PySal Visualization Exploration

M. Kolak | Advanced Spatial Statistics, ASU | Spring 2016

Overview

Investigate and enhance visualization aspects of PySAL.

Initial Goals:

- Build on the identified goals and roadmap identified in PySAL, as summarized in the Visualization Project module home.
- Improved visualization module index and development of interactive visualization for the PySAL Visualization module.
- Explore aspects of further developing a PySAL Viz API to integrate PySAL visualization functions in dynamic web environments.

Project Objectives

Initial Objectives:

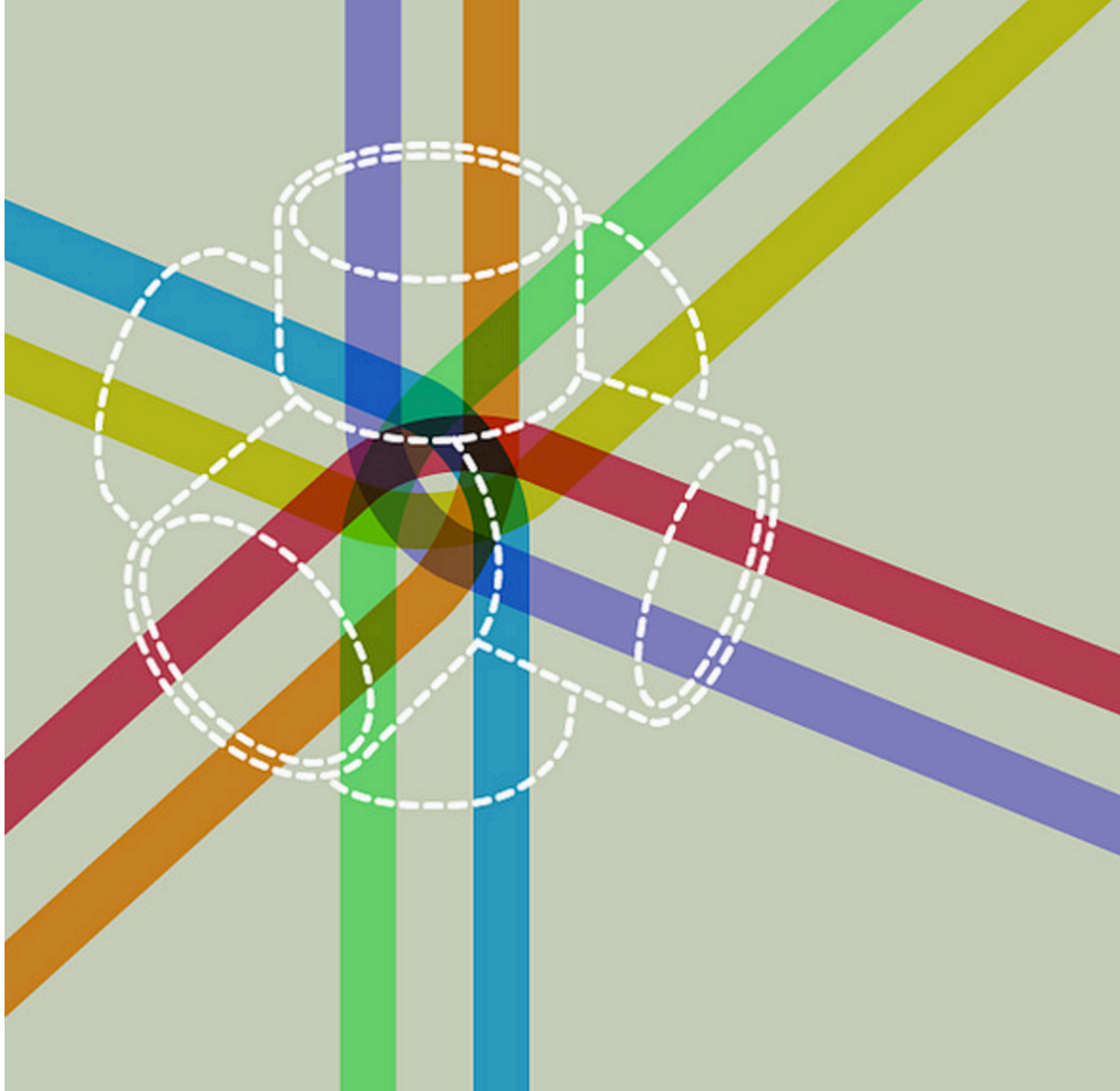
1. Review and Indexing of Visualization Notebooks *(Feb)*
2. Integrating Interactive Visualizations *(Mar-Apr)*
3. PySAL Visualization API *(May)*

Revised Objectives:

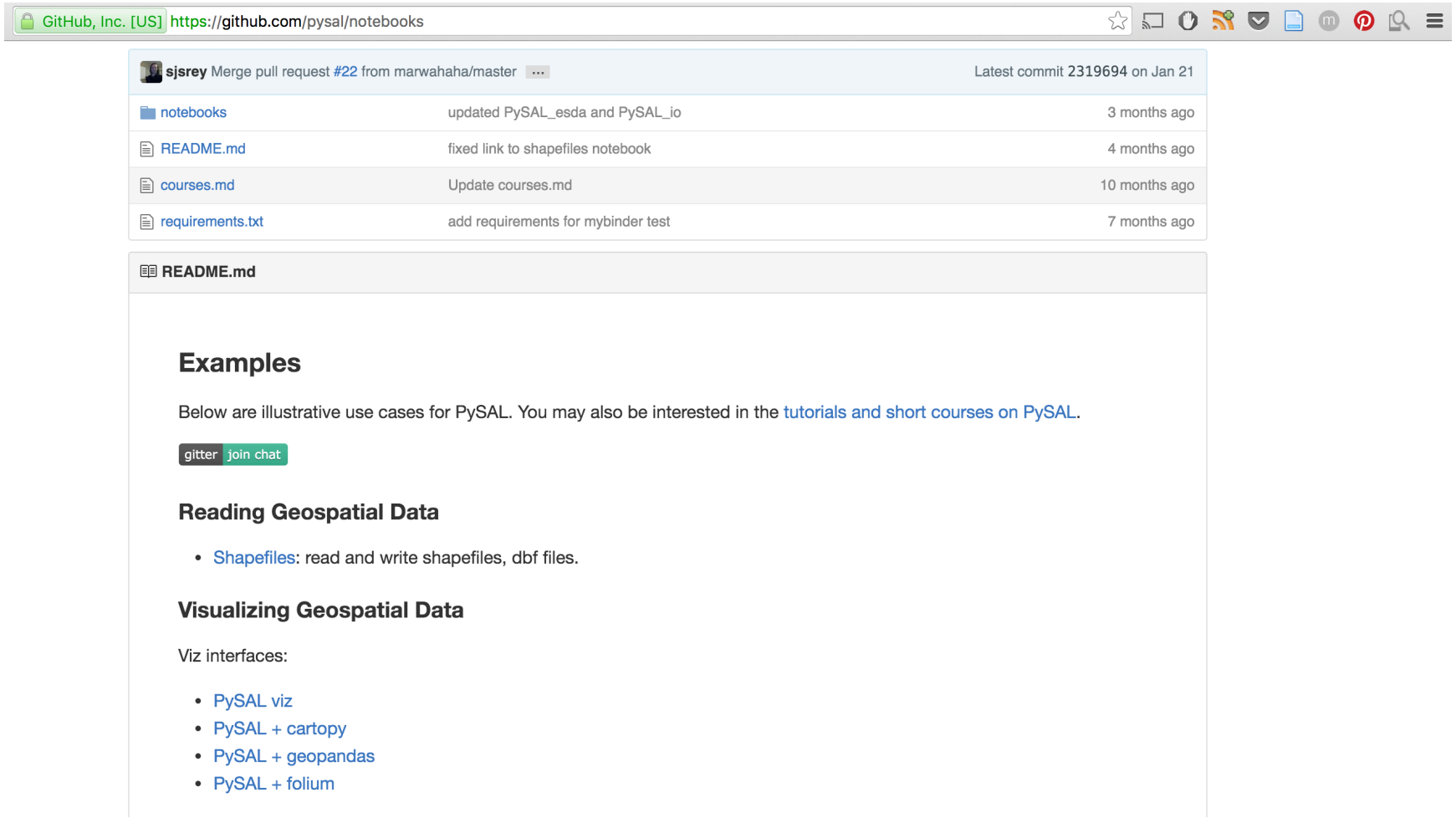
1. Review & Index Notebooks *(Feb)*
2. Notebook Gallery Development *(April)*
3. Document Interactive Visualization Options *(Mar)*
4. Automate Moran Plot as new function in new Viz class *(Apr)*

