independent variables

```

In [3]: def add(a,b):
        return 1+((1/120)*(a+b))

        def con(u,v):
            return (0*(u)*(v))+0.3

        def sp(u,v):
            return 1+1/3240*(36-(6-u/2)**2)*(36-(6-v/2)**2)

In [4]: x = np.linspace(0, 25, 25)
        y = np.linspace(25, 0, 25)
        X, Y = np.meshgrid(x, y)

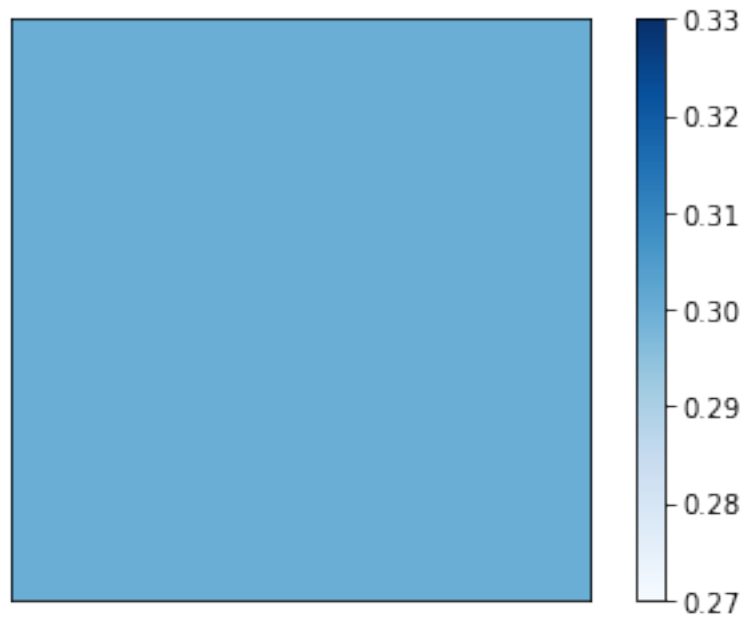
In [5]: x1=np.random.normal(0,1,625)
        x2=np.random.normal(0,1,625)
        error = np.random.normal(0,0.1,625)

In [6]: B0=con(X,Y)
        B1=add(X,Y)
        B2=sp(X,Y)

In [7]: plt.imshow(B0, extent=[0,10, 0, 10], origin='lower',cmap='Blues')
        plt.colorbar()
        plt.axis(aspect='image')
        plt.xticks([])
        plt.yticks([])

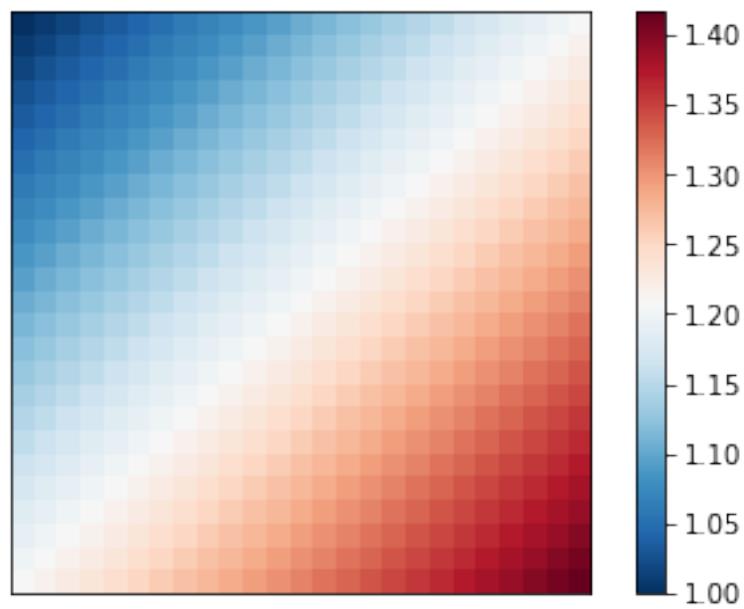
Out[7]: ([], <a list of 0 Text yticklabel objects>)

