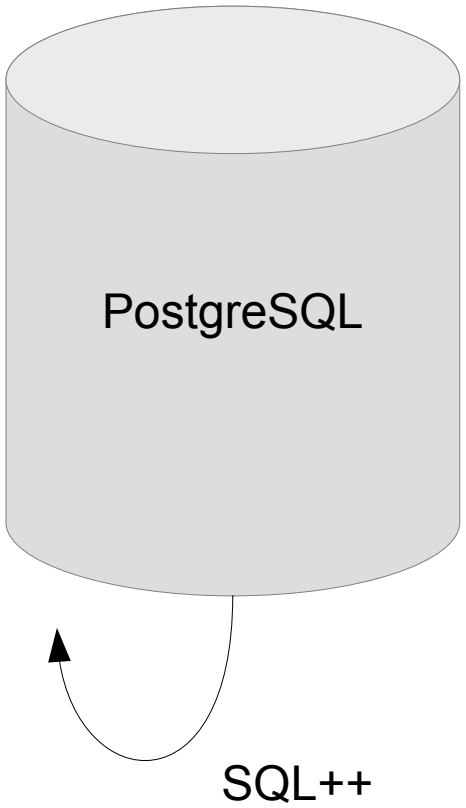
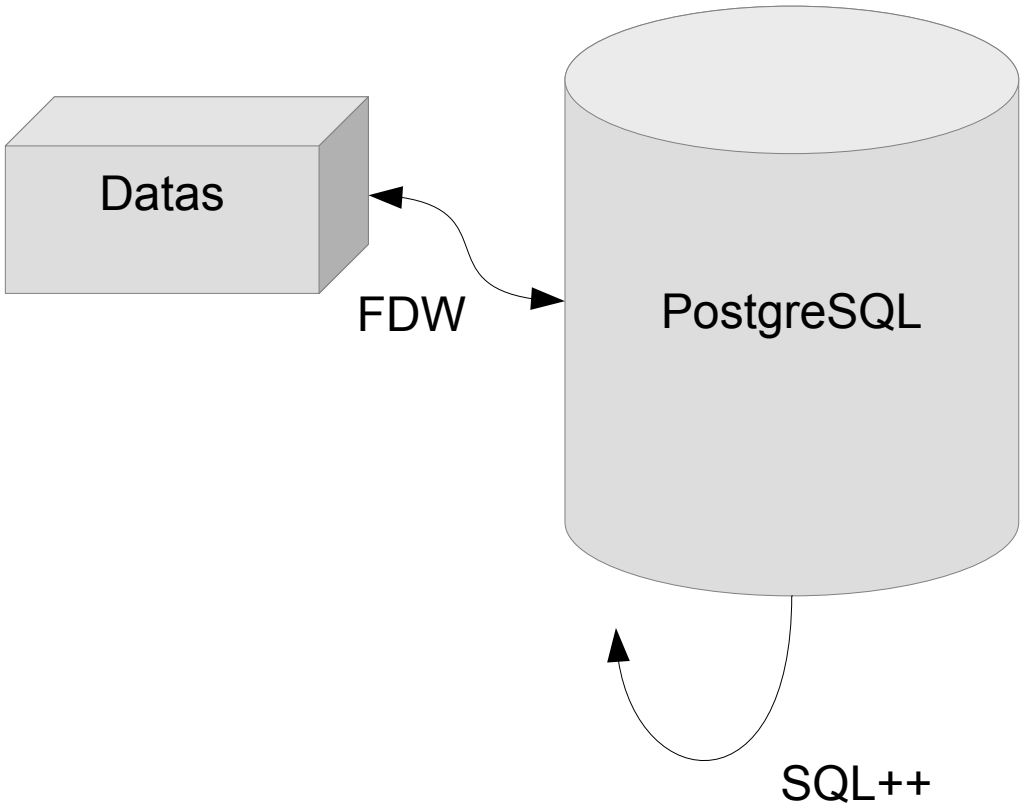
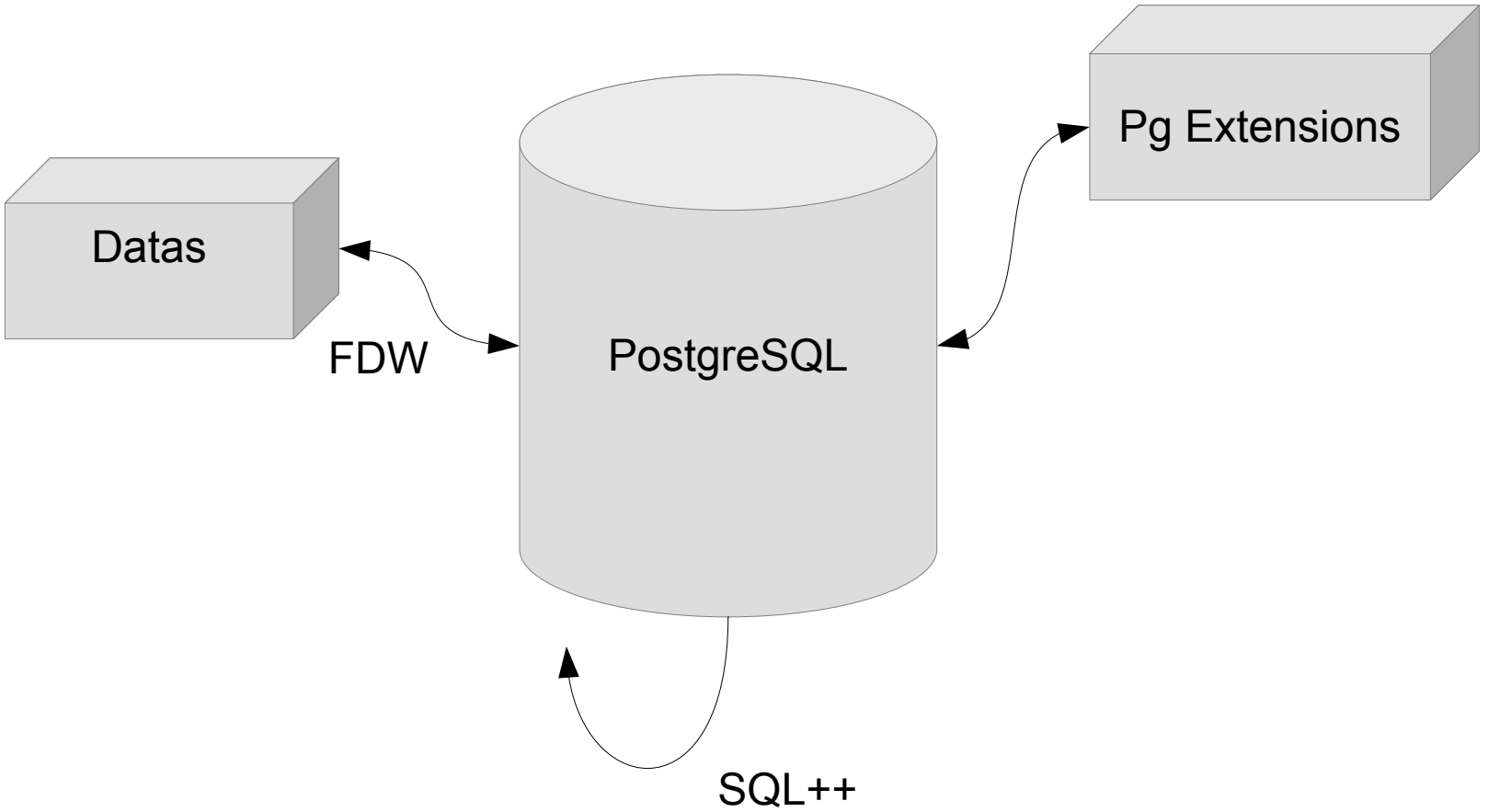
GeoDataScience