Review & Index Notebooks

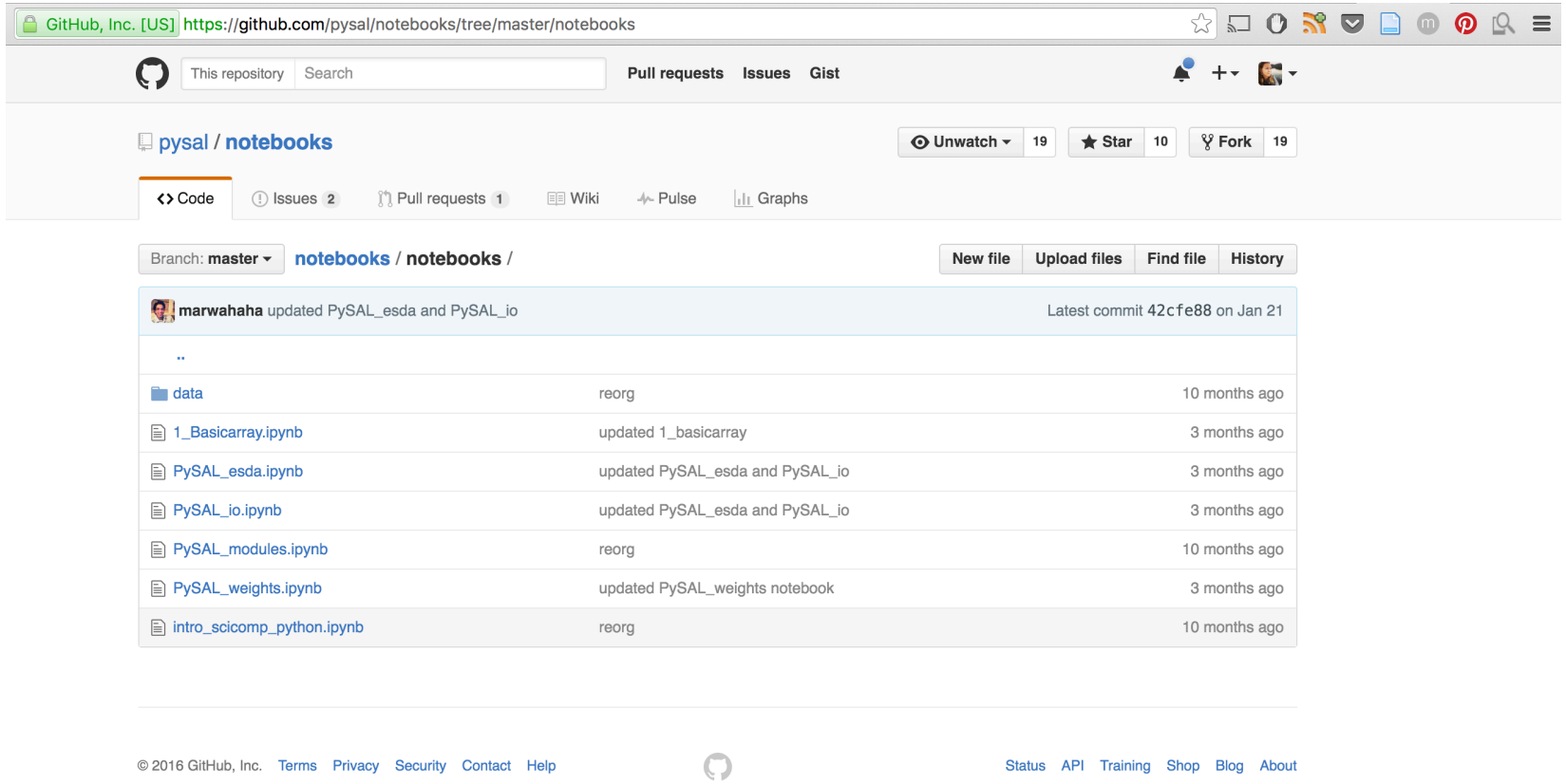
- Notebooks found throughout PySAL, its power users, and community
- How to best capture notebooks featured by, or featuring, PySAL?
- Consider options for collecting notebooks in a more efficient way



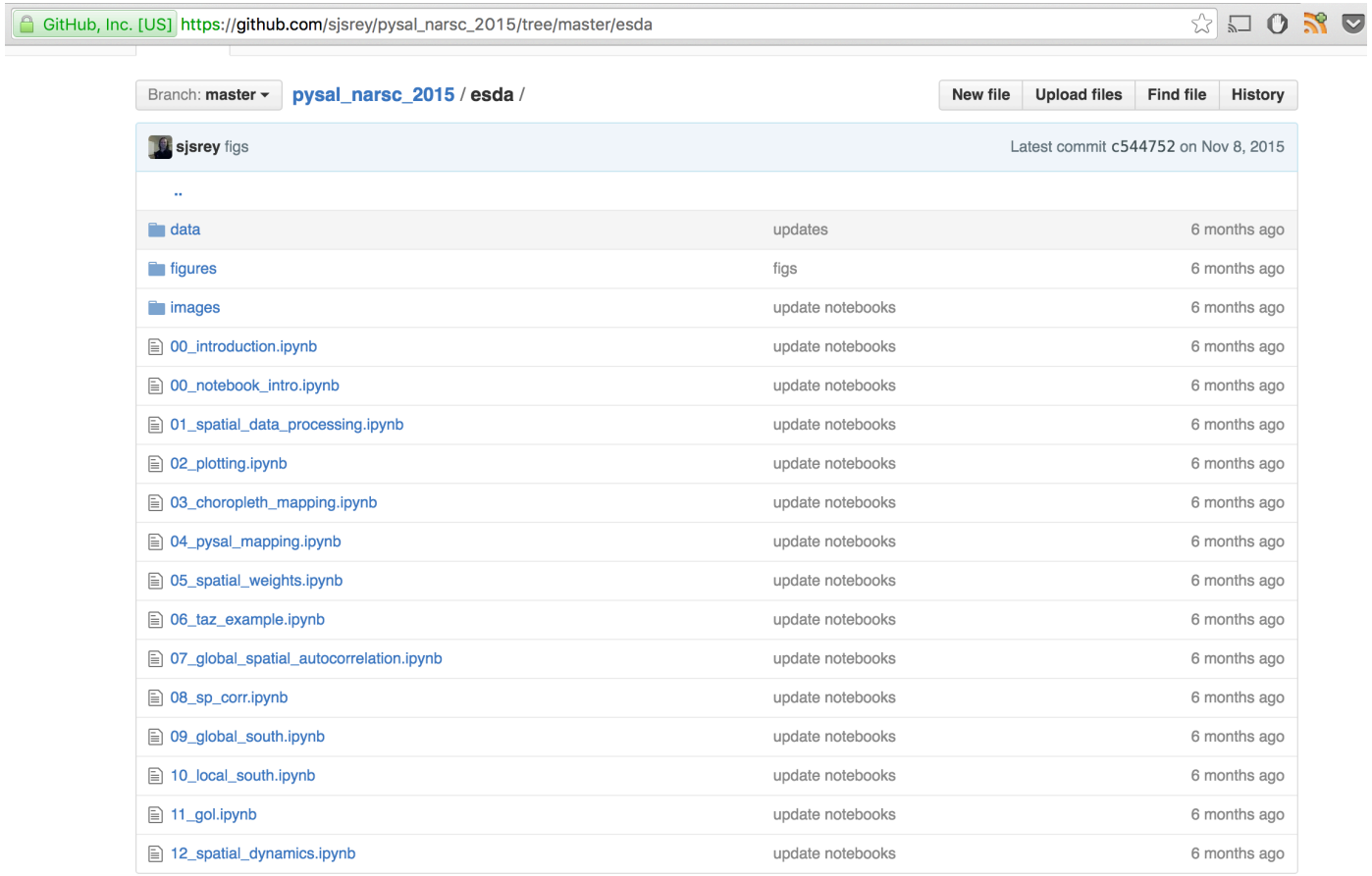
pysal/notebooks/README.md



pysal/notebooks/notebooks



PySAL team tutorials (/pysal_narsc2015)



GitHub, Inc. [US] https://github.com/sjsrey/pysal_narsc_2015/tree/master/esda

Branch: master ▾ **pysal_narsc_2015** / **esda** / New file | Upload files | Find file | History

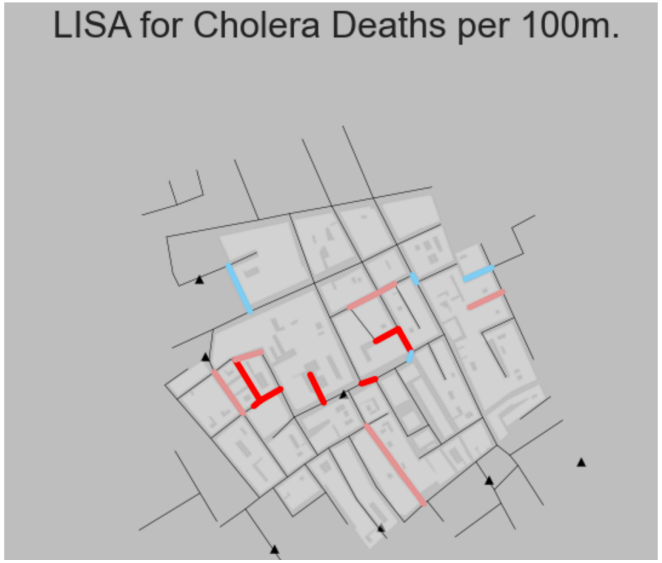
sjsrey figs — Latest commit c544752 on Nov 8, 2015