```



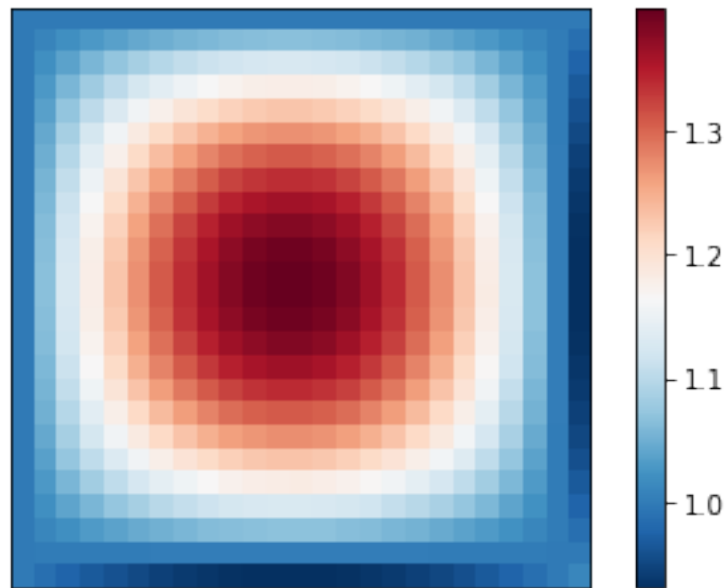
```
In [8]: plt.imshow(B1, extent=[0,10, 0, 10], origin='lower',cmap='RdBu_r')  
        plt.colorbar()  
        plt.axis(aspect='image')  
        plt.xticks([])  
        plt.yticks([])
```

Out[8]: ([], <a list of 0 Text yticklabel objects>)



```
In [9]: plt.imshow(B2, extent=[0,25, 0, 25], origin='lower',cmap='RdBu_r')
plt.colorbar()
plt.axis(aspect='image')
plt.xticks([])
plt.yticks([])
```

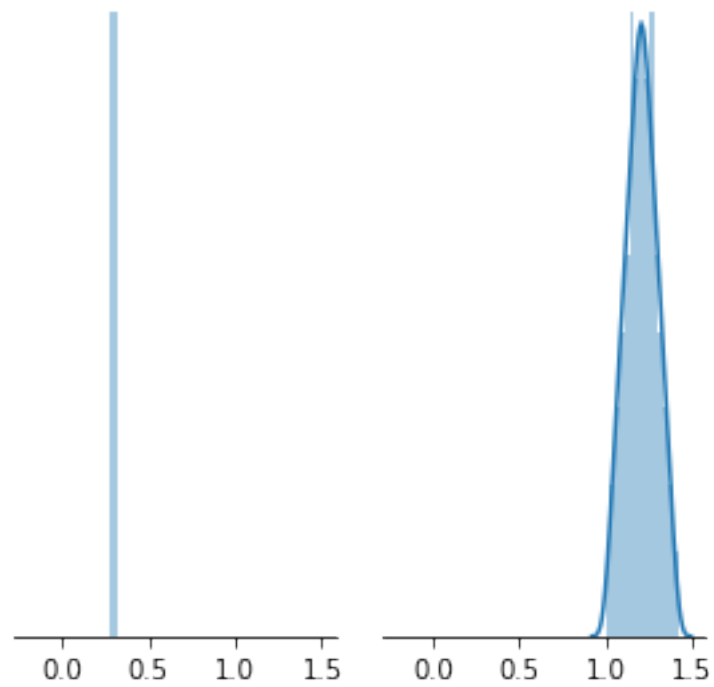
```
Out[9]: ([], <a list of 0 Text yticklabel objects>)
```



```
In [10]: B0=B0.reshape(-1,1)
B1=B1.reshape(-1,1)
B2=B2.reshape(-1,1)
```