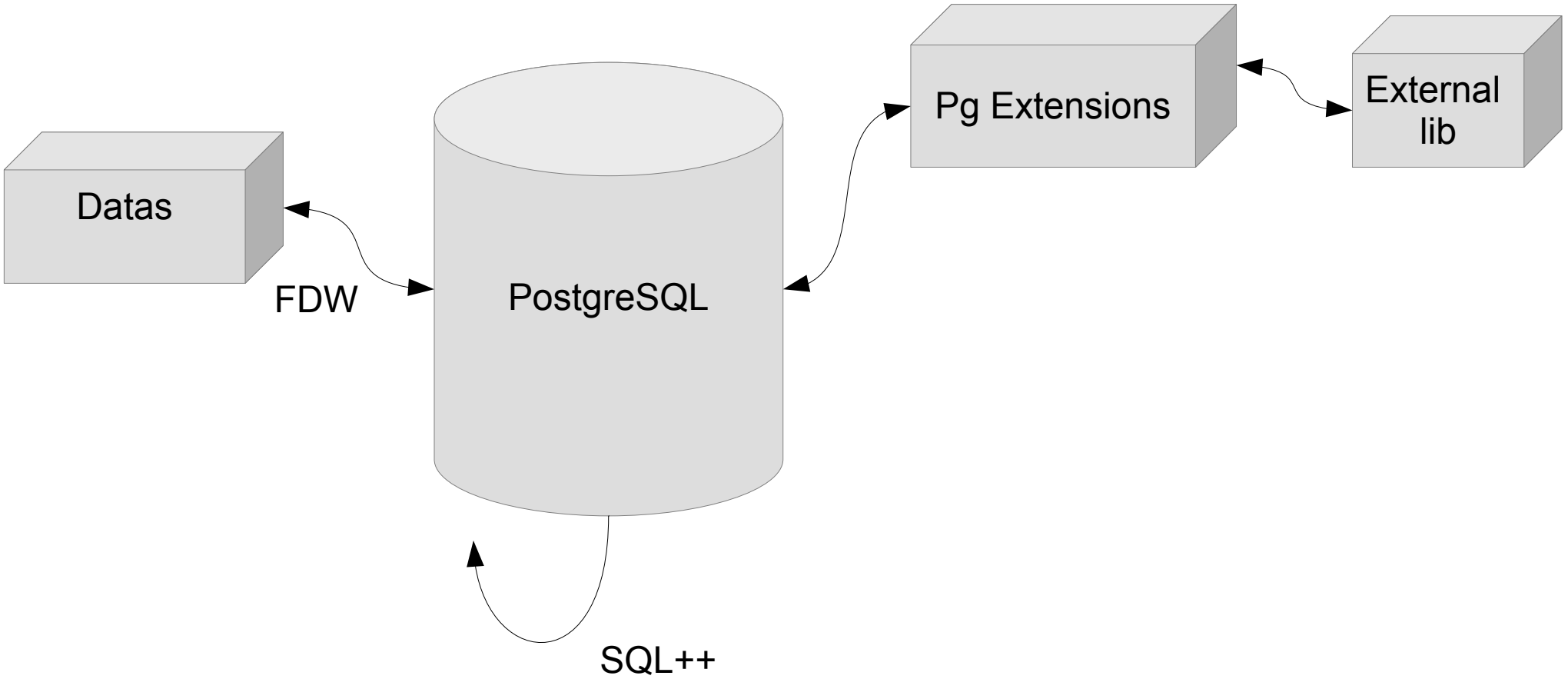
PgDay Fr 2017 – Toulouse

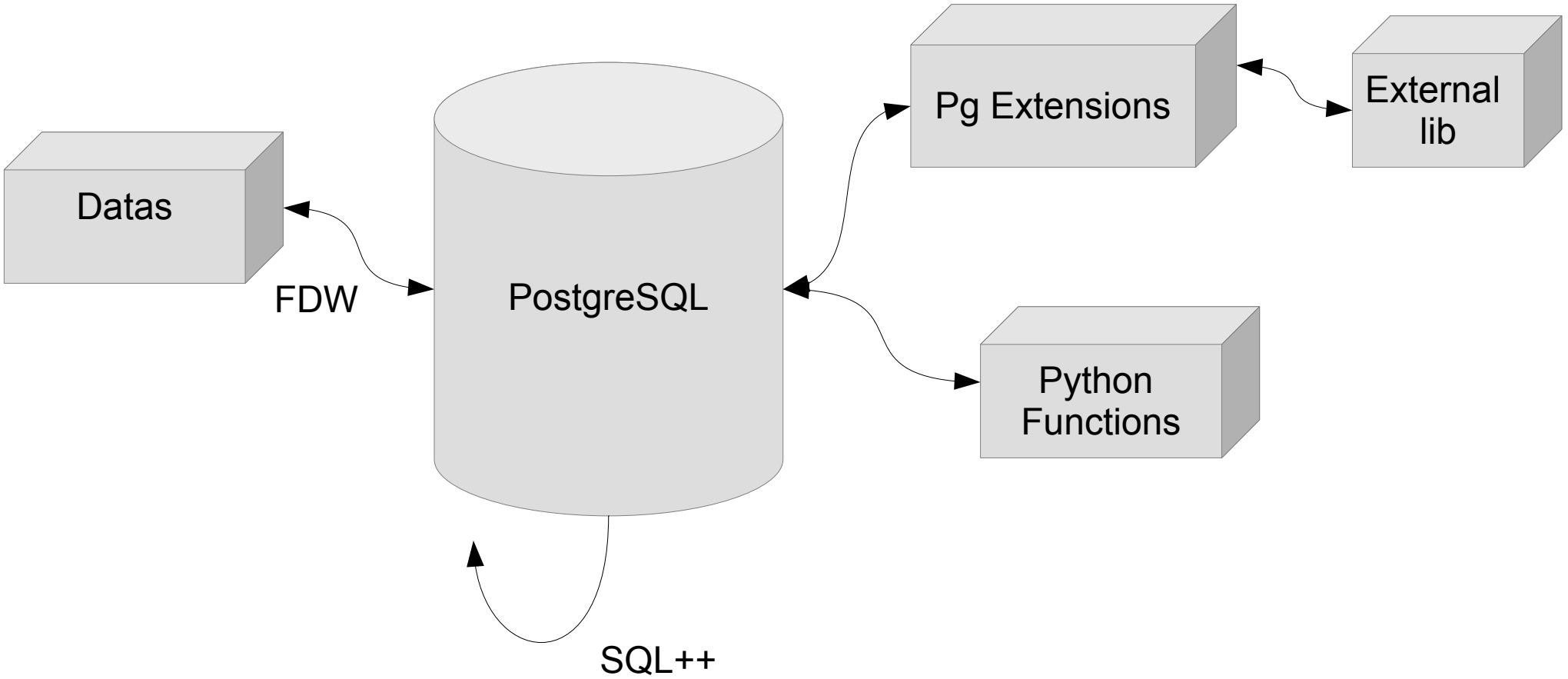
PostgreSQL is not a database
But a framework...


PostgreSQL
SQL++


Datas
FDW
PostgreSQL
SQL++


Datas
FDW
PostgreSQL
Pg Extensions
SQL++


Datas
FDW
PostgreSQL
Pg Extensions
External
lib
SQL++


Datas
FDW
PostgreSQL
Pg Extensions
External lib
Python Functions
SQL++

PostGIS 2.3.x

http://download.osgeo.org/postgis/source/postgis-2.3.2.tar.gz

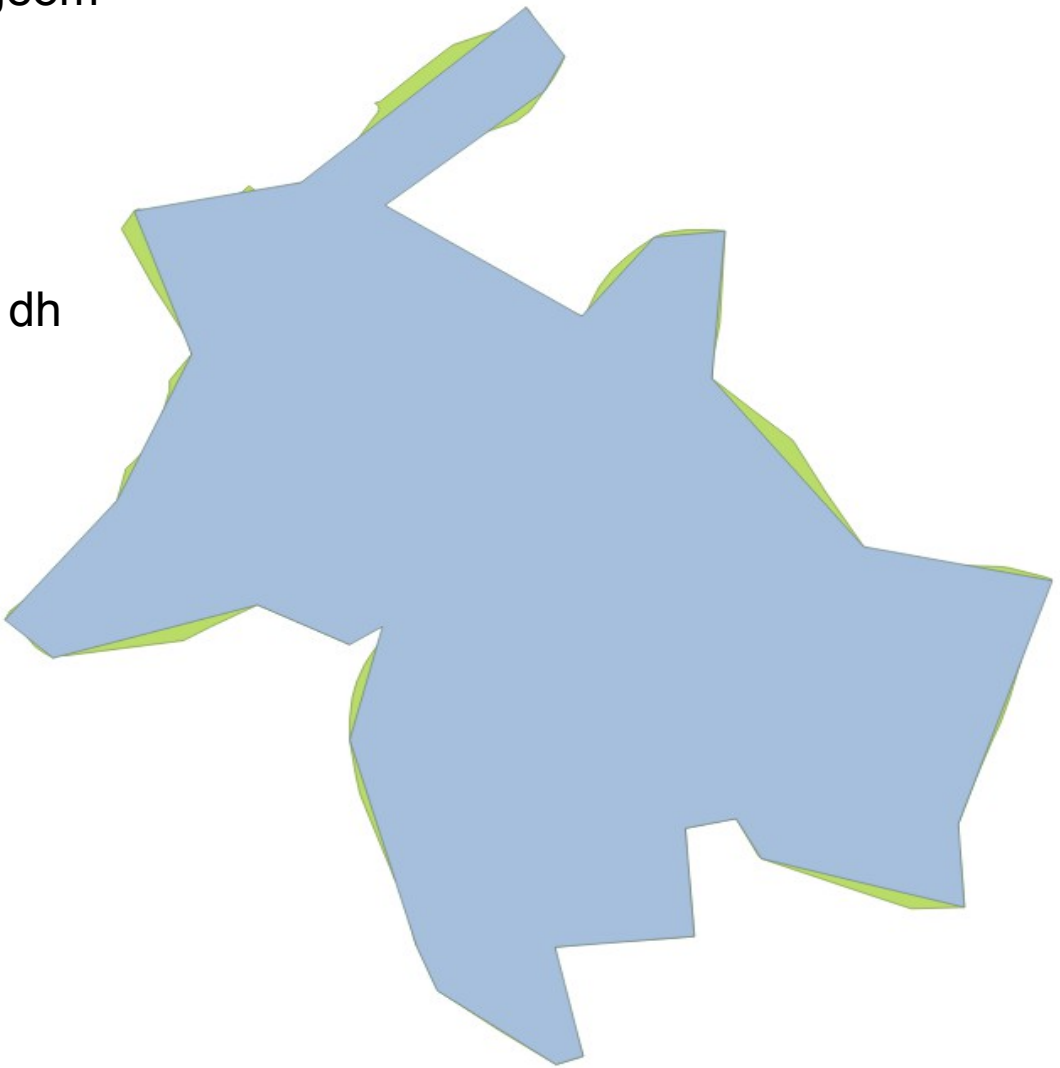
" Everything is related to everything else,
but near things are more related than distant things. "

W. Tobler

```
CREATE EXTENSION fuzzystrmatch;



SELECT levenshtein ('same', 'same');    - - and not different
0



SELECT levenshtein ('gdal', 'pdal');
1



SELECT levenshtein ('postgis', 'oracle spatial');
12
```

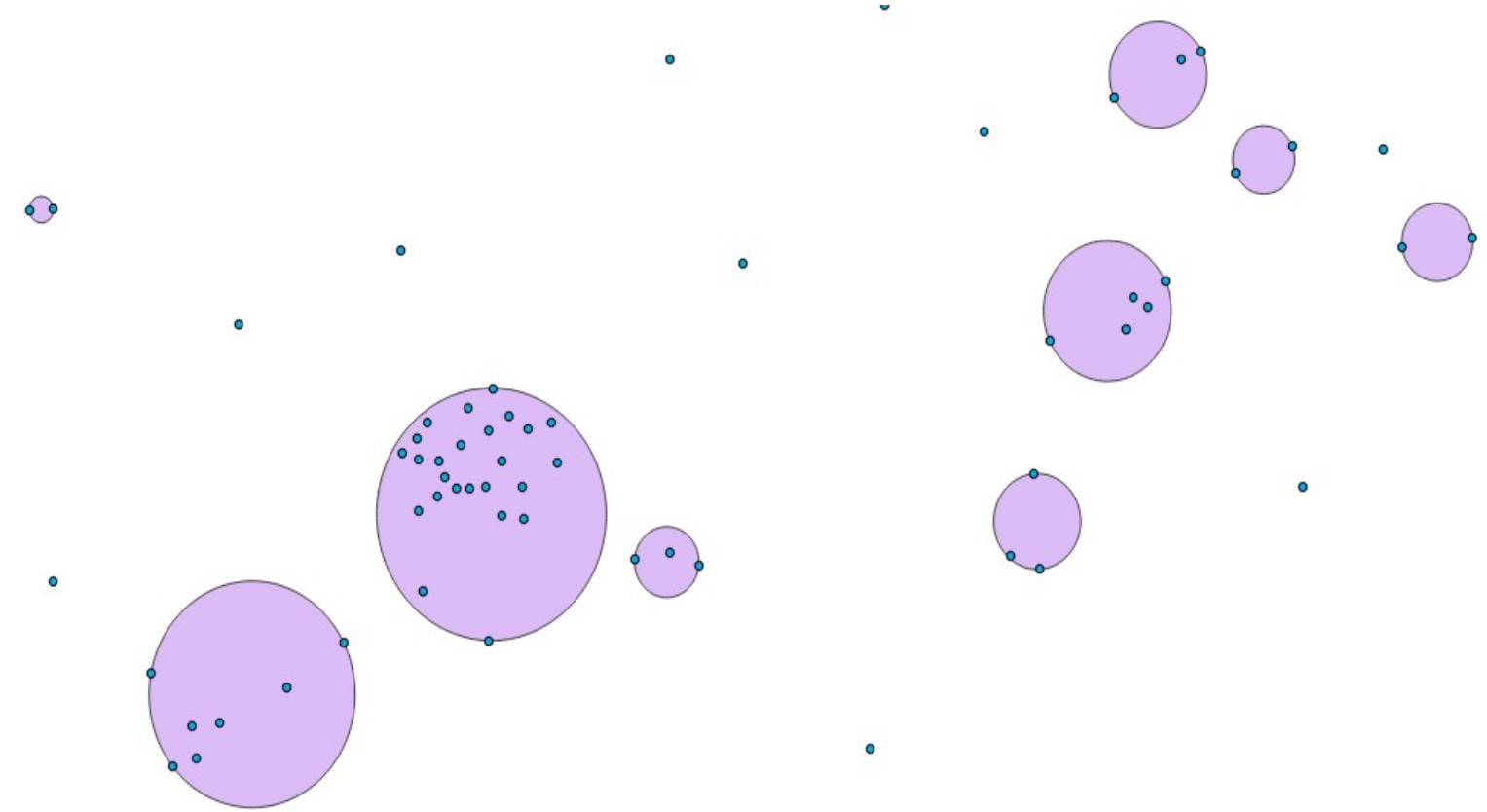
ST_HausdorffDistance

```
WITH a AS (
  SELECT id, ST_Simplify(geom, 5000) AS geom
  FROM own.commune
)

SELECT a.id, b.id,
ST_HausdorffDistance(a.geom, b.geom) AS dh
FROM a, own.commune b
WHERE nom_com = 'Lyon'
ORDER BY dh ASC
LIMIT 5;
```



```
 id  |  id  |        dh
------+------+------------------
 1347 | 1347 | 185.139093997864
 1072 | 1347 | 6681.60493070321
 2461 | 1347 | 6817.89817025694
 2824 | 1347 | 7149.21791806655
  344 | 1347 | 7929.70883765602
```