..		
data	updates	6 months ago
figures	figs	6 months ago
images	update notebooks	6 months ago
00_introduction.ipynb	update notebooks	6 months ago
00_notebook_intro.ipynb	update notebooks	6 months ago
01_spatial_data_processing.ipynb	update notebooks	6 months ago
02_plotting.ipynb	update notebooks	6 months ago
03_choropleth_mapping.ipynb	update notebooks	6 months ago
04_pysal_mapping.ipynb	update notebooks	6 months ago
05_spatial_weights.ipynb	update notebooks	6 months ago
06_taz_example.ipynb	update notebooks	6 months ago
07_global_spatial_autocorrelation.ipynb	update notebooks	6 months ago
08_sp_corr.ipynb	update notebooks	6 months ago
09_global_south.ipynb	update notebooks	6 months ago
10_local_south.ipynb	update notebooks	6 months ago
11_gol.ipynb	update notebooks	6 months ago
12_spatial_dynamics.ipynb	update notebooks	6 months ago

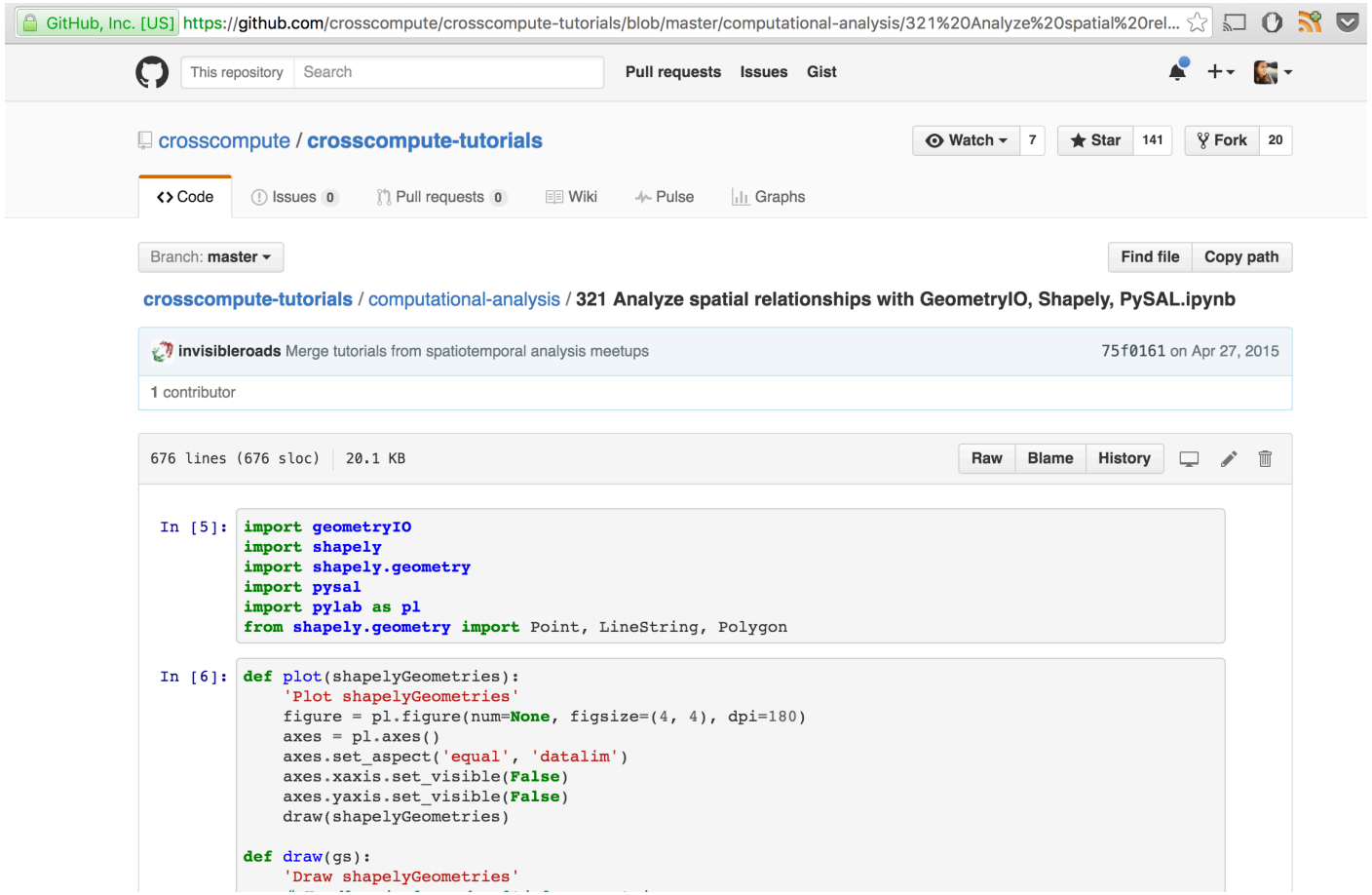
PySAL power user labs and examples

darribas.org/gds15/content/labs/lab_06.html

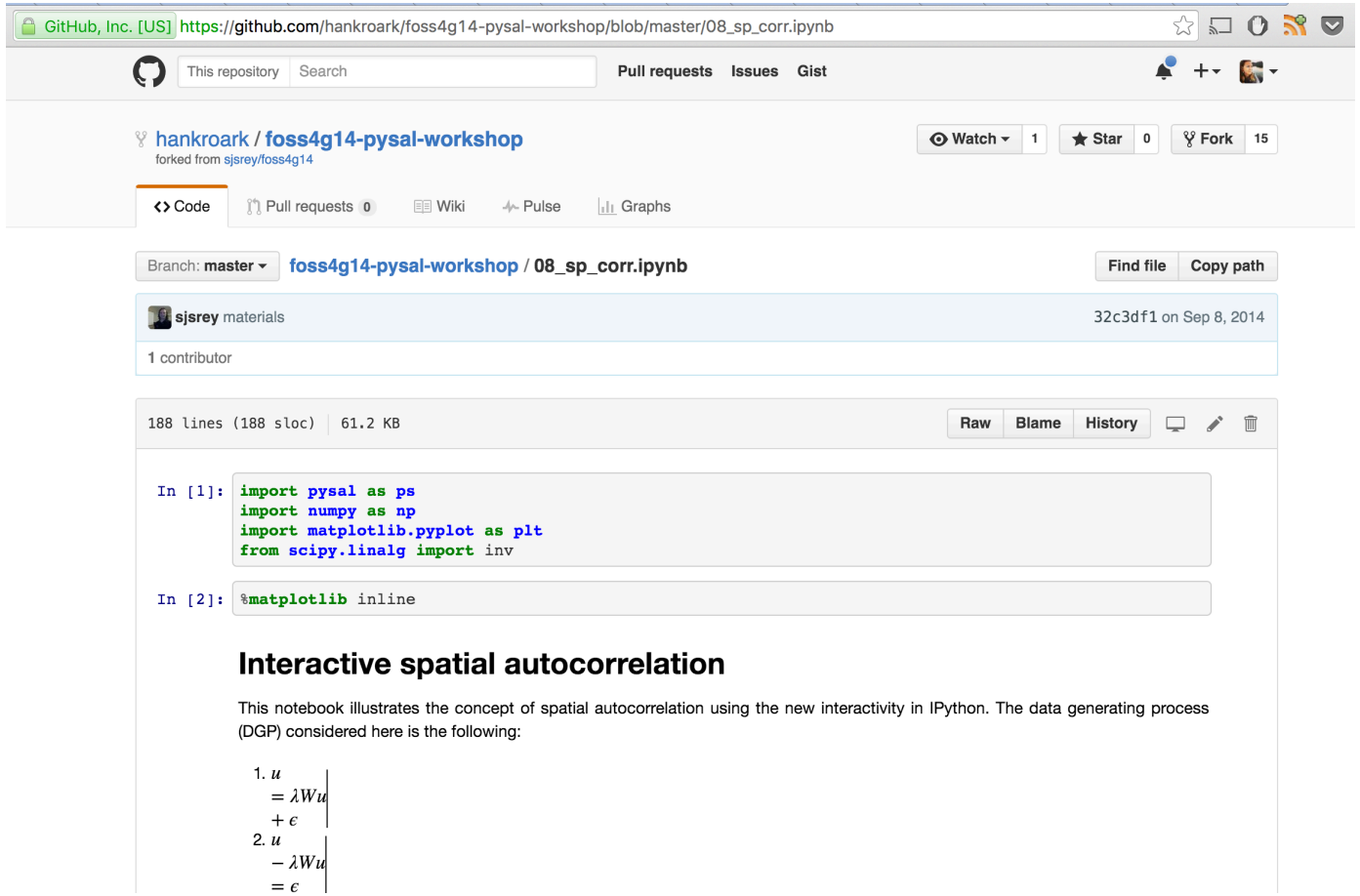
```
for line in ln:
    gpd.plotting.plot_multilinestring(ax, line, color='#83cef4', linewidth=5)
# Plot HL clusters
hl = js.loc[(js['quadrant']==4) & (js['significant']==True), 'geometry']
for line in hl:
    gpd.plotting.plot_multilinestring(ax, line, color='#e59696', linewidth=5)
# Plot pumps
xys = np.array([(pt.x, pt.y) for pt in pumps.geometry])
ax.scatter(xys[:, 0], xys[:, 1], marker='^', color='k', s=50)
# Style and draw
f.suptitle('LISA for Cholera Deaths per 100m.', size=30)
f.set_facecolor('0.75')
ax.set_axis_off()
plt.axis('equal')
plt.show()
```



Others using PySAL in analytic overviews



Others giving PySAL workshops (/foss4g14)