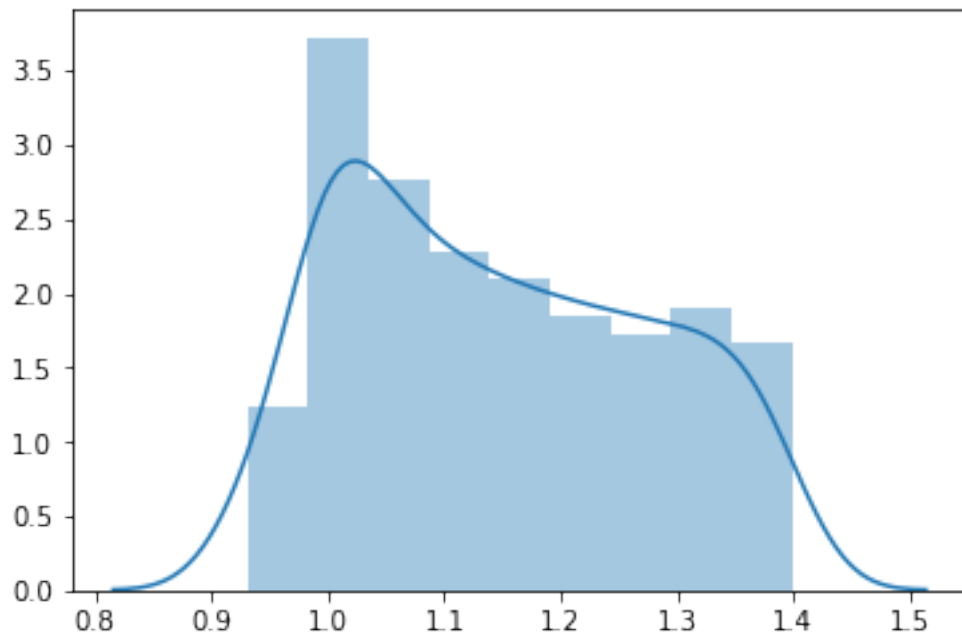
```
In [11]: f, axes = plt.subplots(1, 2, figsize=(4, 4), sharex=True)
sns.despine(left=True)
sns.distplot(B0,ax=axes[0])
sns.distplot(B1,ax=axes[1])

plt.setp(axes, yticks=[])
plt.tight_layout()
```



```
In [12]: sns.distplot(B2)
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x24e1fcd0828>
```



```
In [13]: lat=Y.reshape(-1,1)
lon=X.reshape(-1,1)

In [14]: x1=x1.reshape(-1,1)
x2=x2.reshape(-1,1)

In [15]: param = np.hstack([B0,B1,B2])
cons=np.ones_like(x1)
X=np.hstack([cons,x1,x2])
```

```
In [16]: param.shape,X.shape
```

```
Out[16]: ((625, 3), (625, 3))
```

**Creating y variable with Poisson distribution** Incorporating step from - Chapter 6. Simulating Generalized Linear Models, p. 153

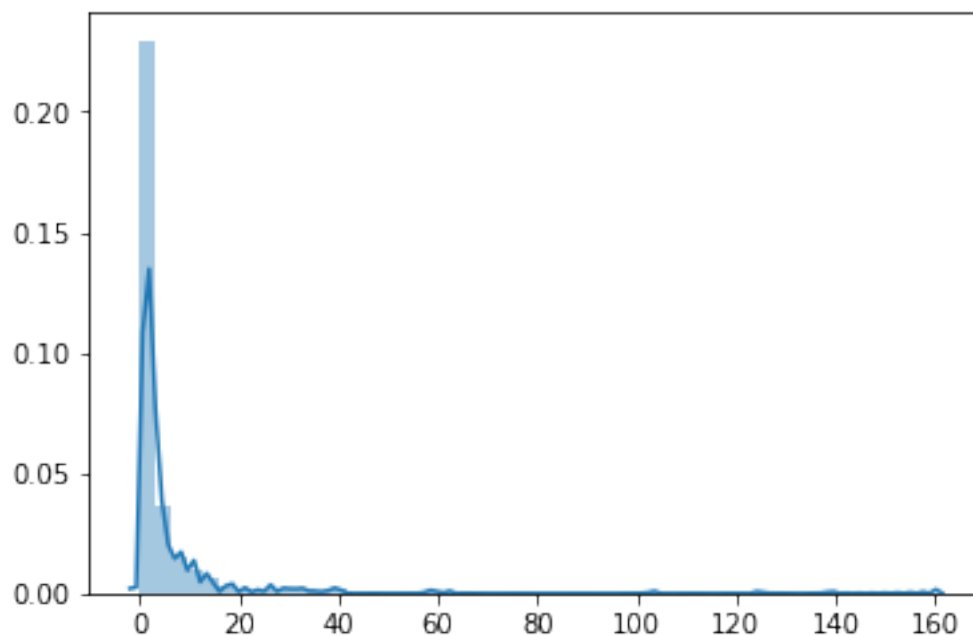
```
In [17]: #y
y=(np.exp((np.sum(X * param, axis=1)+error).reshape(-1, 1)))
y_new = np.random.poisson(y)
```

```
In [18]: y.shape
```

```
Out[18]: (625, 1)
```

