# Advanced spatial analysis with PostgreSQL, PostGIS and Python

@o_courtin

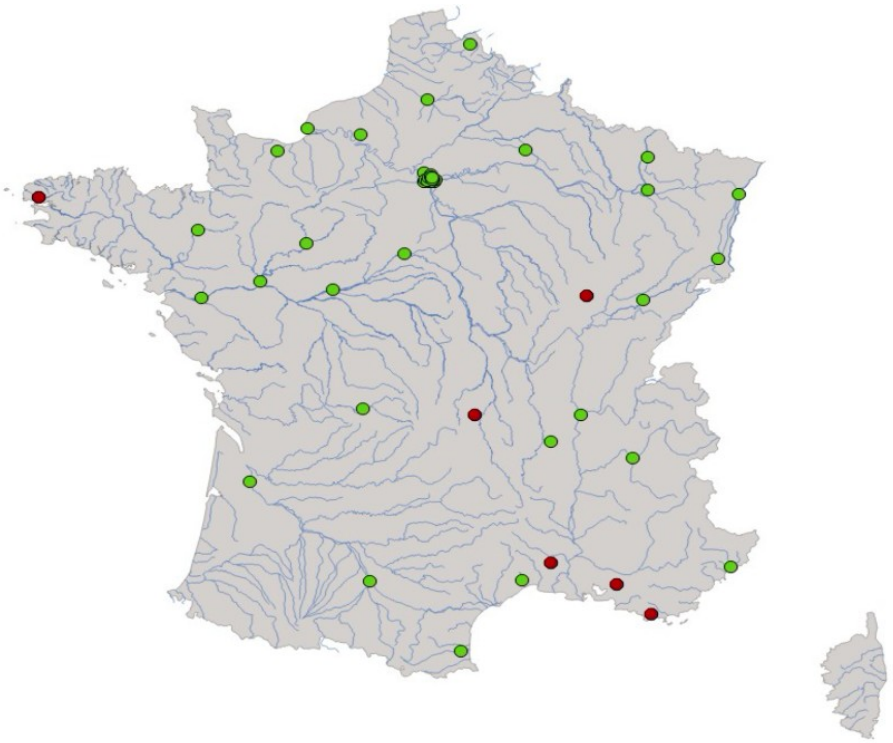
PGConf.ru 2018

Main rivers, bridges and cities...

```
SELECT city_name
FROM cities
WHERE population > 100000


EXCEPT


SELECT DISTINCT city_name
FROM cities c, rivers r
WHERE c.population > 100000
AND r.type = ‘major’
AND ST_Intersects(c.geom, r.geom)
```

```
SELECT city_name
FROM cities
WHERE population > 100000

EXCEPT

SELECT DISTINCT city_name
FROM cities c, rivers r
WHERE c.population > 100000
AND r.type = ‘major’
AND ST_Intersects(c.geom, r.geom)
```

CLERMONT-FERRAND
NIMES
BREST
DIJON
AIX-EN-PROVENCE
TOULON



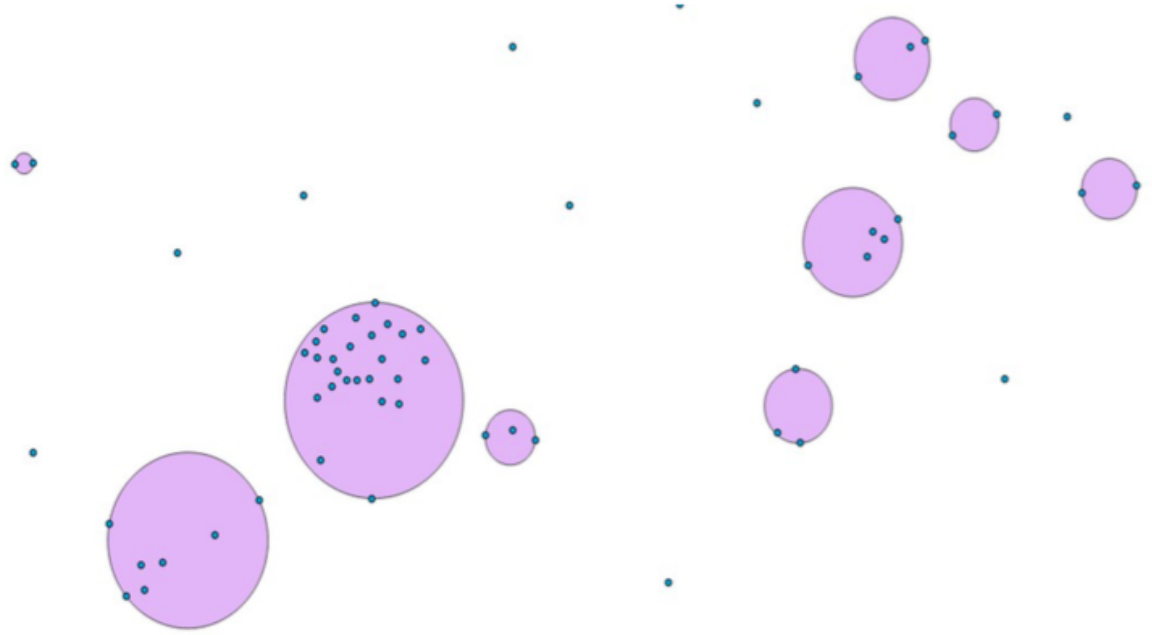
“Everything is related to everything else,
but near things are more related than distant things.”