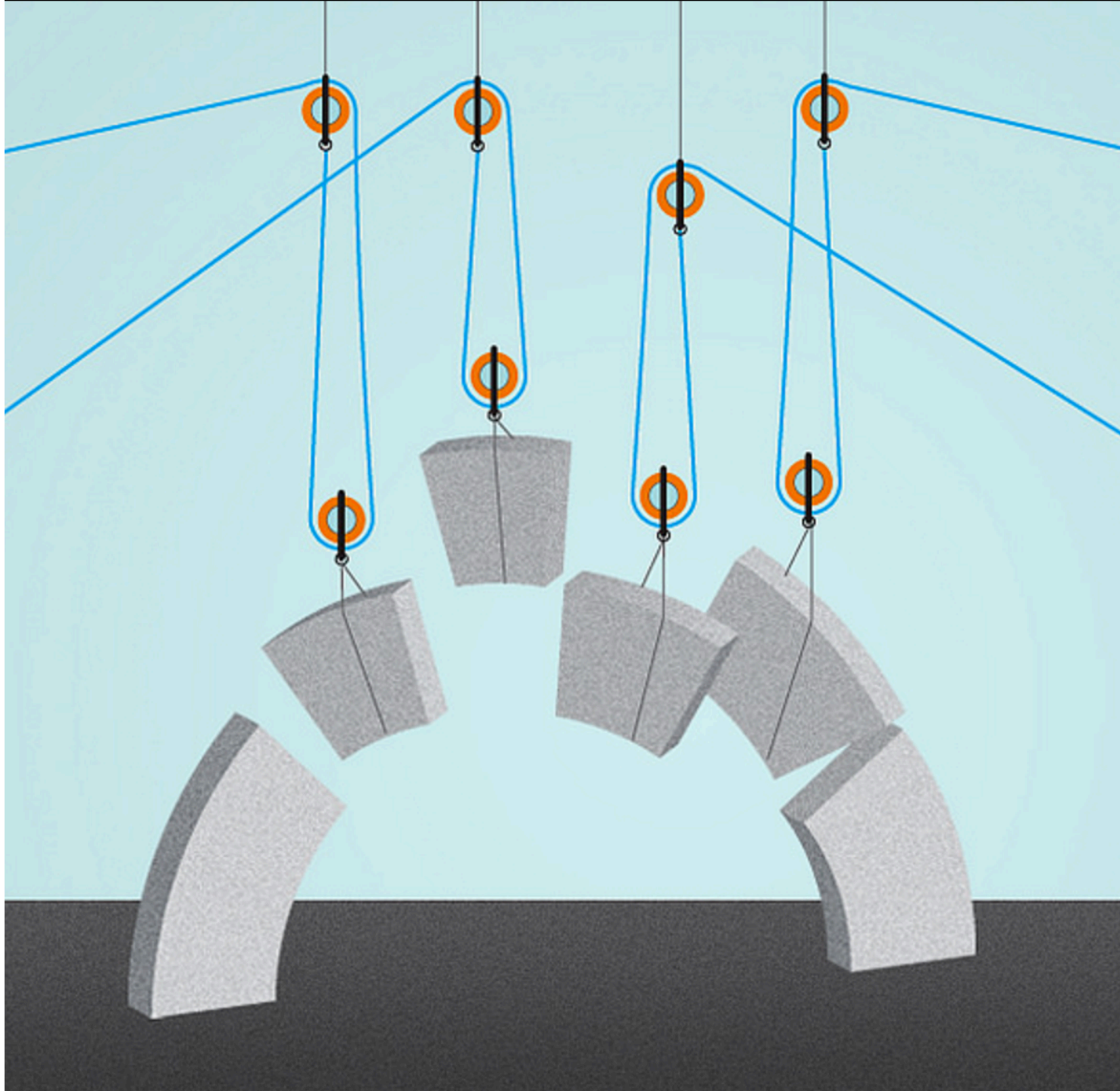


Review & Index Notebook: Deliverables

- Pysal/notebooks/README.md serves as existing featured collection
- New index of relevant notebooks found, by category (*temp*)
- Notebook gallery recommended

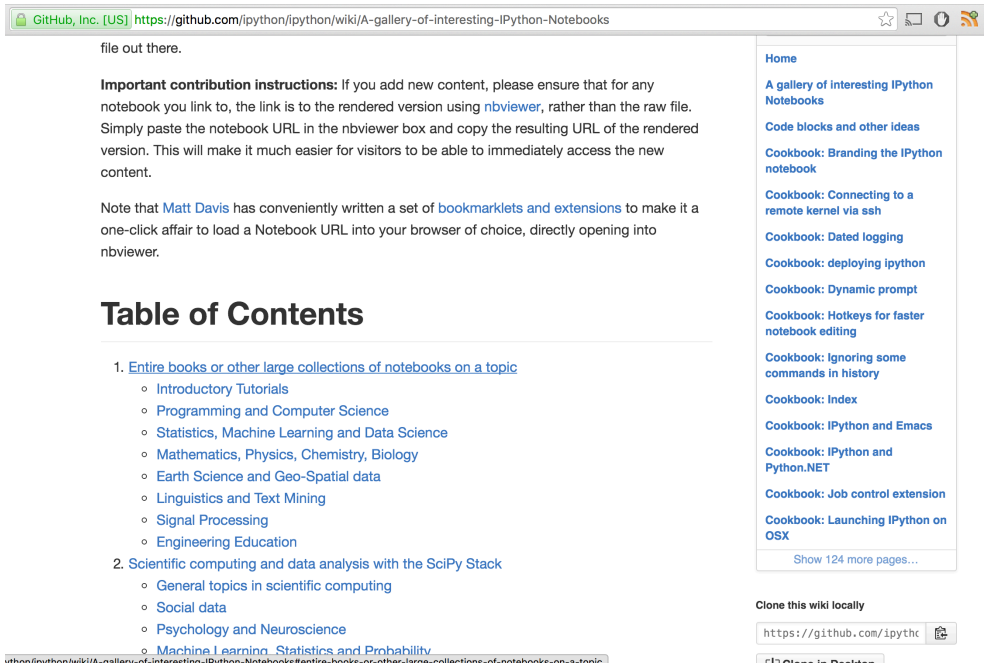
Notebook Gallery

- Visual > Not Visual
- Featured > Random
- Simple > Complex
- Automated > Manual



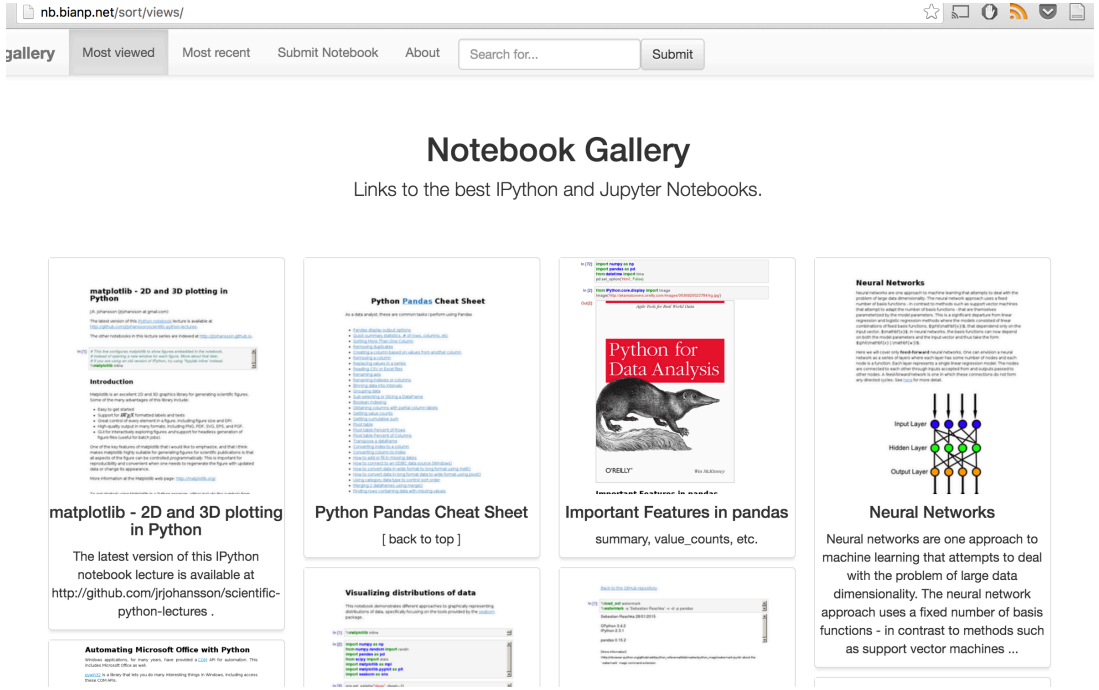
Notebook Gallery Options: links or pics?