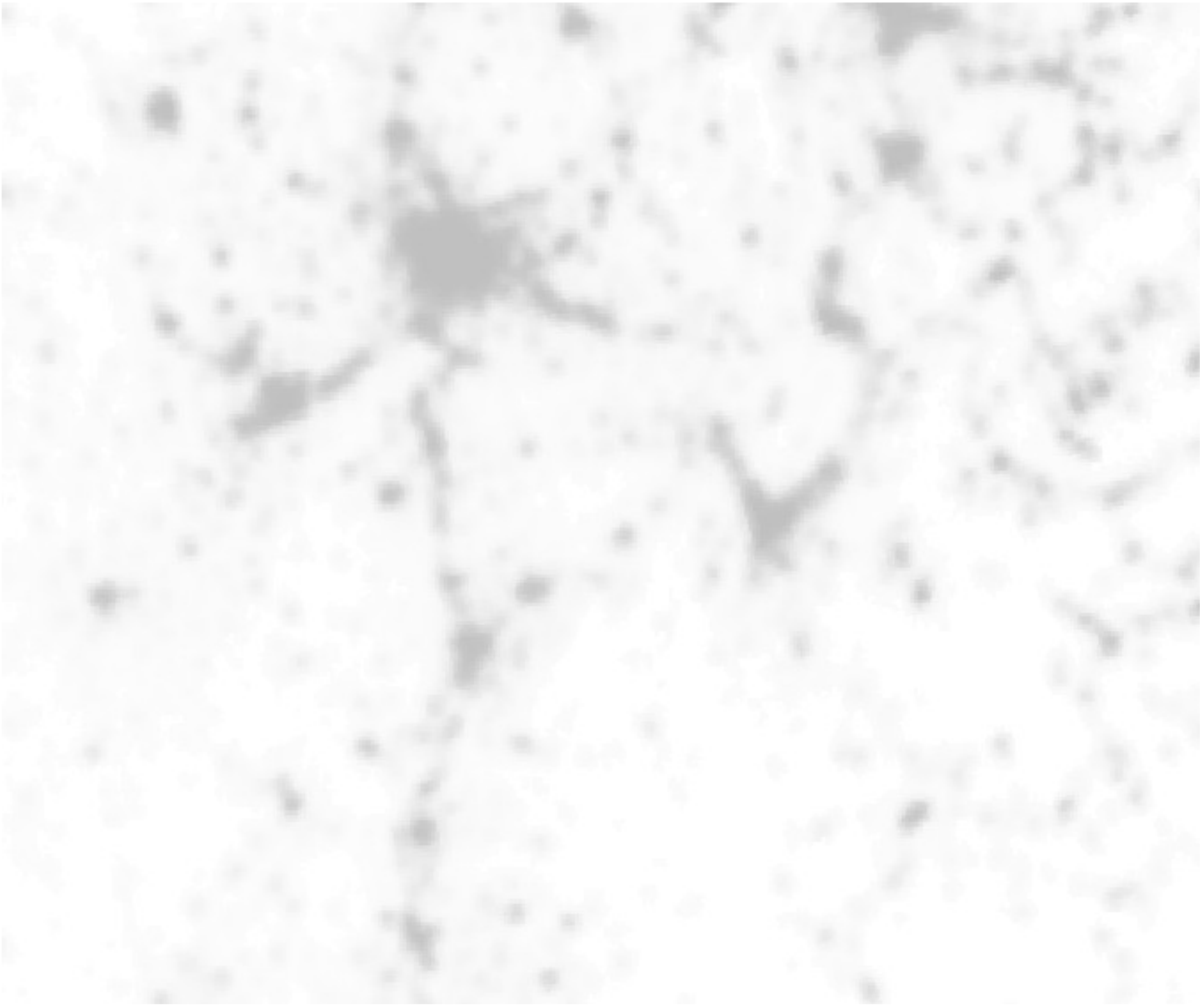
ST_ClusterWithin



```
WITH

 c AS (
   SELECT unnest(ST_ClusterWithin(ST_Centroid(geom), 12000)) AS geom
   FROM own.commune
   WHERE population > 10000
 )

 SELECT row_number() OVER() AS id, ST_MinimumBoundingCircle(geom) AS geom
 FROM c
 WHERE ST_NumGeometries(geom) > 1
```

But, could we get a bit deeper in our (spatial) analysis ?

Light Pollution @Night

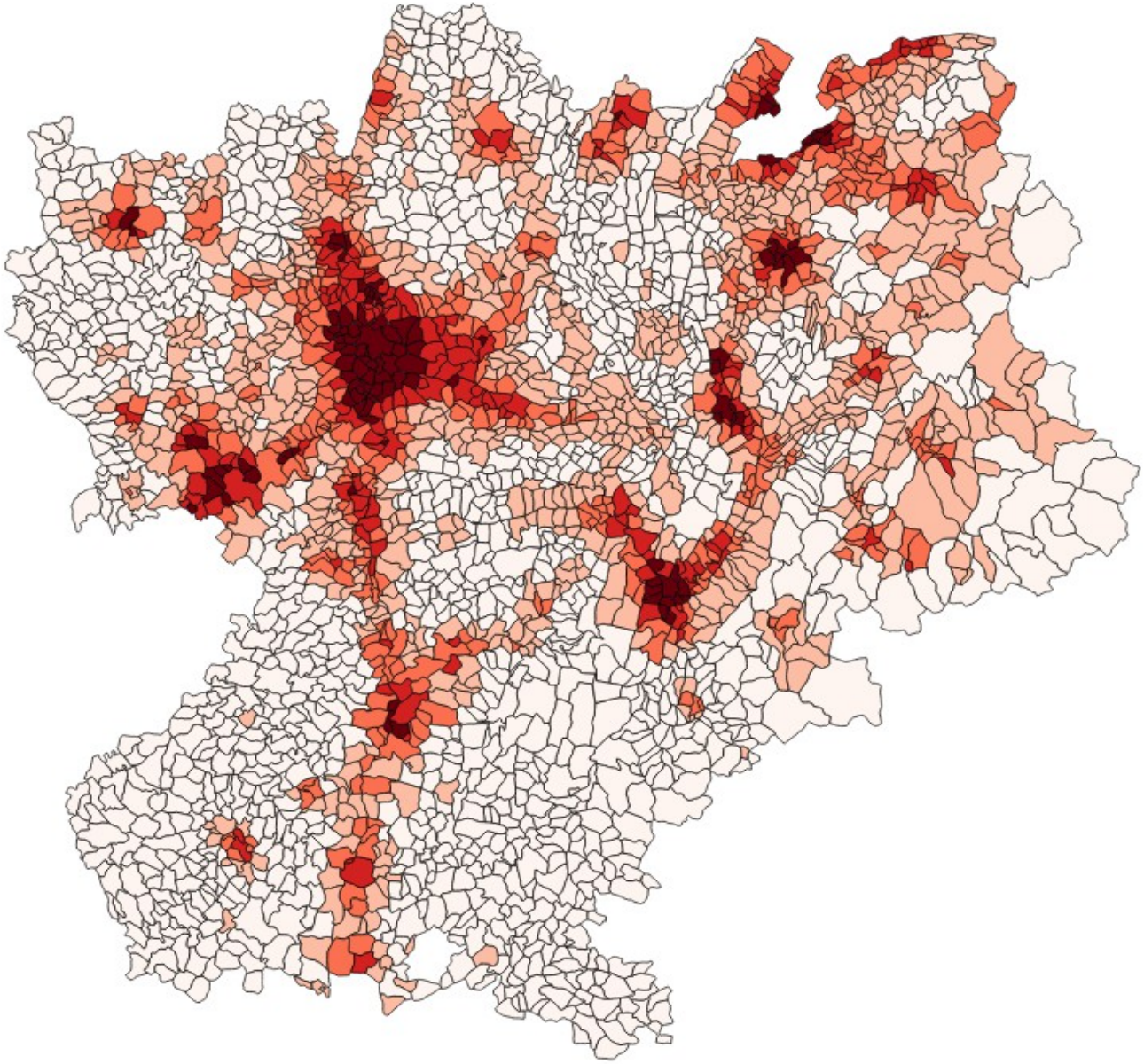


Open Data from : http://geodata.grid.unep.ch - 2003 Raster

Raster (light pollution) / Vector (area) Intersection

```
WITH ln AS
(
   SELECT id, avg(px) AS light

   FROM
   (
      SELECT  id,  ST_Value(rast, ST_SetSrid((ST_Dumppoints(pts)).geom, 2154)) AS px
      FROM (
                  SELECT id, geom AS pts  FROM own.commune
               ) AS t , r
      WHERE ST_Intersects(rast, pts)
    ) AS tt

   GROUP BY id
)

UPDATE own.commune c SET light = ln.light_pollution FROM ln WHERE c.id = ln.id
```

Light pollution by area



Road density by area

```
ALTER TABLE own.commune ADD COLUMN road_density_2016 numeric;


WITH rd AS (

	SELECT c.id,
	   (SUM(ST_Length(  ST_Intersection(c.geom, r.geom)))   /    ST_Area(c.geom)) AS road_density
	FROM own.commune c, osm.roads_2016 r
	WHERE ST_Intersects(c.geom, r.geom)
	GROUP BY c.id

)

UPDATE own.commune c SET road_density_2016 = rd.road_density FROM rd WHERE c.id = rd.id
```

Table 9-50. Aggregate Functions for Statistics

Function	Argument Type	Return Type	Description
corr(*Y*, *X*)	double precision	double precision	correlation coefficient
covar_pop(*Y*, *X*)	double precision	double precision	population covariance
covar_samp(*Y*, *X*)	double precision	double precision	sample covariance
regr_avgx(*Y*, *X*)	double precision	double precision	average of the independent variable (sum(*X*)/*N*)
regr_avgy(*Y*, *X*)	double precision	double precision	average of the dependent variable (sum(*Y*)/*N*)
regr_count(*Y*, *X*)	double precision	bigint	number of input rows in which both expressions are nonnull
regr_intercept(*Y*, *X*)	double precision	double precision	y-intercept of the least-squares-fit linear equation determined by the (*X*, *Y*) pairs
regr_r2(*Y*, *X*)	double precision	double precision	square of the correlation coefficient
regr_slope(*Y*, *X*)	double precision	double precision	slope of the least-squares-fit linear equation determined by the (*X*, *Y*) pairs
regr_sxx(*Y*, *X*)	double precision	double precision	sum(*X*^2) - sum(*X*)^2/*N* ("sum of squares" of the independent variable)
regr_sxy(*Y*, *X*)	double precision	double precision	sum(*X***Y*) - sum(*X*) * sum(*Y*)/*N* ("sum of products" of independent times dependent variable)
regr_syy(*Y*, *X*)	double precision	double precision	sum(*Y*^2) - sum(*Y*)^2/*N* ("sum of squares" of the dependent variable)
stddev(*expression*)	smallint, int, bigint, real, double precision, or numeric	double precision for floating-point arguments, otherwise numeric	historical alias for stddev_samp
stddev_pop(*expression*)	smallint, int, bigint, real, double precision, or numeric	double precision for floating-point arguments, otherwise numeric	population standard deviation of the input values
stddev_samp(*expression*)	smallint, int, bigint, real, double precision, or numeric	double precision for floating-point arguments, otherwise numeric	sample standard deviation of the input values
variance(*expression*)	smallint, int, bigint, real, double precision, or numeric	double precision for floating-point arguments, otherwise numeric	historical alias for var_samp
var_pop(*expression*)	smallint, int, bigint, real, double precision, or numeric	double precision for floating-point arguments, otherwise numeric	population variance of the input values (square of the population standard deviation)
var_samp(*expression*)	smallint, int, bigint, real, double precision, or numeric	double precision for floating-point arguments, otherwise numeric	sample variance of the input values (square of the sample standard deviation)

SELECT corr (pop_density, light)::numeric(4,4) FROM own.commune;

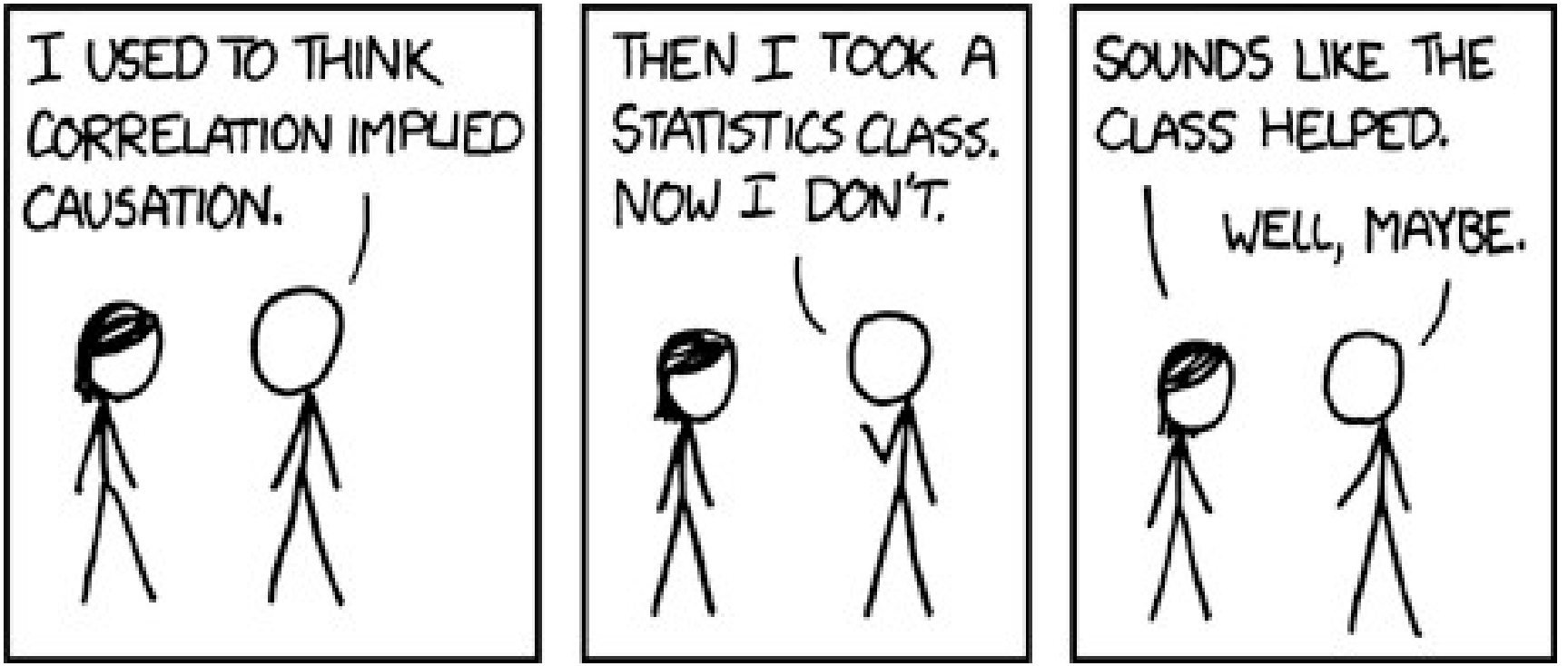
0.6533

- - OSM 08/2014
SELECT corr (road_density, light)::numeric(4,4) FROM own.commune;