**W. Tobler**

# LATERAL and KNN

```
SELECT
   bus.gid, bus.nom, lat.gid, lat.nom, lat.dist
FROM
   own.tcl as bus
   , LATERAL (
      SELECT
         bar.gid, bar.nom, bar.geom
         , ST_Distance(bar.geom, bus.geom) as dist
      FROM
         own.water as bar
      ORDER BY
         bar.geom <-> bus.geom  -- forbidden without lateral
      LIMIT 2
   ) AS lat
ORDER BY
   bus.gid, lat.dist desc;
```

# ST_ClusterWithin

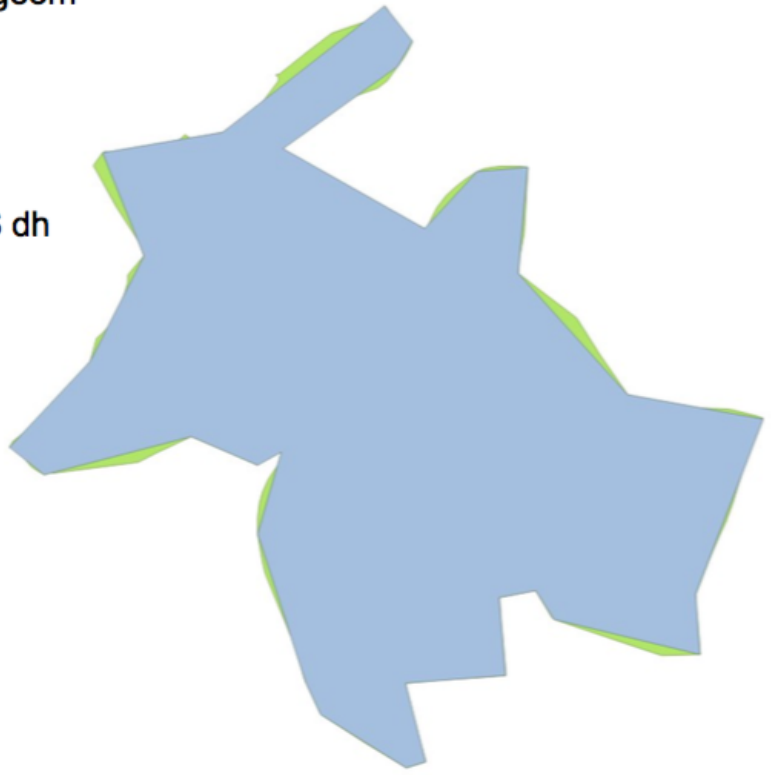


```
WITH

 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
```

Some PostGIS built-in functions
for advanced spatial analysis

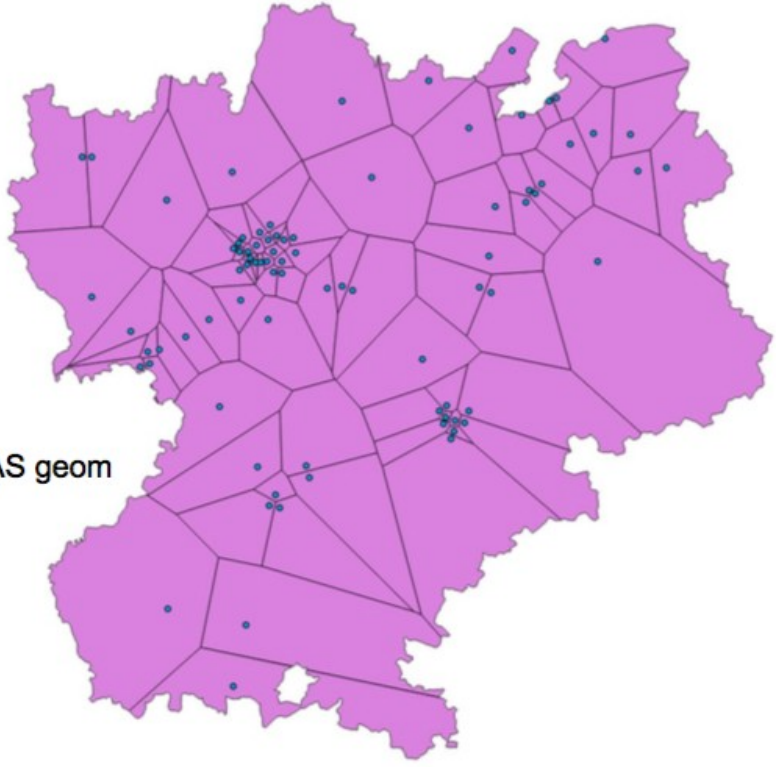
# ST_HausdorffDistance

```
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
```