Collection of links



https://github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks

Collection of thumbnails



http://nb.bianp.net/sort/views/

Notebook Gallery Considerations

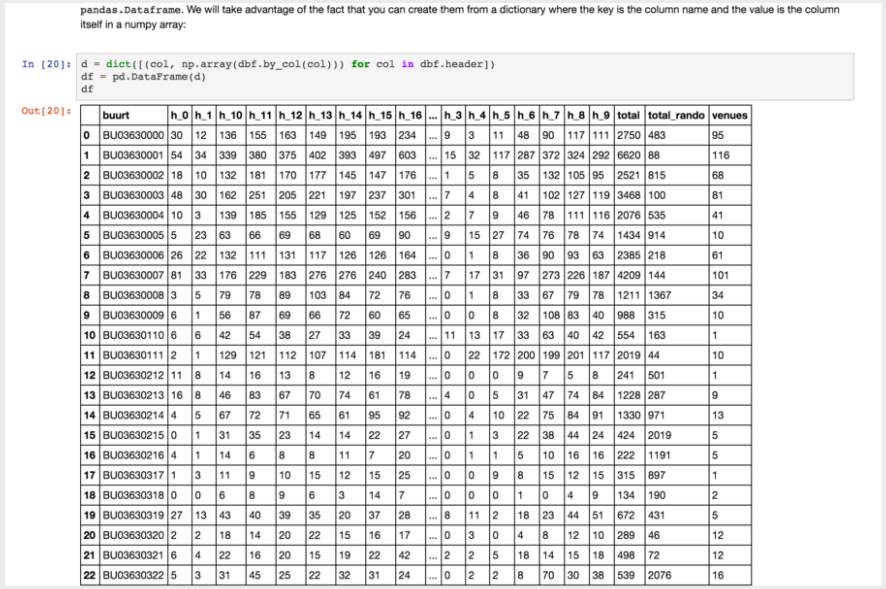
- Application should be light and easily served.
- Application should be responsive.
- PySAL team should be able to push featured, standard examples.
- Featured notebooks should pass build standards.
- Adding a new notebook to the gallery should be simple.
- Gallery application can have a phased development.
- The gallery should encourage and convert new users in PySAL ways.
- What does PySAL offer that other packages or suites don’t have?

Notebook Collection

- Long-term:
  - Automated and Easy to Use and View
  - Tagged entries, option for upvoting
  - Input link -> test for build quality -> approve as featured, recommend revision, or push to general page
  - Application templates: nbviews (python/django), pinterest (ruby/rails)
- Short-term:
  - Use github pages to serve
  - Simple HTML/CSS, Javascript with Bootstrap Gridded Template
  - README.md document with directions for adding (URL, picture)

Reading Geospatial Data

Reading Shapefiles



pandas.Dataframe. We will take advantage of the fact that you can create them from a dictionary where the key is the column name and the value is the column itself in a numpy array:

```
In [20]: d = dict([(col, np.array(dbf.by_col(col))) for col in dbf.header])
         df = pd.DataFrame(d)
         df
```

Out[20]:

	buurt	h_0	h_1	h_10	h_11	h_12	h_13	h_14	h_15	h_16	...	h_3	h_4	h_5	h_6	h_7	h_8	h_9	total	total_rando	venues
0	BU03630000	30	12	136	155	163	149	195	193	234	...	9	3	11	48	90	117	111	2750	483	95
1	BU03630001	54	34	339	380	375	402	393	497	603	...	15	32	117	287	372	324	292	6620	88	116
2	BU03630002	18	10	132	181	170	177	145	147	176	...	1	5	8	35	132	105	95	2521	815	68
3	BU03630003	48	30	162	251	205	221	197	237	301	...	7	4	8	41	102	127	119	3468	100	81
4	BU03630004	10	3	139	185	155	129	125	152	156	...	2	7	9	46	78	111	116	2076	535	41
5	BU03630005	5	23	63	66	69	68	60	69	90	...	9	15	27	74	76	78	74	1434	914	10
6	BU03630006	26	22	132	111	131	117	126	126	164	...	0	1	8	36	90	93	63	2385	218	61
7	BU03630007	81	33	176	229	183	276	276	240	283	...	7	17	31	97	273	226	187	4209	144	101
8	BU03630008	3	5	79	78	89	103	84	72	76	...	0	1	8	33	67	79	78	1211	1367	34
9	BU03630009	6	1	56	87	69	66	72	60	65	...	0	0	8	32	108	83	40	988	315	10
10	BU03630110	6	6	42	54	38	27	33	39	24	...	11	13	17	33	63	40	42	554	163	1
11	BU03630111	2	1	129	121	112	107	114	181	114	...	0	22	172	200	199	201	117	2019	44	10
12	BU03630212	11	8	14	16	13	8	12	16	19	...	0	0	0	9	7	5	8	241	501	1
13	BU03630213	16	8	46	83	67	70	74	61	78	...	4	0	5	31	47	74	84	1228	287	9
14	BU03630214	4	5	67	72	71	65	61	95	92	...	0	4	10	22	75	84	91	1330	971	13
15	BU03630215	0	1	31	35	23	14	14	22	27	...	0	1	3	22	38	44	24	424	2019	5
16	BU03630216	4	1	14	6	8	8	11	7	20	...	0	1	1	5	10	16	16	222	1191	5
17	BU03630317	1	3	11	9	10	15	12	15	25	...	0	0	9	8	15	12	15	315	897	1
18	BU03630318	0	0	6	8	9	6	3	14	7	...	0	0	0	1	0	4	9	134	190	2
19	BU03630319	27	13	43	40	39	35	20	37	28	...	8	11	2	18	23	44	51	672	431	5
20	BU03630320	2	2	18	14	20	22	15	16	17	...	0	3	0	4	8	12	10	289	46	12
21	BU03630321	6	4	22	16	20	15	19	22	42	...	2	2	5	18	14	15	18	498	72	12
22	BU03630322	5	3	31	45	25	22	32	31	24	...	0	2	2	8	70	30	38	539	2076	16

Read and write shapefiles and dbf files.

CSV to Points in Python

830562	11:51.3	NaN	True	1	2214817-20150616	2394354	280407	2	QUALITY TRUCK TRAILER REPAIR INCORPORATED
830563	11:51.3	NaN	True	1	39187-20150616	2393075	33878	1	SUPERMERCADO CHAPULTEPEC I
830567	11:51.3	NaN	True	1	2404874-20150528	2404874	398795	1	ADRIAN ESTRAD ROLANDO SEGU
830568	11:51.3	NaN	True	1	1144544-20150516	2387769	215498	1	CARRIBANA BAR GRILL, INC.
830570	11:51.3	NaN	True	1	2404813-20150527	2404813	361769	3	ROQUE BEAUTY LOUNGE-THE LA SPOT, INC.

1000 rows × 35 columns

```
In [80]: inCSV  = 'businesslic.csv'
         shpOut = 'businessSites.shp'
         lng = 'longitude'
         lat = 'latitude'
```

```
In [90]: schema = { 'geometry': 'Point', 'properties': {
                           'SITEID': 'str:24',
                           'lic_desc': 'str:24' }}

         df = pd.read_csv(inCSV)

         data = df
         with collection(shpOut, "w", "ESRI Shapefile", schema) as output:
             for index, row in data.iterrows():
                 point = Point(row[lng], row[lat])
                 output.write({
                           'properties': {
                           'SITEID': 'str:24',
                           'lic_desc': 'str:24' },
                           'geometry': mapping(point)
                       })
```

Turn CSV file with lat/long data to point file with geometry objects.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other head content must come *after* these tags -->
    <title>PySAL Notebook Gallery</title>

    <!-- Bootstrap -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-
    1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">



    <!-- Latest compiled and minified CSS -->


<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css" integrity="sha384-
fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r" crossorigin="anonymous">


    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-
      0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>

    <div class='navbar navbar-default navbar-static-top'>
      <div class='container-fluid'>
        <div class="navbar-header">
          <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class='navbar-brand' href='index.html'>PySal Notebook Gallery</a>
```

PySAL NB Gallery Instructions

PysalNBGallery

Update gallery with the following steps:

- Save a screenshot of the notebook in the images folder
- Add the image, link, and description to the appropriate category in index.html

For example, if I'd like to add a new notebook to the Spatial Dynamics section, insert here:

```
<div class='container-fluid'>
<div class="jumbotron">
  <div class='row'></div>

  <h2  align="left">Space-Time Analysis</h2> <br>
  <div class='row'>

<div class='col-md-6'>
 <a href="https://github.com/sjsrey/aerus2015/blob/master/esda/12_spatial_dynamics.ipynb">
 <img src="images/dynamics.png"style="height:300px;"> </a><br><br>
 <p> Spatial Dynamics. </p>
</div>

<div class='col-md-6'>
 <a href="LINK TO YOUR NOTEBOOK HERE">
 <img src="images/LINK TO IMAGE HERE"style="height:300px;"> </a><br><br>
 <p> DESCRIPTION OF YOUR NOTEBOOK HERE. </p>
</div>