0.7573

- - OSM 08/2016
SELECT corr (road_density, light)::numeric(4,4) FROM own.commune;

0.7782

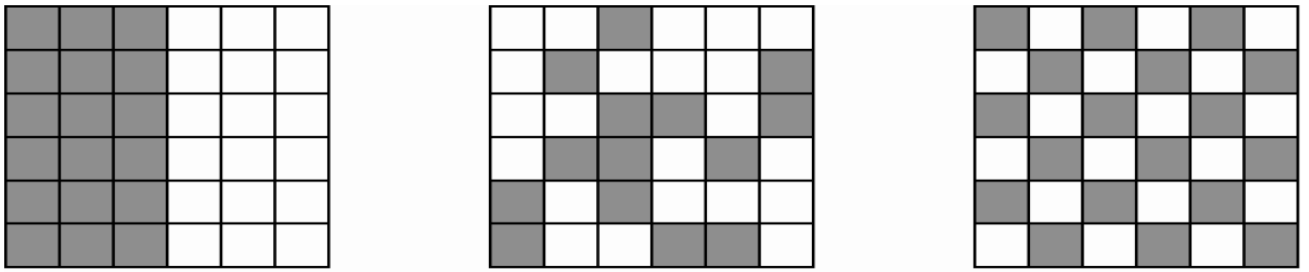


https://en.wikipedia.org/wiki/Correlation_does_not_imply_causation

" Everything is related to everything else,
but near things are more related than distant things. "

W. Tobler

Moran I - Spatial Autocorrelation Coefficient



Positive spatial autocorrelation

No spatial autocorrelation

Negative spatial autocorrelation

1 ~0 -1

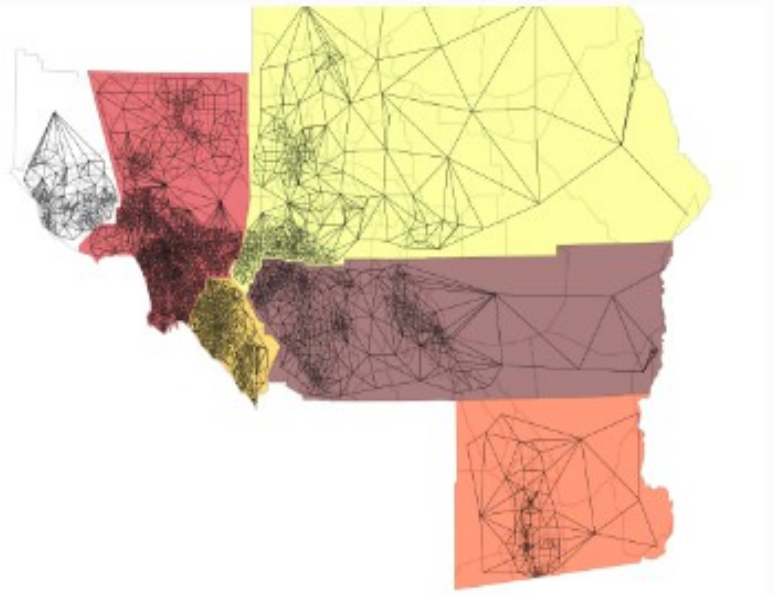
$$I = \frac{N}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(X_i - \bar{X})(X_j - \bar{X})}{\sum_i (X_i - \bar{X})^2}$$

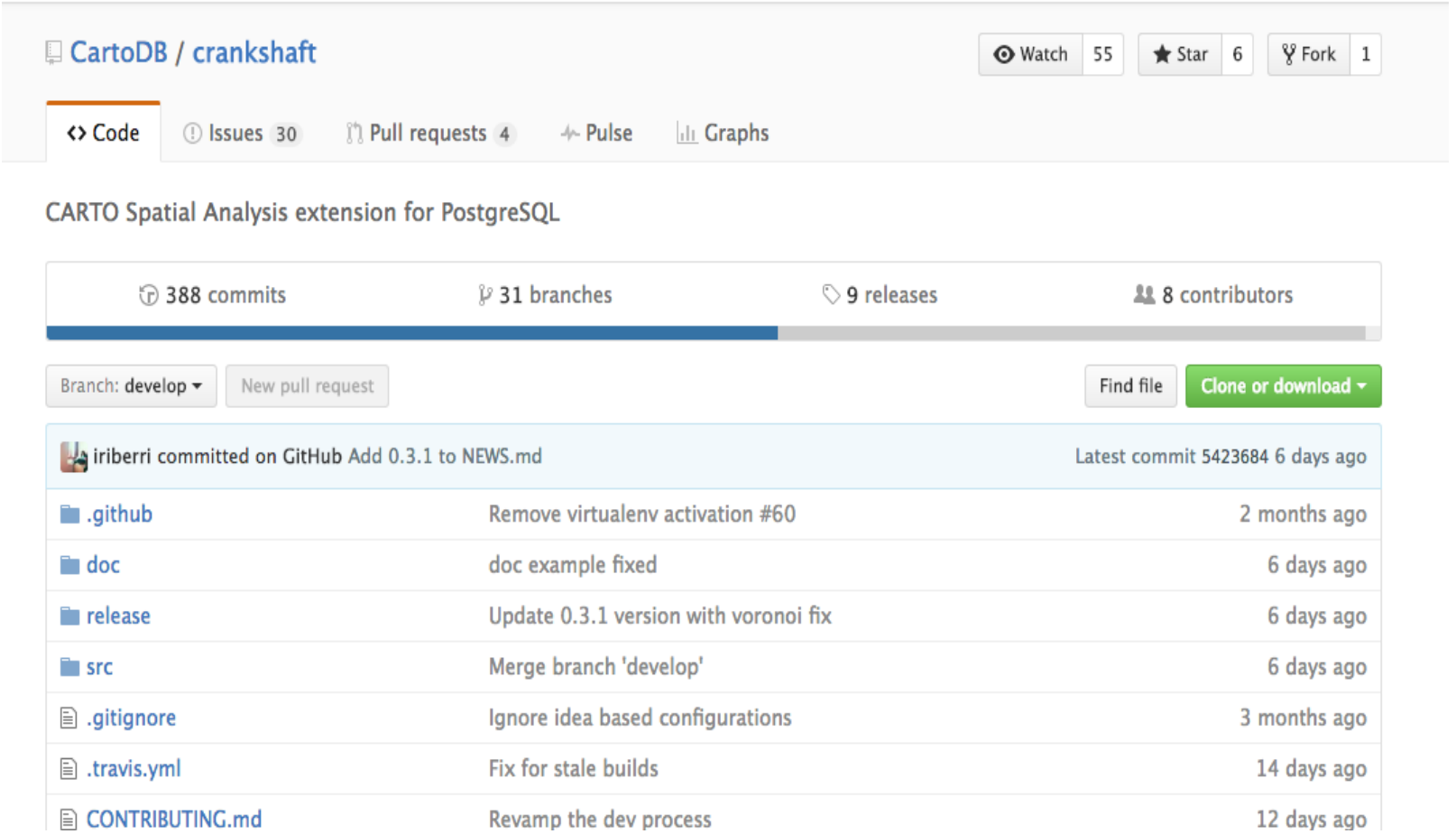
Humm, do we really need R ?

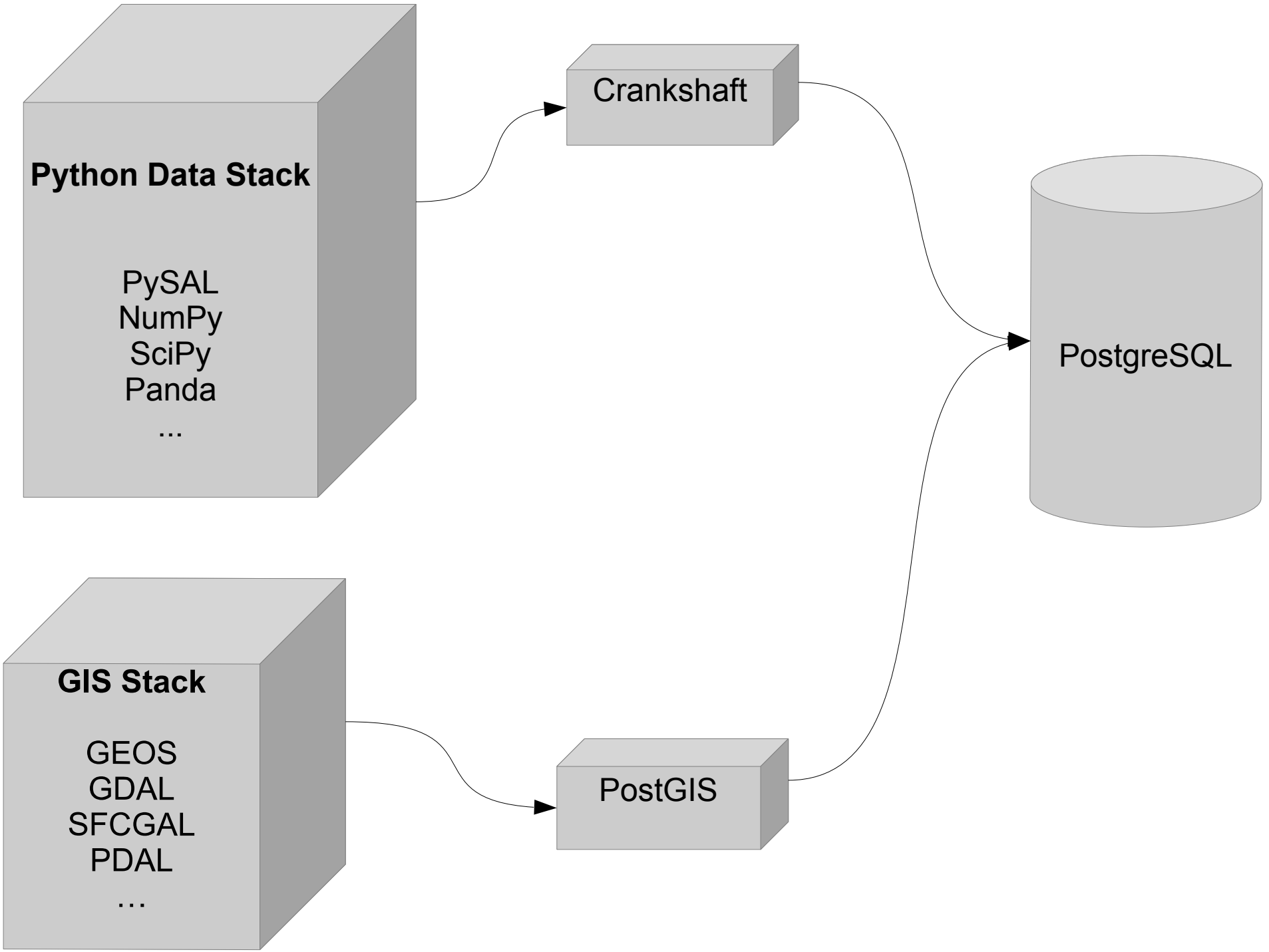
http://pysal.github.io/grid.html

PySAL: Python Spatial Analysis Library

This page collects links to examples using pysal. Click on each figure to see access the full example with code included.