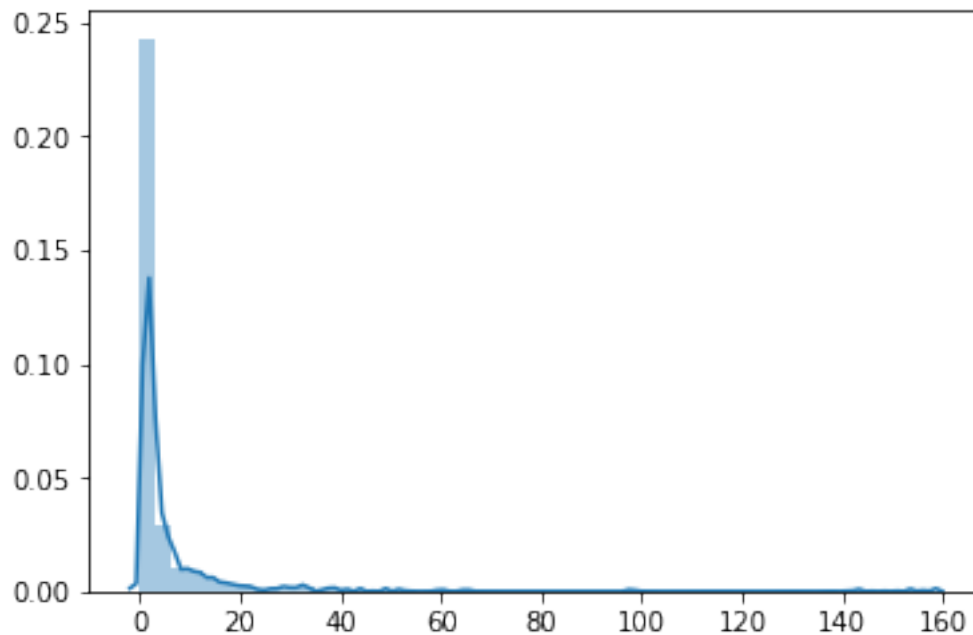
```
In [19]: sns.distplot(y)
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x24e1fdaaa90>
```



```
In [20]: sns.distplot(y_new)
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x24e1fef3828>
```



```
In [21]: coords = np.array(list(zip(lon,lat)))  
y = np.array(y_new).reshape((-1,1))
```

```
In [22]: x=x1  
x_std = (x-x.mean(axis=0))/x.std(axis=0)  
X=np.hstack([x1,x2])  
X_std = (X-X.mean(axis=0))/X.std(axis=0)
```

### 0.03 Univariate example

**First example: checking GWR and MGWR models with one independent variable and constant = False**

```
In [23]: bw=Sel_BW(coords,y,x,family=Poisson(),offset=None,constant=False)  
bw=bw.search()  
gwr_model=GWR(coords,y,x,bw,family=Poisson(),offset=None,constant=False).fit()  
bw
```

```
Out[23]: 43.0
```

```
In [24]: selector=Sel_BW(coords,y,x,multi=True,family=Poisson(),offset=None,constant=False)
        selector.search(verbose=True)
```

```
Current iteration: 1 ,SOC: 0.0
Bandwidths: 43.0
```

```
Out[24]: array([43.])
```

```
In [25]: mgwr_model=MGWR(coords,y,x,selector,family=Poisson(),offset=None,constant=False).fit()

HBox(children=(IntProgress(value=0, description='Inference', max=1), HTML(value='')))
```

### Bandwidth: Random initialization check

```
In [26]: selector.search(verbose=True,init_multi=600)
```

```
Current iteration: 1 ,SOC: 0.0201116
Bandwidths: 43.0
Current iteration: 2 ,SOC: 0.0
Bandwidths: 43.0
```

```
Out[26]: array([43.])
```

### Parameters check

```
In [27]: np.sum(((gwr_model.params-mgwr_model.params)==0.0))==True)
```

```
Out[27]: 625
```

```
In [28]: gwr_model.aic,mgwr_model.aic
```

```
Out[28]: (3458.604213385903, 3476.5331190431302)
```

```
In [29]: np.sum((gwr_model.predy-mgwr_model.predy==0))==True)
```

```
Out[29]: 625
```