    </div></div></div>
```

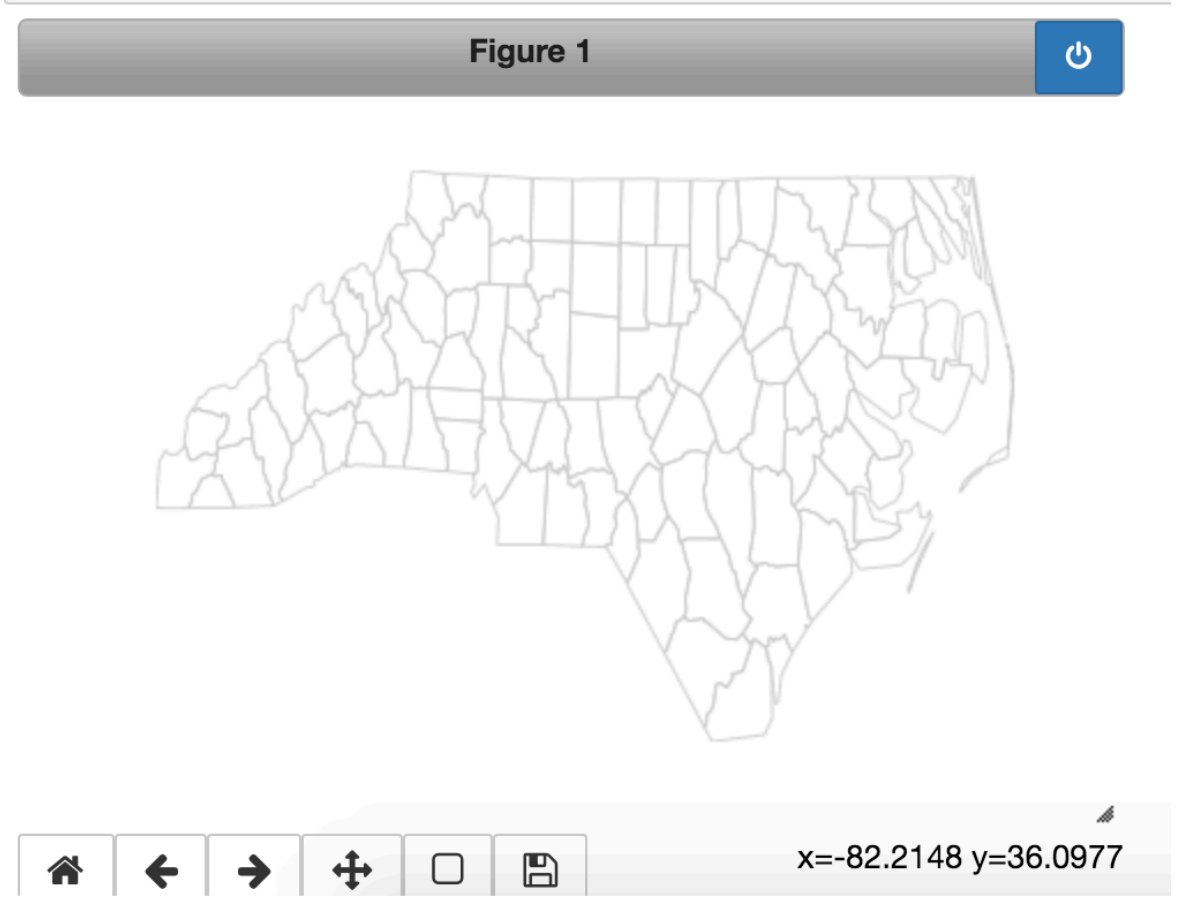
Interaction Exploration

- What are options for enabling interaction in notebooks?
- (ex) Click on point in Moran Plot, and the corresponding polygon is highlighted
- Tasks to consider: zoom, move, hover, pop-up information, brush, link

Notebook Visualization Exploration

	About	Pros	Cons
Matplotlib %inline	Matplotlib object enabled within notebook	• Enables matplotlib image within notebook	• no interaction
Matplotlib %notebook	Matplotlib object enabled with interaction within notebook	• Simplicity: 1-line addition to enable existing matplotlib images • Zoom, move, hover	• Requires kernel running • Pop-up & much advanced interaction not enabled as default, not straightforward • **Rasterized output of vectors**
Mpld3	Converts matplotlib object to javascript, and employs D3 library	• Once object has converted to javascript correctly, limitless options & flexibility	• More complicated • Necessitates additional dependency; long-term viability • Event-handling issues remain to separate vector from raster mpl

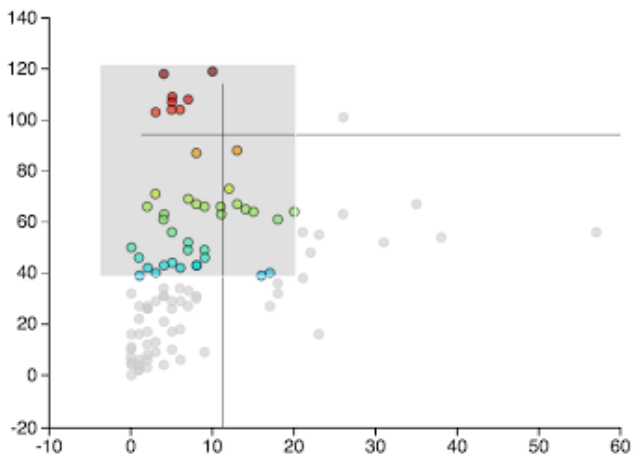
%matplotlib notebook

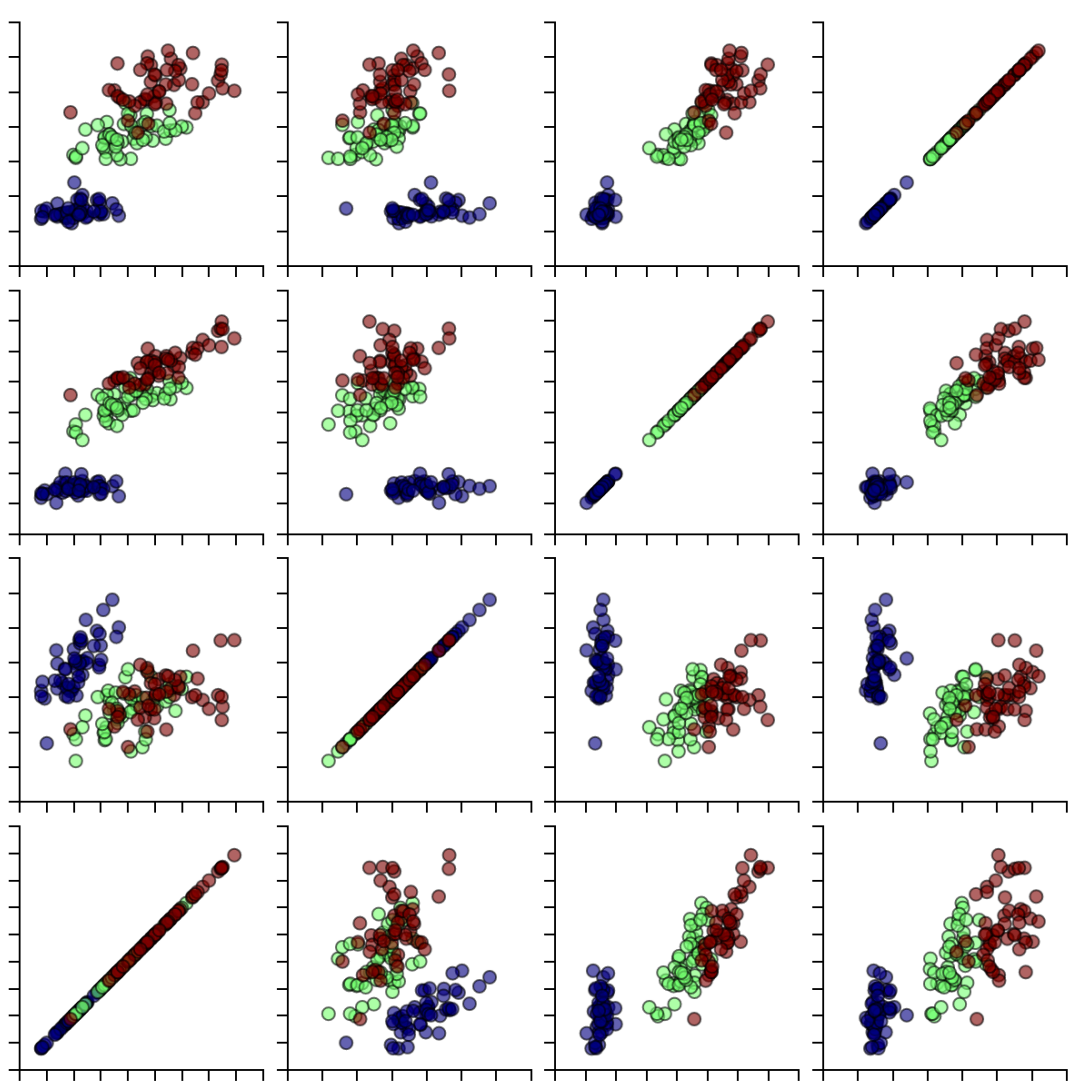


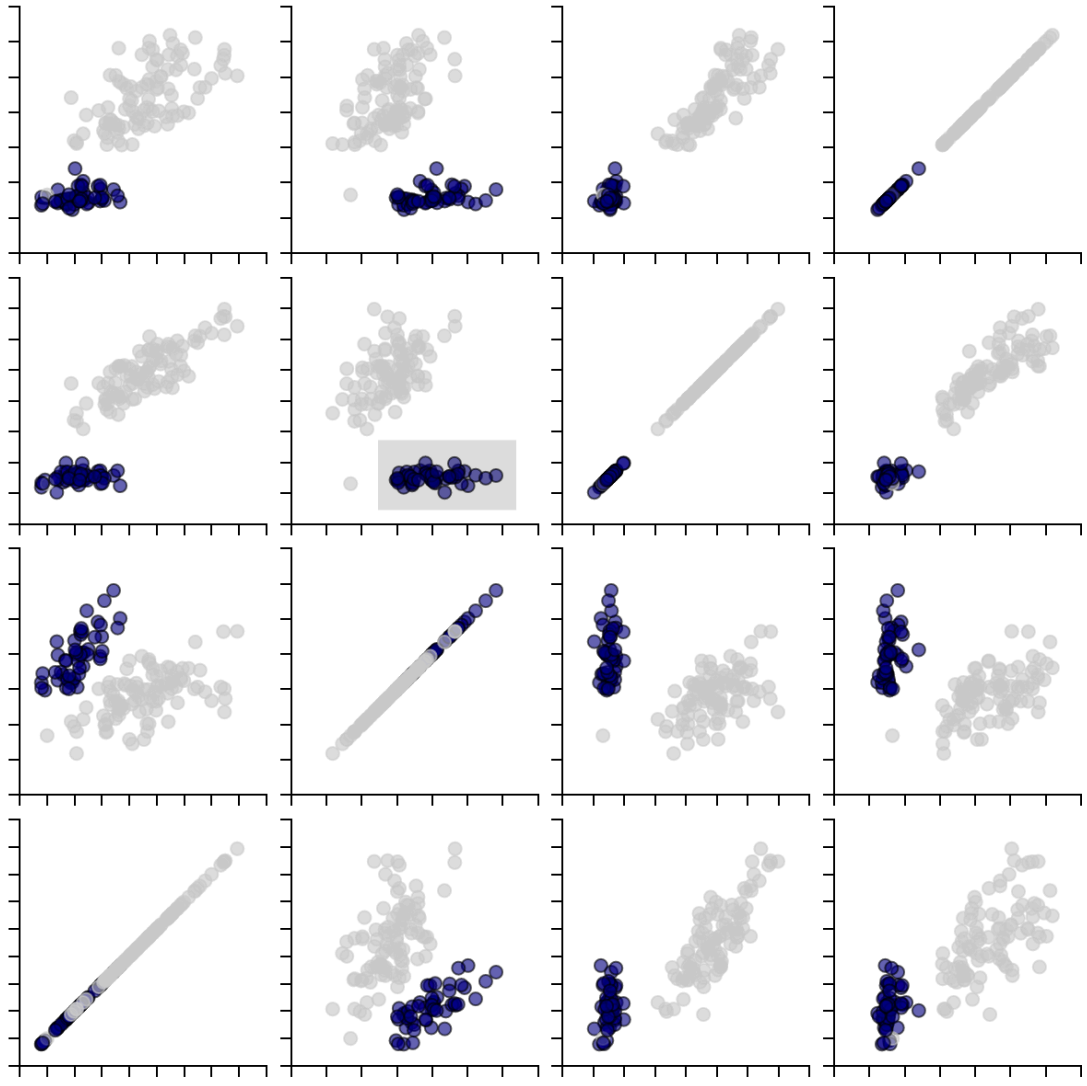
MPLD3 Exploration

In [21]:
```
f = ps.open(ps.examples.get_path("sids2.dbf"))
bir79 = np.array(f.by_col('BIR79'))
sids79 = np.array(f.by_col('SID79'))
w = ps.open(ps.examples.get_path("sids2.gal")).read()
w_sids79 = ps.lag_spatial(w, sids79)
x = sids79
y = w_sids79