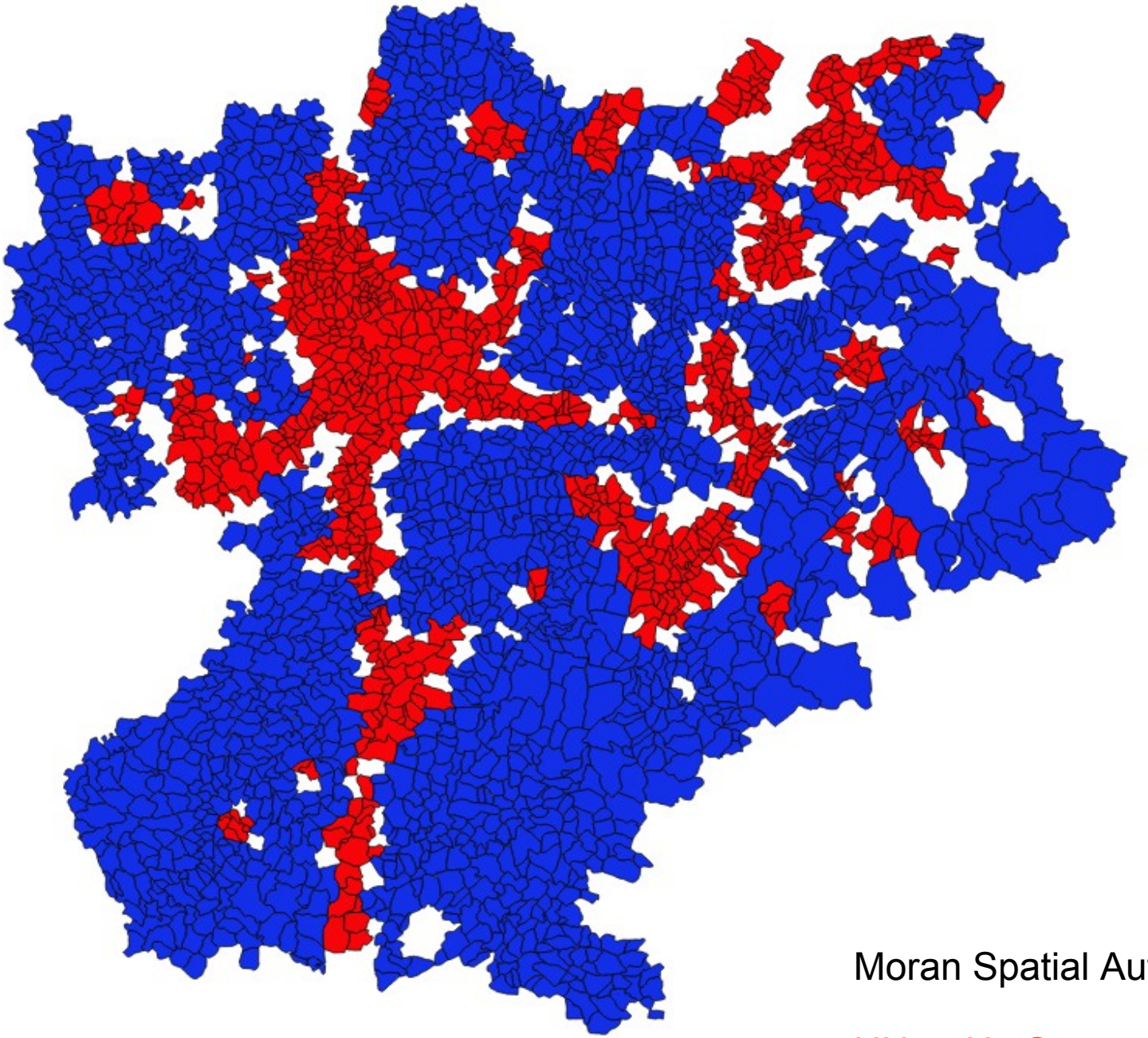

CartoDB / crankshaft
Watch 55
Star 6
Fork 1
Code
Issues 30
Pull requests 4
Pulse
Graphs
CARTO Spatial Analysis extension for PostgreSQL
388 commits
31 branches
9 releases
8 contributors
Branch: develop
New pull request
Find file
Clone or download
iriberri committed on GitHub Add 0.3.1 to NEWS.md
Latest commit 5423684 6 days ago
.github Remove virtualenv activation #60 2 months ago
doc doc example fixed 6 days ago
release Update 0.3.1 version with voronoi fix 6 days ago
src Merge branch 'develop' 6 days ago
.gitignore Ignore idea based configurations 3 months ago
.travis.yml Fix for stale builds 14 days ago
CONTRIBUTING.md Revamp the dev process 12 days ago


Python Data Stack
PySAL
NumPy
SciPy
Panda
...
Crankshaft
PostgreSQL
GIS Stack
GEOS
GDAL
SFCGAL
PDAL
...
PostGIS

```
SELECT moran::numeric(10, 4)

FROM  cdb_crankshaft.cdb_areasofinterestGlobal(

        'SELECT * FROM own.commune',  - - data table
        'light',    - - column name to check
        'knn',      - - weight : queen or knn
        5,          - - k value (for knn)

        99, 'geom', 'id'
    )
```

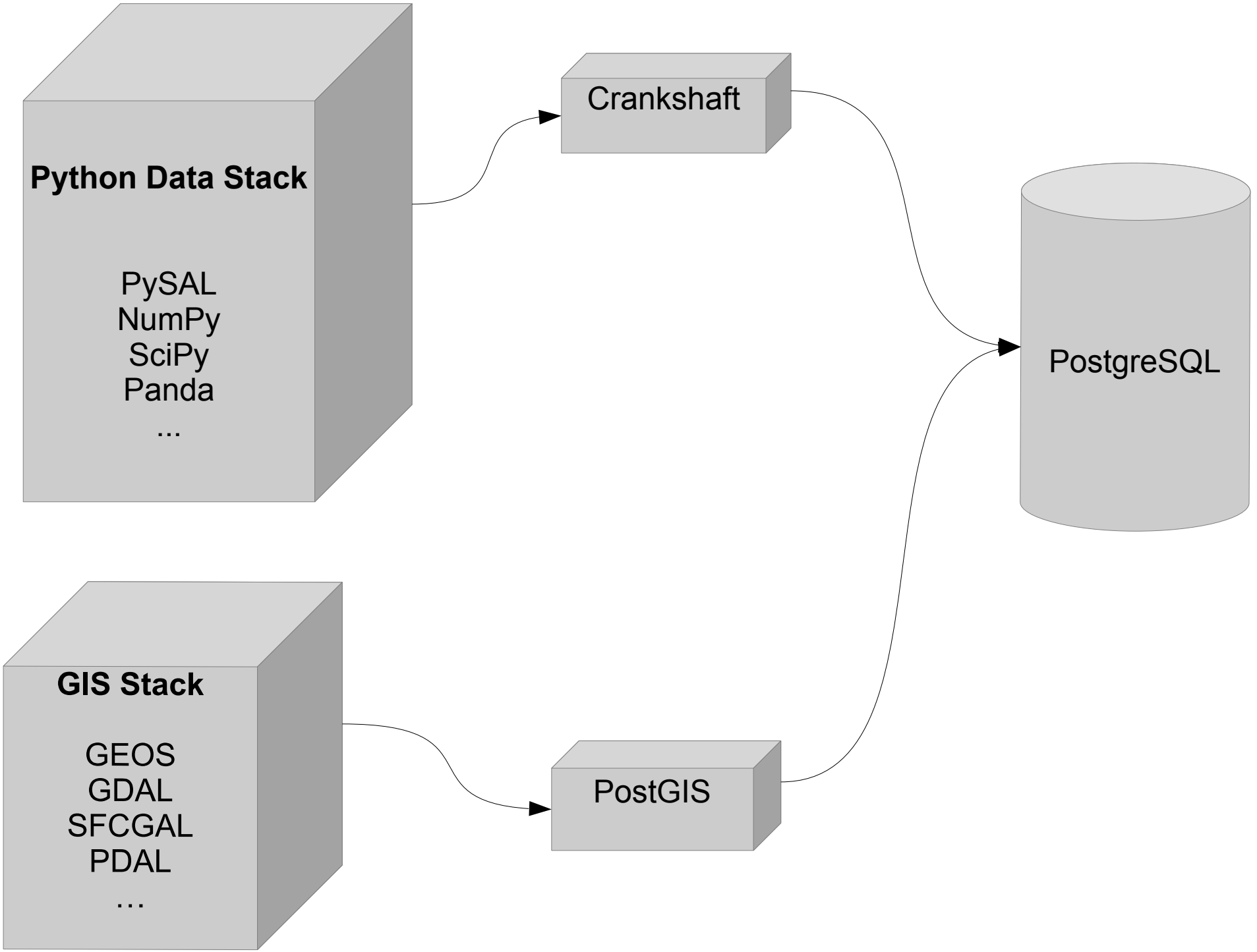
queen	0.8235
knn5	0.8201
knn20	0.6687
knn50	0.5220

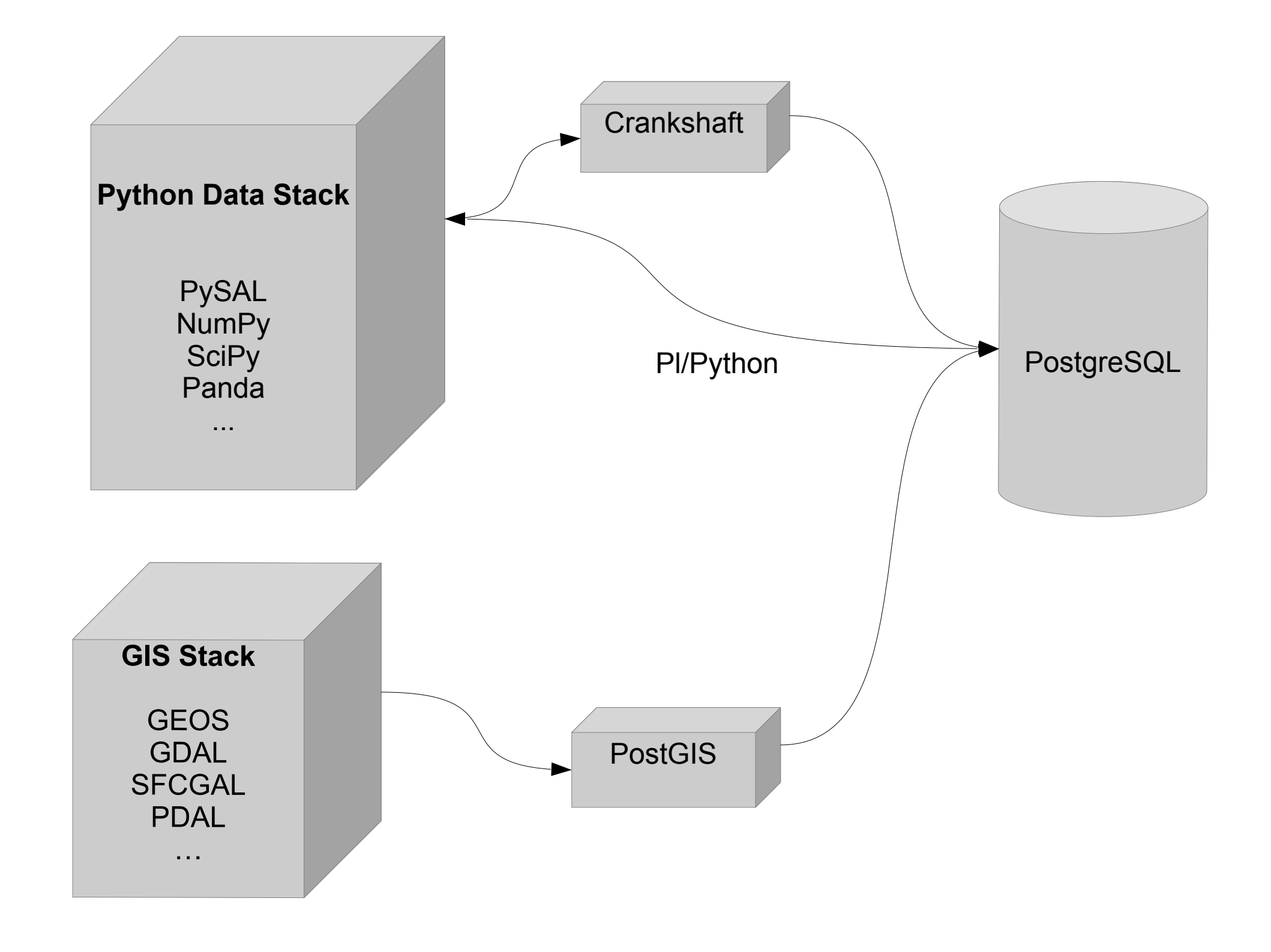
```
WITH m AS (

  SELECT aoi.*, c.id, c.nom_com , c.geom

  FROM cdb_crankshaft.cdb_areasofinterestlocal(
                    'SELECT * FROM own.commune',
                    'light',
                    'knn',
                    5,
                    99,
                    'geom',
                    'id') As aoi
  JOIN own.commune As c
  ON c.id = aoi.rowid

)

SELECT quads, geom, ow_number() OVER() AS id
FROM u
WHERE quads = 'HH' OR quads = 'LL'
```