## 0.0.4 Multivariate example

### Second example for multiple bandwidths

```
In [30]: bw=Sel_BW(coords,y,X,family=Poisson(),offset=None)
        bw=bw.search()
        gwr_model=GWR(coords,y,X,bw,family=Poisson(),offset=None).fit()
        bw
```



Out[30]: 103.0

```
In [31]: selector=Sel_BW(coords,y,X,multi=True,family=Poisson(),offset=None)
        selector.search(verbose=True)
```

```
Current iteration: 1 ,SOC: 0.0056638
Bandwidths: 400.0, 43.0, 43.0
Current iteration: 2 ,SOC: 0.0011879
Bandwidths: 400.0, 48.0, 44.0
Current iteration: 3 ,SOC: 0.0001425
Bandwidths: 400.0, 48.0, 44.0
Current iteration: 4 ,SOC: 2.35e-05
Bandwidths: 400.0, 48.0, 44.0
Current iteration: 5 ,SOC: 6.6e-06
Bandwidths: 400.0, 48.0, 44.0
```

Out[31]: array([400., 48., 44.])

### Bandwidths: Random initialization check

```
In [ ]: selector.search(verbose=True,init_multi=600)
```

```
In [32]: mgwr_model=MGWR(coords,y,X,selector,family=Poisson(),offset=None).fit()
```

```
HBox(children=(IntProgress(value=0, description='Inference', max=1), HTML(value='')))
```

### Parameters check

```
In [34]: max(gwr_model.predy-mgwr_model.predy)[:10]
```

Out[34]: array([39.63245513])

```
In [35]: gwr_model.aic, mgwr_model.aic
```

Out[35]: (651.1442030497388, 838.3196815747232)

### 0.0.5 Global model parameter check

```
In [36]: import statsmodels.api as sma
```

```
In [37]: X_glob=sma.add_constant(X)
```

```
In [38]: poisson_mod = sma.Poisson(y, X_glob)
```

```
In [39]: poisson_res = poisson_mod.fit(method="newton")
        print(poisson_res.summary())
```

Optimization terminated successfully.  
Current function value: 1.604261  
Iterations 7

#### Poisson Regression Results

Dep. Variable:	y	No. Observations:	625
Model:	Poisson	Df Residuals:	622
Method:	MLE	Df Model:	2
Date:	Thu, 04 Jul 2019	Pseudo R-squ.:	0.7832
Time:	11:47:08	Log-Likelihood:	-1002.7
converged:	True	LL-Null:	-4624.4
		LLR p-value:	0.000

---

	coef	std err	z	P> z	[0.025	0.975]
const	0.2894	0.036	7.972	0.000	0.218	0.361
x1	1.2009	0.019	62.905	0.000	1.163	1.238
x2	1.1628	0.021	54.264	0.000	1.121	1.205

```
In [40]: selector=Sel_BW(coords,y,X,multi=True,family=Poisson(),offset=None)
```

```
In [41]: selector.search(verbose=True,multi_bw_min=[625,625,625], multi_bw_max=[625,625,625])
```

```
Current iteration: 1 ,SOC: 0.0067699
Bandwidths: 625.0, 625.0, 625.0
Current iteration: 2 ,SOC: 0.0006156
Bandwidths: 625.0, 625.0, 625.0
Current iteration: 3 ,SOC: 0.0001351
Bandwidths: 625.0, 625.0, 625.0
Current iteration: 4 ,SOC: 4.2e-06
Bandwidths: 625.0, 625.0, 625.0
```

```
Out[41]: array([625., 625., 625.])
```

```
In [42]: mgwr_model=MGWR(coords,y,X,selector,family=Poisson(),offset=None).fit()
```

```
HBox(children=(IntProgress(value=0, description='Inference', max=1), HTML(value='')))
```

```
In [43]: mgwr_model.params[:5]
```

```
Out[43]: array([[0.24911341, 1.22606055, 1.19712663],
                [0.25309895, 1.2283422 , 1.19713621],
                [0.25724177, 1.23072827, 1.19708916],
                [0.26154154, 1.23322148, 1.19697204],
                [0.26599496, 1.23582348, 1.19676824]])
```

parameters similar for global Poisson model and forced global MGWR Poisson model