# dither the data for clearer plotting
x += 0.1 * np.random.random(x.shape)

# Setup the figure and axis
fig, ax = plt.subplots(1, figsize=(7, 5))
# Add vertical and horizontal lines
plt.axvline(0, c='k', alpha=0.5)
plt.axhline(0, c='k', alpha=0.5)

points = scatter(x, y, c=y, s=40, alpha=0.6)

plugins.connect(fig, plugins.LinkedBrush(points))

mpld3.display()
```

Out[21]:

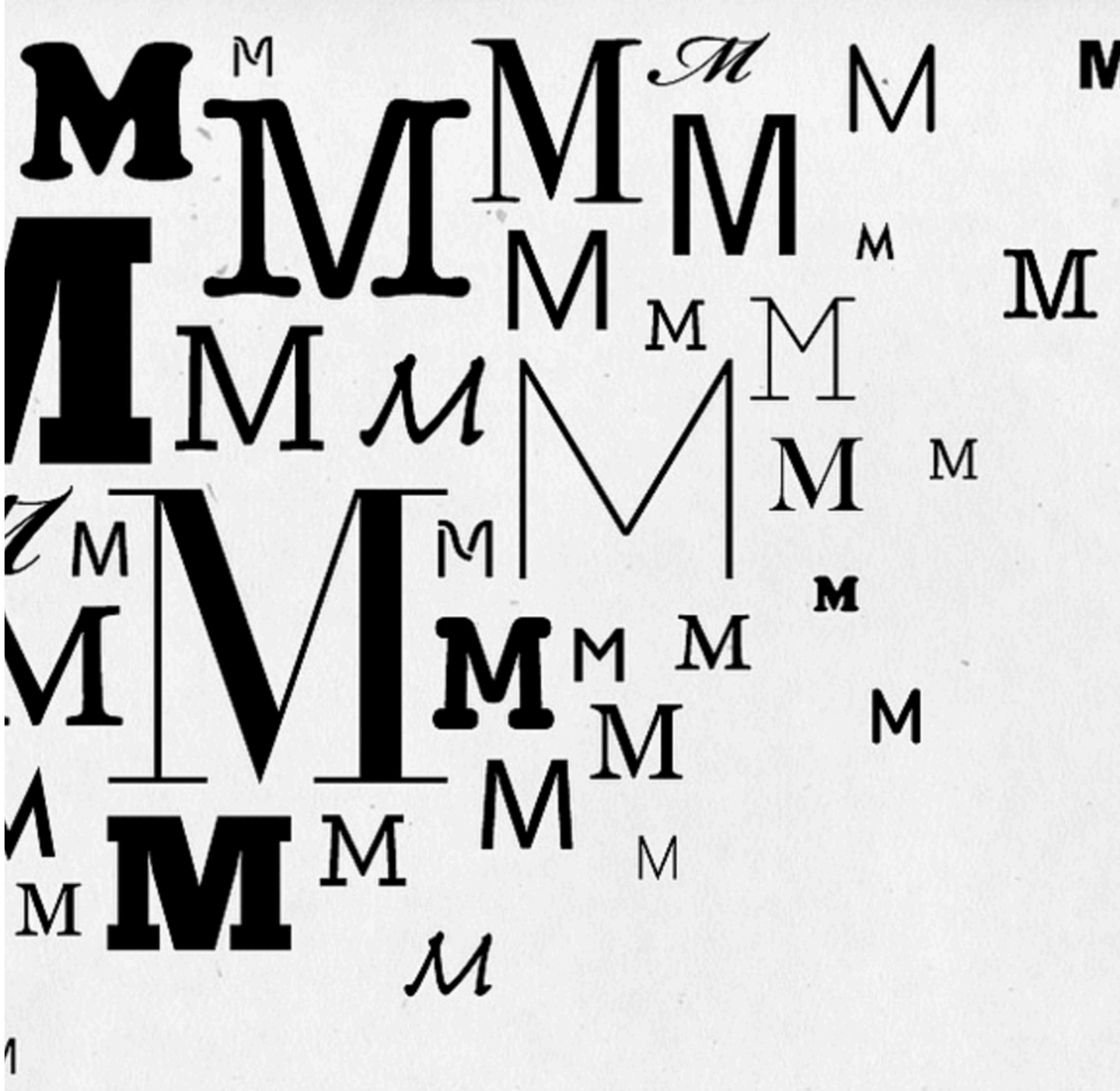






Moran Plot Function

- pysal/contrib/viz
- Cooked or Canned View for easy plotting
- Inputs: variable and weight, or MI?
- Output: figure with default style
- Allow for customization
- Forward thinking: PySAl ViZ API



Moran’s I Prep

Moran's I Analysis

Open data and the weight files.

In [220]:
```
SIDS = ps.open(ps.examples.get_path("sids2.dbf"))
w = ps.open(ps.examples.get_path("sids2.gal")).read() # spatial weights instance
```

Assign variable names from columns in data array

In [86]:
```
SID79 = np.array(SIDS.by_col('SID79')) # an event variable measured across n spatial units
BIR79 = np.array(SIDS.by_col('BIR79')) # a population-at-risk variable measured across n spatial units
SIDR79 = np.array(SIDS.by_col("SIDR79"))
```

Calculate Moran's I

In [239]:
```
# class pysal.esda.moran.Moran_Rate(e, b, w, adjusted=True, transformation='r', permutations=999, two_tailed=True)

mir = ps.esda.moran.Moran_Rate(SID79, BIR79, w, adjusted=True, transformation='r', permutations=9999, two_tailed=False)
```

Moran’s I for Rate Variables Review

Review Moran's I Statistic

In [242]:
```
def moranSummary(mir):
    mir = mir
    print
    print 'Adjusted Moran’s I Global Autocorrelation Statistic for Rate Variables'
    print
    print 'Moran Rate Summary Report'
    print '========================'
    print 'mir.i        {}   observed value of Moran’s I'.format("%6.4f" % mir.I)
    print 'mir.EI      {}   expected value under normality assumption'.format("%6.4f" % mir.EI)
    print 'mir.EI_sim  {}   average value of I from permutations  '.format("%6.4f" % mir.EI_sim)
    print 'mir.z_sim    {}   standardized I based on permutations'.format("%6.4f" % mir.z_sim)
    print 'mir.p_z_sim  {}   p-value based on standard normal approximation from permutations'.format("%6.4f" % mir.p_z_sim)
    print 'mir.p_sim    {}   p-value based on permutations'.format("%6.4f" % mir.p_sim)
```

In [243]:
```
moranSummary(mir)
```