Moran Spatial AutoCorrelation

HH -> HotSpot
LL -> LowSpot


Python Data Stack
PySAL
NumPy
SciPy
Panda
...
Crankshaft
PostgreSQL
GIS Stack
GEOS
GDAL
SFCGAL
PDAL
...
PostGIS


Python Data Stack
PySAL
NumPy
SciPy
Panda
...
Crankshaft
PostgreSQL
Pl/Python
GIS Stack
GEOS
GDAL
SFCGAL
PDAL
...
PostGIS

docs.scipy.org/doc/scipy/reference/signal.html

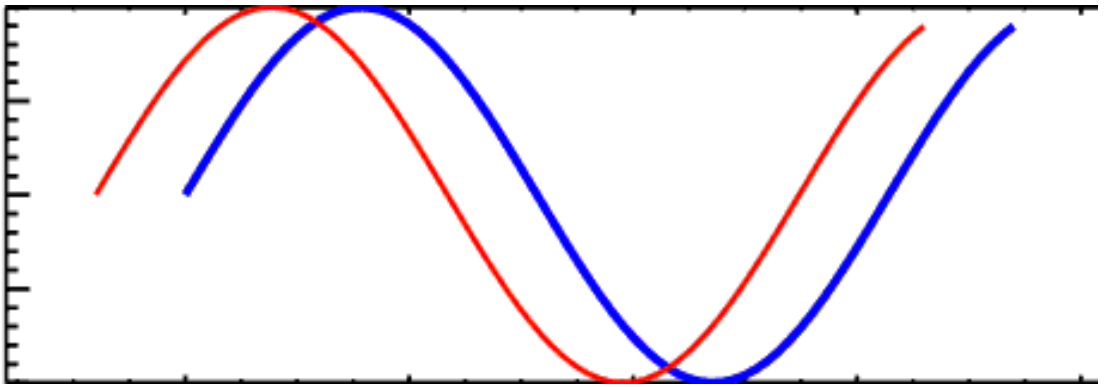
SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | SciPy v0.18.1 Reference Guide

Signal processing (scipy.signal)

Convolution

convolve(in1, in2[, mode])	Convolve two N-dimensional arrays.
correlate(in1, in2[, mode])	Cross-correlate two N-dimensional arrays.
fftconvolve(in1, in2[, mode])	Convolve two N-dimensional arrays using FFT.
convolve2d(in1, in2[, mode, boundary, fillvalue])	Convolve two 2-dimensional arrays.
correlate2d(in1, in2[, mode, boundary, ...])	Cross-correlate two 2-dimensional arrays.
sepfir2d((input, hrow, hcol) -> output)	Description:



```
CREATE OR REPLACE FUNCTION signal_correlate(a float[], b float[])
RETURNS numeric
AS $$

from scipy import signal
import numpy as np

return np.argmax(signal.correlate(a, b)) - len(a)

$$ LANGUAGE plpythonu;
```

```
[[[ 0.63112545  0.65073329  0.67818427 ...,  0.54854906  0.54070592
    0.54462749]
  [ 0.56535292  0.57319605  0.58888233 ...,  0.56031376  0.54462749
    0.54070592]
  [ 0.44629803  0.47374901  0.51688629 ...,  0.57600003  0.55247062
    0.54070592]

  ...,
  [ 0.21225882  0.26716077  0.32877645 ...,  0.38489804  0.34176078
    0.36921176]
  [ 0.23186666  0.24363138  0.26995292 ...,  0.31823137  0.29862353
    0.34960392]
  [ 0.27892548  0.25539607  0.25818822 ...,  0.26332942  0.26725098
    0.34176078]]]
```

PandaPost

This Release: PandaPost 0.2.0
Date: 2016-08-19
Status: Unstable
Abstract: Python NumPy ndarray data type for Postgres
Released By: decibel
License: The (two-clause) FreeBSD License
Resources: www ✦ git ✦ repo ✦ bugs
Special Files: LICENSE ✦ META.json ✦ Makefile ✦ PandaPost.control
Tags: python ✦ numpy ✦ ndarray

Extensions

PandaPost 0.2.0

Python NumPy ndarray data type for Postgres

WKB Raster

Read WKB rasters to Numpy arrays.

Docs

```
wkb_raster.read_wkb_raster(wkb)
```

Parameters

- wkb – file-like object. Binary raster in WKB format.

With WKB from PostGIS Raster. Use ST_AsBinary to return the WKB representation of the raster.

```
SELECT ST_AsBinary(rast) FROM rasters;
```

Wrap the binary buffer in `cStringIO.StringIO`:

```
from cStringIO import StringIO
from wkb_raster import read_wkb_raster

raster = read_wkb_raster(StringIO(buf))
raster['bands'][0]
```

https://github.com/nathancahill/wkb-raster

WKB Raster

Read WKB rasters to Numpy arrays.

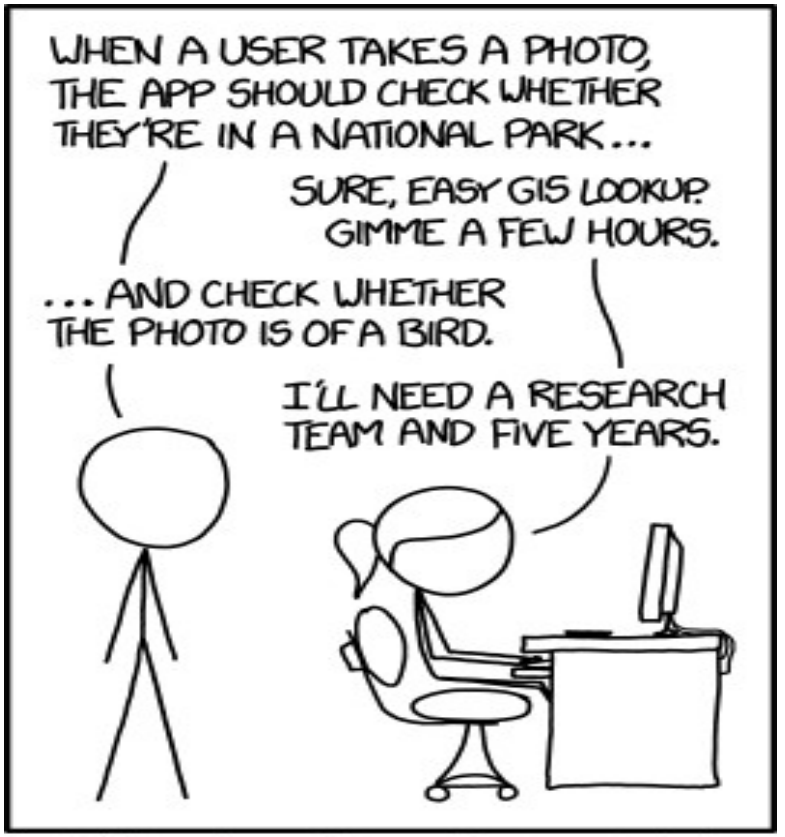
Docs

```
wkb_raster.read_wkb_raster(wkb)
```