```
Adjusted Moran’s I Global Autocorrelation Statistic for Rate Variables

Moran Rate Summary Report
========================
mir.i        0.1662   observed value of Moran’s I
mir.EI      -0.0101   expected value under normality assumption
mir.EI_sim  -0.0095   average value of I from permutations
mir.z_sim    2.6247   standardized I based on permutations
mir.p_z_sim  0.0043   p-value based on standard normal approximation from permutations
mir.p_sim    0.0084   p-value based on permutations
```

Thinking about operationalizing a Moran Plot

In [218]:

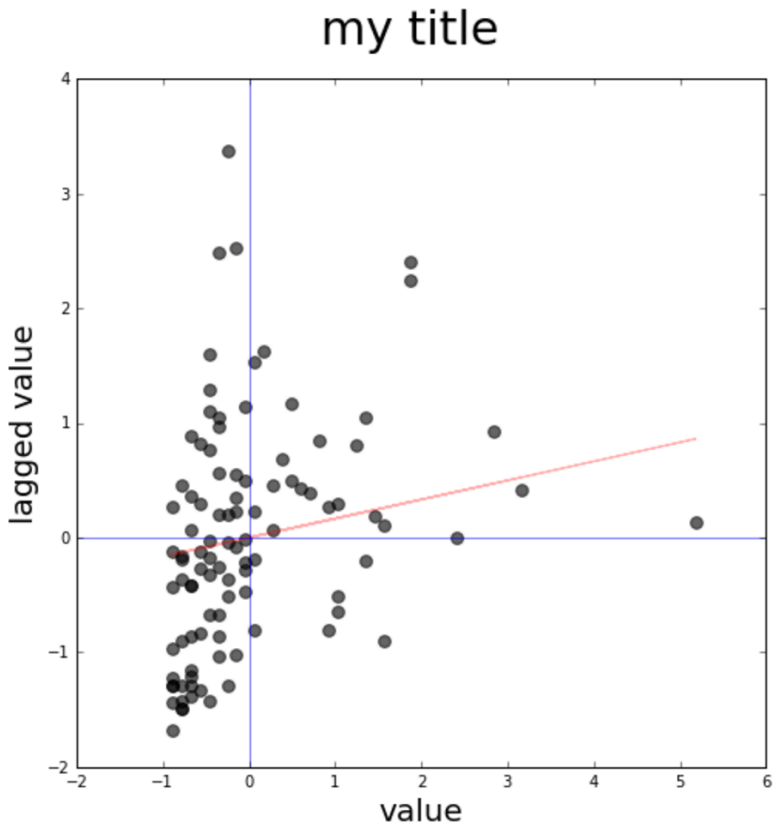
```
## Operationialize a Moran Plot -- Very Basic:
## Input arguments: variable and weight
## maps.plot_moran(var, w)
## options: title='title', xlabel='xlabel', ylabel='ylabel', figsize=()

var = SID79
w = w
slag = ps.lag_spatial(w, var)

y_std   = (var - var.mean())/var.std()
yl_std  = (slag - slag.mean())/slag.std()

#custom
xlabel = "xlabel"
ylabel = "ylabel"
title = "title"
custom = (8,8)

#custom
fig1 = plt.figure(figsize=custom)
plt.xlabel(xlabel, fontsize=20)
plt.ylabel(ylabel, fontsize=20)
plt.suptitle(title, fontsize=30)

plt.scatter(y_std, yl_std, s=60, color='k', alpha=.6)
plt.plot(y_std,mir*y_std, color='r', alpha=.3)

plt.axvline(0, alpha=0.5)
plt.axhline(0, alpha=0.5)

plt.show()
```

Moran Plot function (working iteration)

In [18]:
```
def moran(var, w, xlabel='', ylabel='', title='', custom=(5,5)):

    slag = ps.lag_spatial(w, var)

    y_std   = (var - var.mean())/var.std()
    yl_std  = (slag - slag.mean())/slag.std()

    fig1 = plt.figure(figsize=custom)
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.suptitle(title, fontsize=30)

    plt.scatter(y_std, yl_std, s=60, color='k', alpha=.6)

    plt.axvline(0, alpha=0.5)
    plt.axhline(0, alpha=0.5)

    plt.show()
```

Calling new moran function

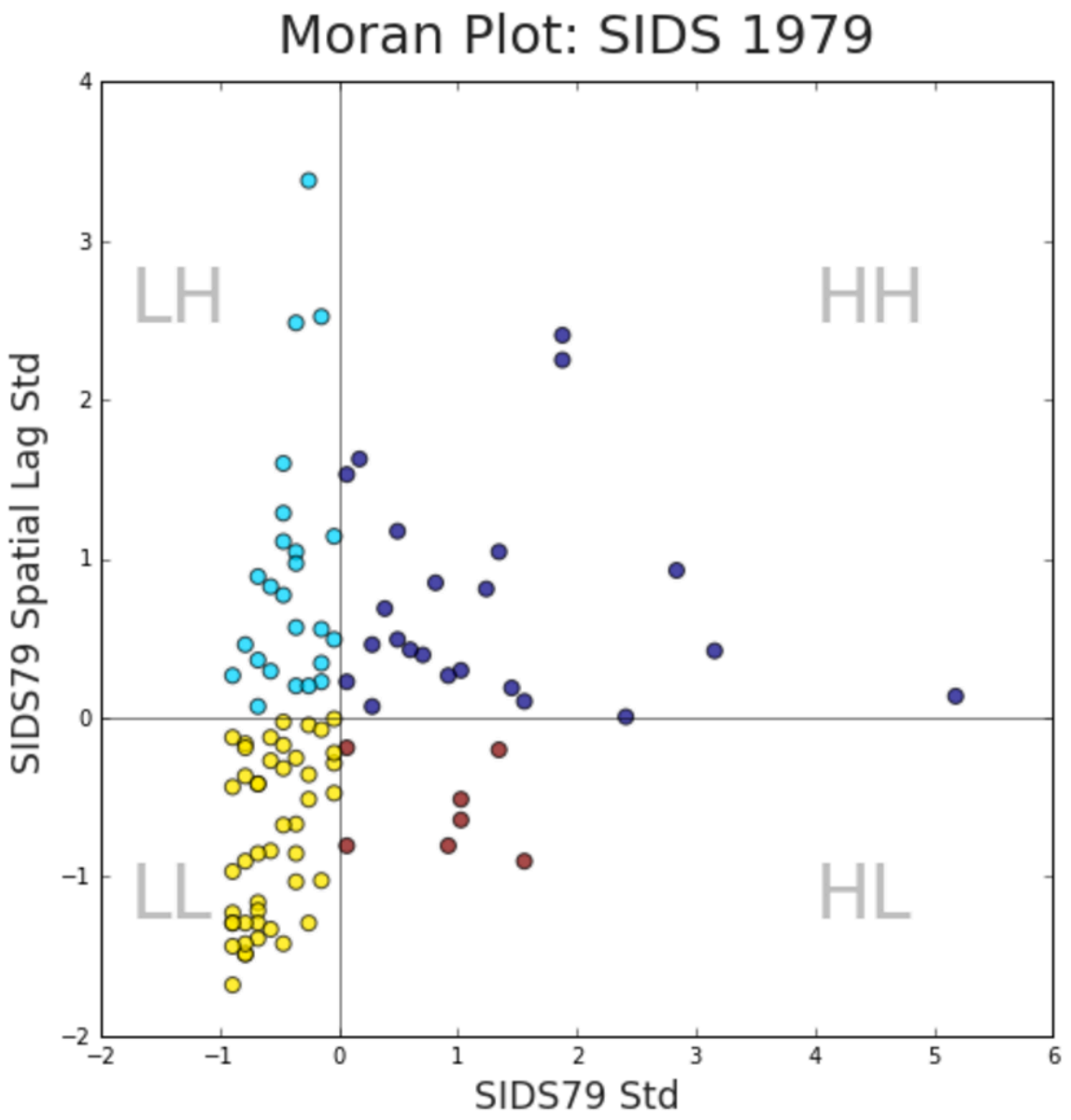
```
In [213]: moran(var,w,"value","lagged value","my title", (8,8) )
```



Moran function issues

Multiple considerations left:

- Inputs: (var, w) versus (mi)
- Recalculating standardization in plot function: global vs local variables
- Weight assumptions
- Pulling different MI slopes within the same graph
- How much to we cook the plot?
- What to call the class? (plot vs. canned)
- Highlighting colors in different quadrants, and/or highlighting certain p-values?
- Default style: (ugly) matplotlib vs (pretty) seaborn? Dependency issues?



Future Work

There is much left to do!

- Refine canned.moran() as a template for future canned views
- Test canned.moran() via PySAL API in ChainBuilder interface
- Develop additional canned views
- Troubleshoot event handling issue in matplotlib objects to allow for proper interaction
- Finalize index of notebooks to be added or stored in gallery

https://github.com/Makosak/PySalViz

https://www.flickr.com/photos/opensourceway/


Q
A