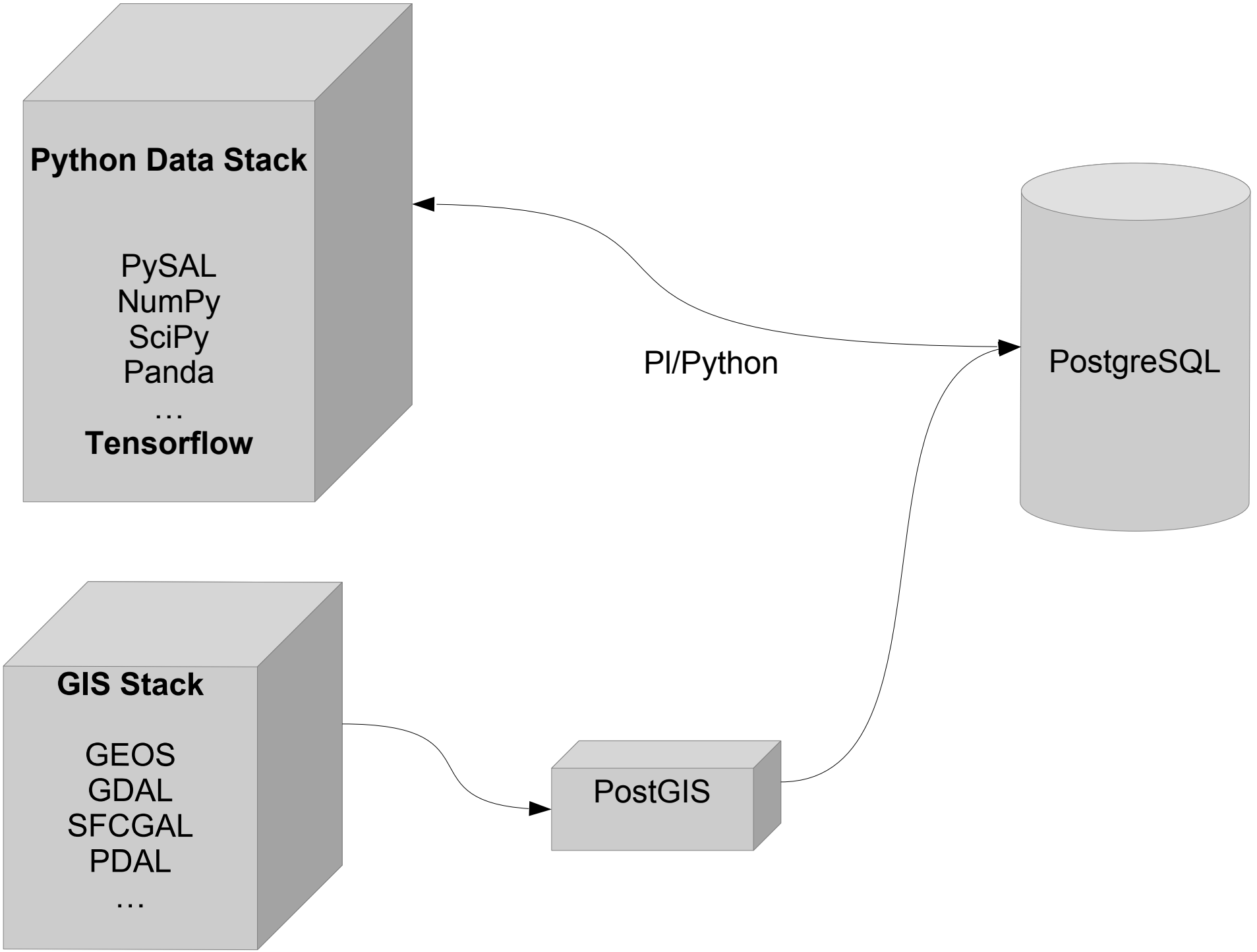
Parameters

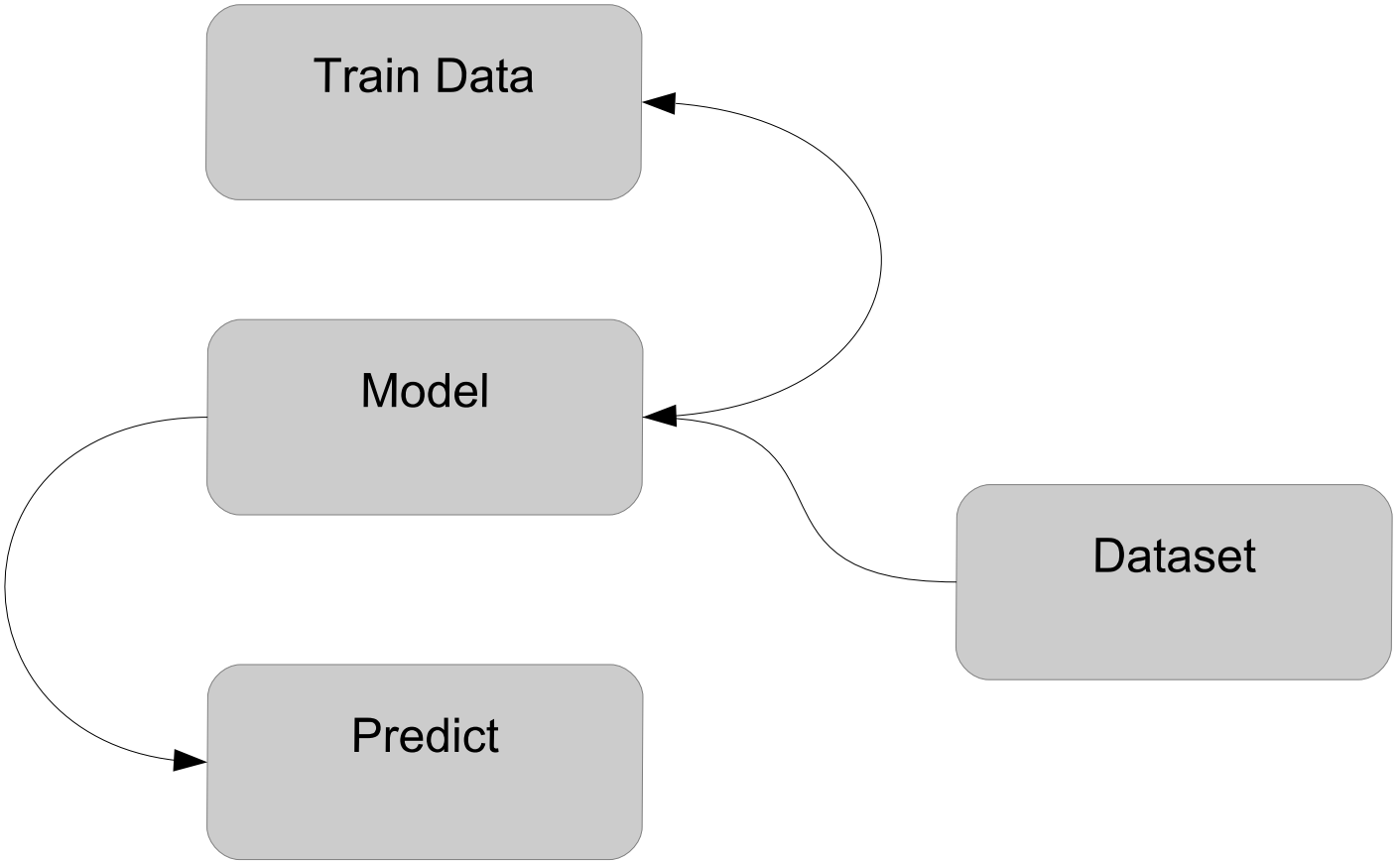
- wkb – file-like object. Binary raster in WKB format.

https://github.com/nathancahill/wkb-raster



IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.


Python Data Stack
PySAL
NumPy
SciPy
Panda
…
Tensorflow
Pl/Python
PostgreSQL
GIS Stack
GEOS
GDAL
SFCGAL
PDAL
…
PostGIS


Train Data
Model
Dataset
Predict

DeepOSM build passing

Classify roads and features in satellite imagery, by training neural networks with OpenStreetMap (OSM) data.

DeepOSM can:

- Download a chunk of satellite imagery
- Download OSM data that shows roads/features for that area
- Generate training and evaluation data
- Display predictions of mis-registered roads in OSM data, or display raw predictions of ON/OFF

Running the code is as easy as install Docker, make dev, and run a script.

Contributions are welcome. Open an issue if you want to discuss something to do, or email me.

Default Data/Accuracy

By default, DeepOSM will analyze about 200 sq. km of area in Delaware. DeepOSM will

- predict if the center 9px of a 64px tile contains road.
- use the infrared (IR) band and RGB bands.
- be 75-80% accurate overall, training only for a minute or so.
- use a single fully-connected relu layer in TensorFlow.
- render, as JPEGs, "false positive" predictions in the OSM data - i.e. where OSM lists a road, but DeepOSM thinks there isn't one.

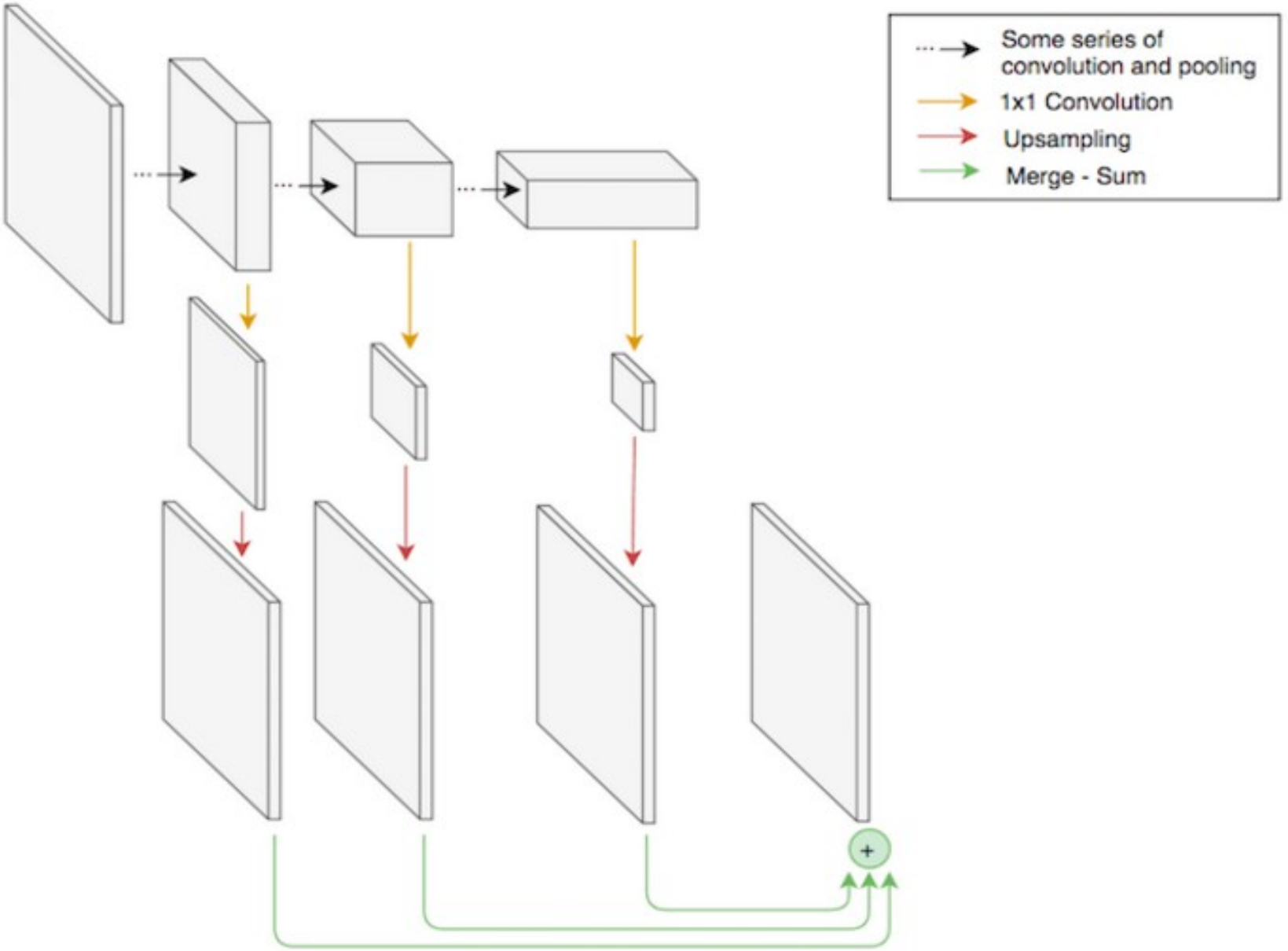
azavea

SOFTWARE DEVELOPMENT

Deep Learning for Semantic Segmentation of Aerial Imagery

By Rob Emanuele on May 30th, 2017

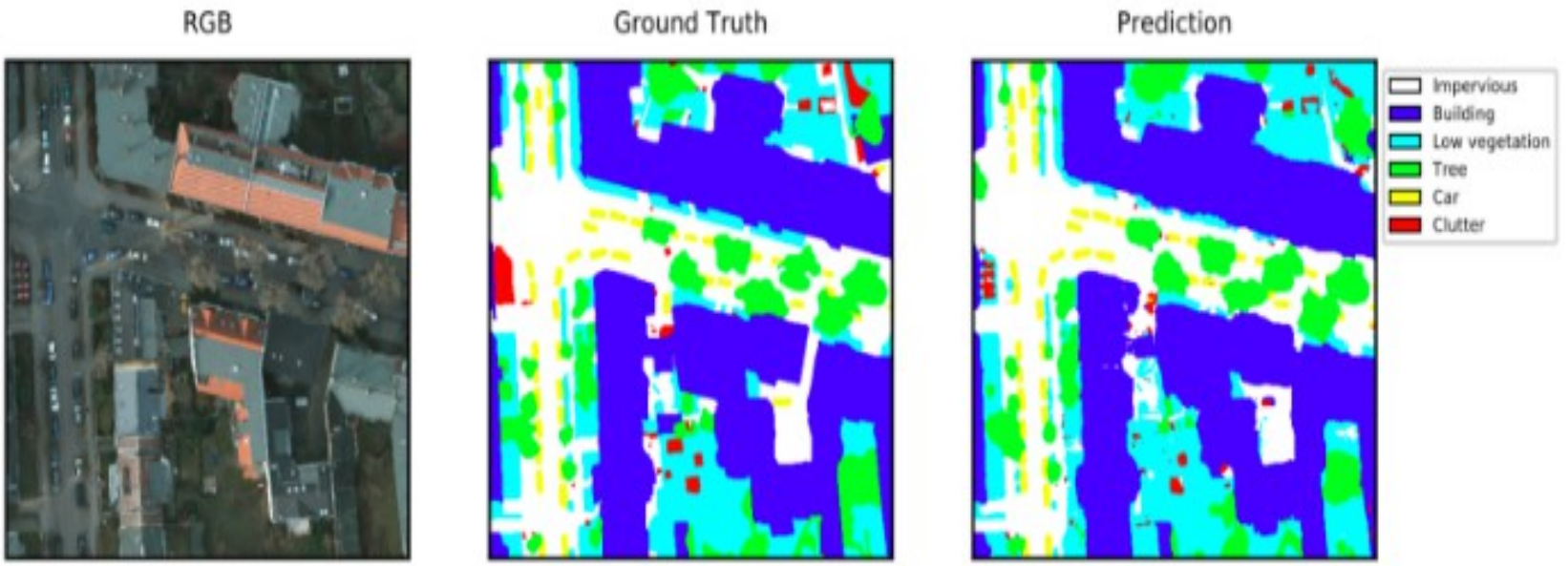
https://www.azavea.com/blog/2017/05/30/deep-learning-on-aerial-imagery/



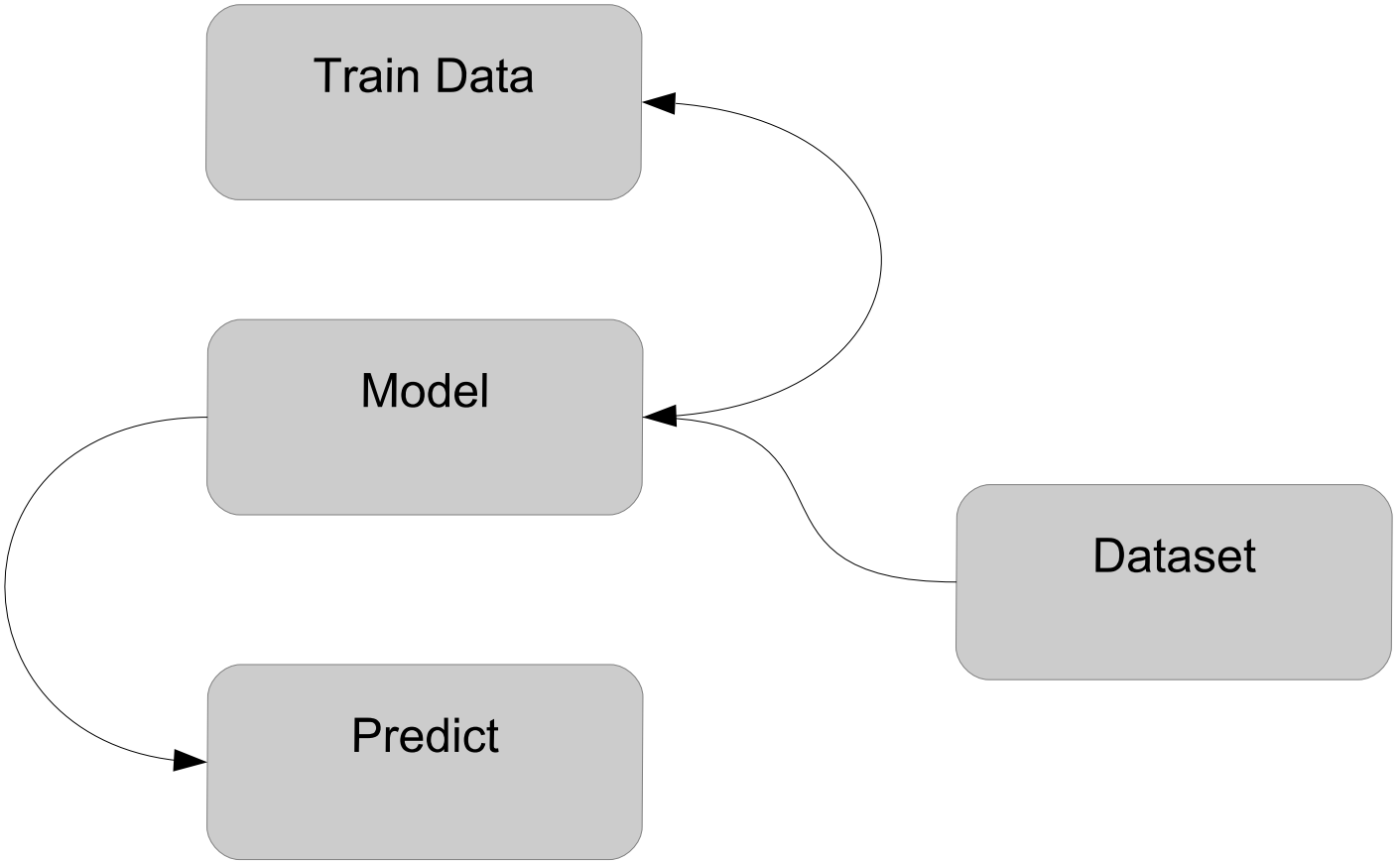
Fully convolutional neural network architecture for semantic segmentation

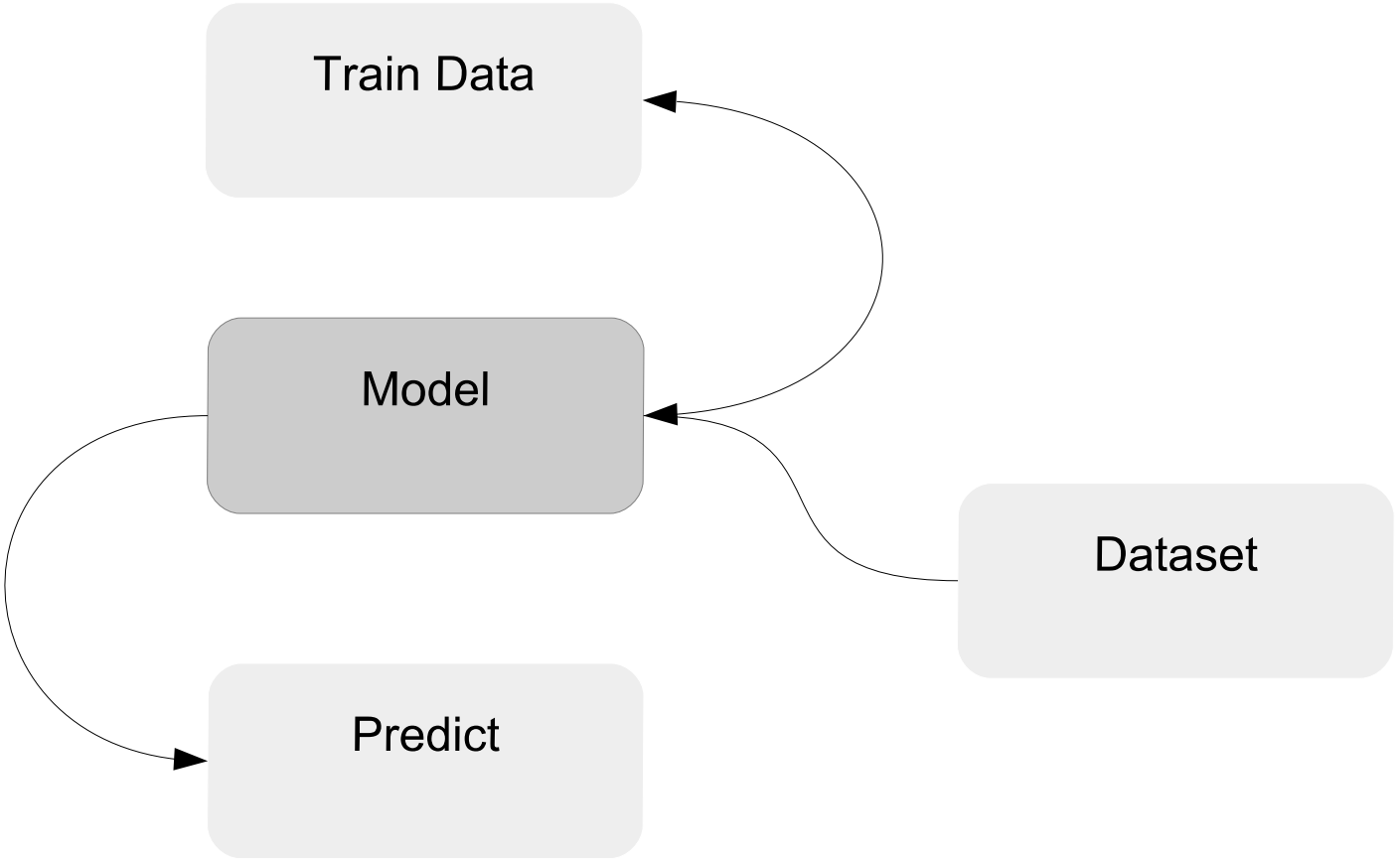
```
# The number of output labels
nb_labels = 6

# The dimensions of the input images
nb_rows = 256
nb_cols = 256

# A ResNet model with weights from training on ImageNet. This will
# be adapted via graph surgery into an FCN.
base_model = ResNet50(
    include_top=False, weights='imagenet', input_tensor=input_tensor)

# Get final 32x32, 16x16, and 8x8 layers in the original
# ResNet by that layers's name.
x32 = base_model.get_layer('final_32').output
x16 = base_model.get_layer('final_16').output
x8 = base_model.get_layer('final_x8').output

# Compress each skip connection so it has nb_labels channels.
c32 = Convolution2D(nb_labels, (1, 1))(x32)
c16 = Convolution2D(nb_labels, (1, 1))(x16)
c8 = Convolution2D(nb_labels, (1, 1))(x8)

```

```
23.
24. # Resize each compressed skip connection using bilinear interpolation.
25. # This operation isn't built into Keras, so we use a LambdaLayer
26. # which allows calling a Tensorflow operation.
27. def resize_bilinear(images):
28.     return tf.image.resize_bilinear(images, [nb_rows, nb_cols])
29.
30. r32 = Lambda(resize_bilinear)(c32)
31. r16 = Lambda(resize_bilinear)(c16)
32. r8 = Lambda(resize_bilinear)(c8)
33.
34. # Merge the three layers together using summation.
35. m = Add()([r32, r16, r8])
36.
37. # Add softmax layer to get probabilities as output. We need to reshape
38. # and then un-reshape because Keras expects input to softmax to
39. # be 2D.
40. x = Reshape((nb_rows * nb_cols, nb_labels))(m)
41. x = Activation('softmax')(x)
42. x = Reshape((nb_rows, nb_cols, nb_labels))(x)
43.
44. fcn_model = Model(input=input_tensor, output=x)
```

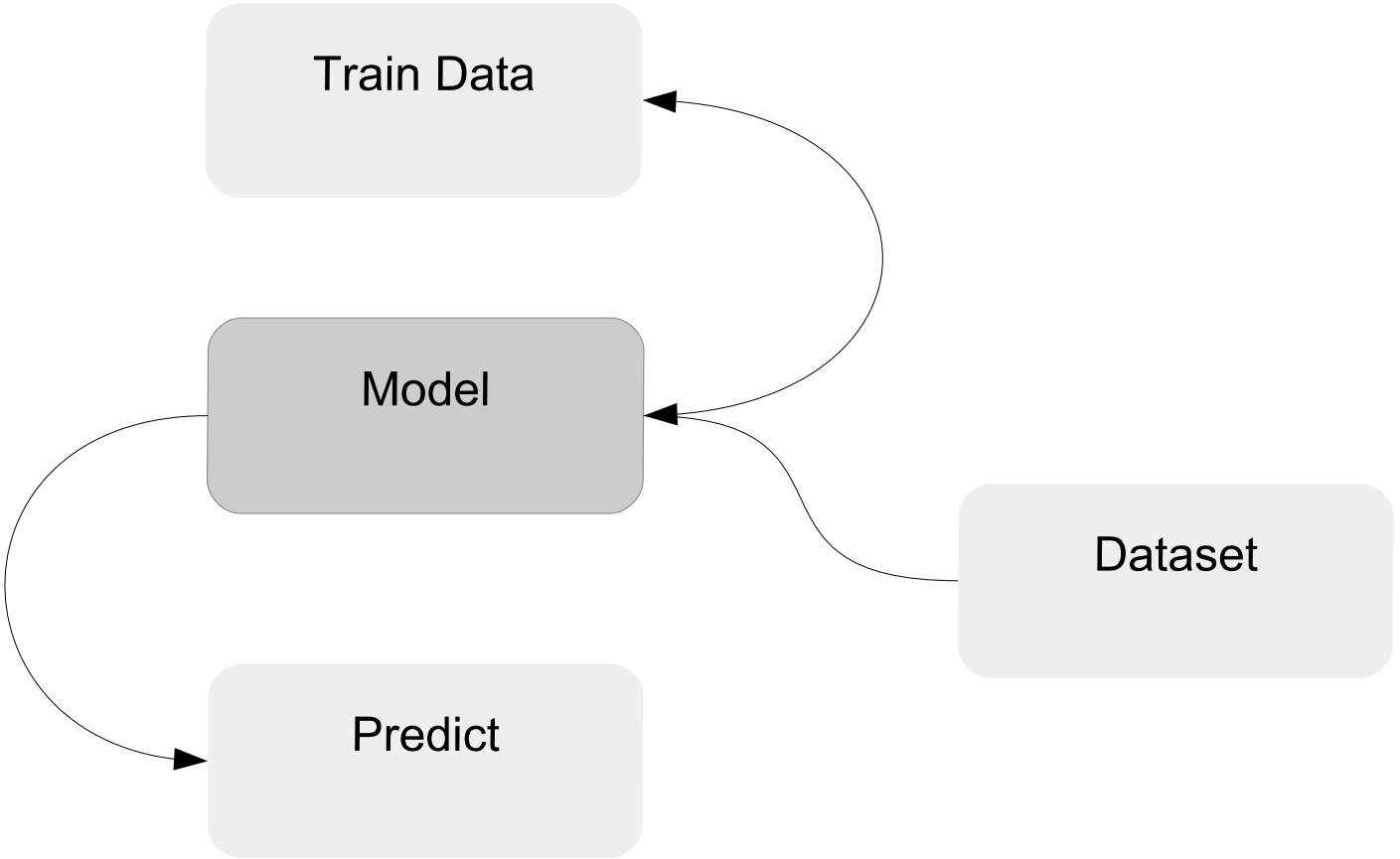


	Overall	Impervious	Building	Low Vegetation	Tree	Car	Clutter
Validation	85.8	89.1	91.8	82.0	83.3	93.7	63.2
Test	89.2	91.4	96.1	86.1	86.6	93.3	46.8


Train Data
Model
Dataset
Predict



How to keep the model persitent ?



How to keep the model persitent ?

Skills to fully play with

SQL++

(Open) Data

PostGIS
ToolBox

Statistical
skills

PG Extension

Python

NVidia GPU
ToolBox

#Conclusions

PostgreSQL behaves like an extensible
and integrated Framework

(modern) SQL and Python acting as glue languages

Possible Bridge beetween
GIS and Python DataScience communities

PG native NumPy type

PG ML Model Persistent

ML Efficient High Resolution Raster Handling

Real NVIDIA OSS Alternate Stack ?

Thanks !

 o_courtin