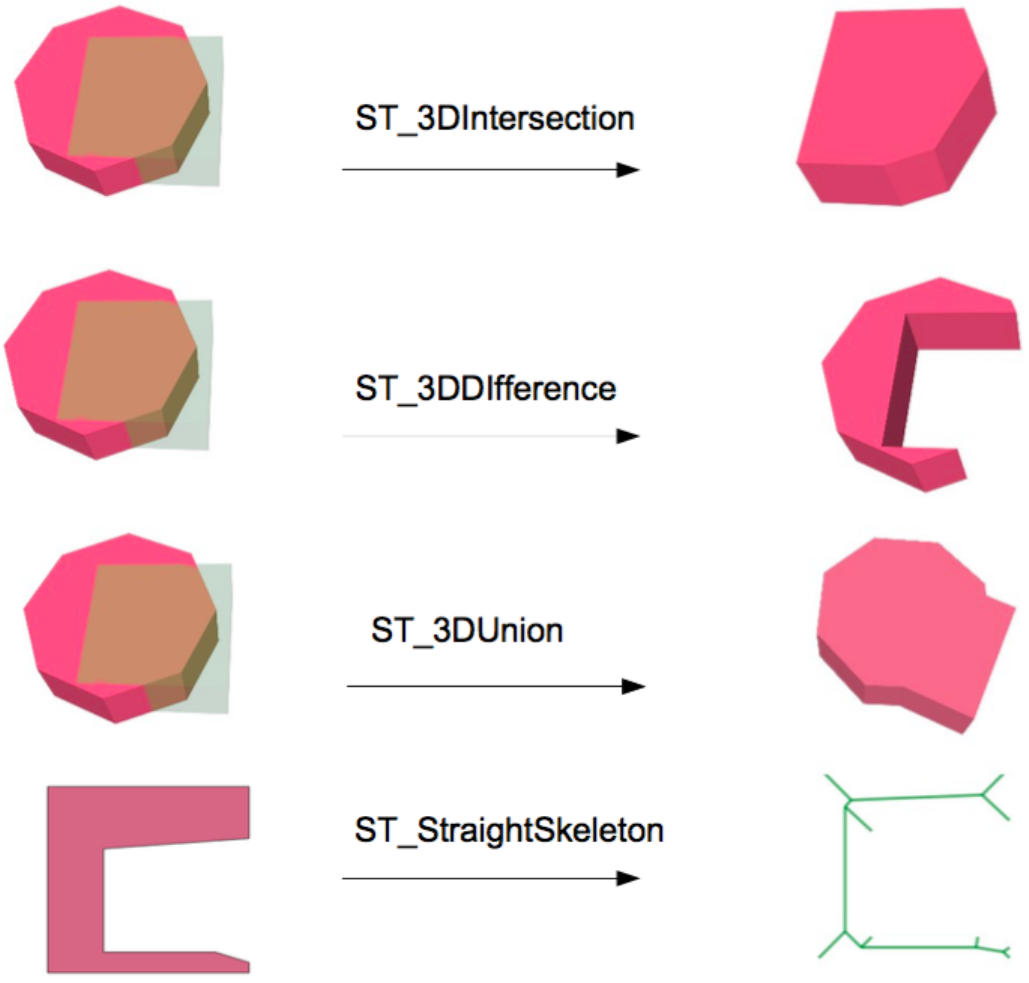
# ST_Voronoi



```
WITH c AS (

  SELECT ST_Centroid((ST_Dump(geom)).geom) AS geom
  FROM own.commune
  WHERE population > 10000

), v AS (

  SELECT ST_Intersection (

     (ST_Dump(ST_CollectionHomogenize(ST_Voronoi(ST_Collect(geom))))).geom,
           (SELECT ST_Union(geom) FROM own.commune)
       ) AS geom
  FROM c
)

SELECT geom, row_number() OVER() AS id FROM v
```

Need to deal with 3D too ?

# SFCGAL Extension



You still need more spatial functions ?

# Using existing Python Library from PostgreSQL
# Call throught SQL function

An example with GeoPy, Installation :

```
sudo apt-get install postgresql-plpython-9.4 python3-geopy

createdb db
createlang plpython3u db
psql db -c "CREATE EXTENSION postgis"
```

*Register on GeoNames*
*Enable your account to use the free WebService*

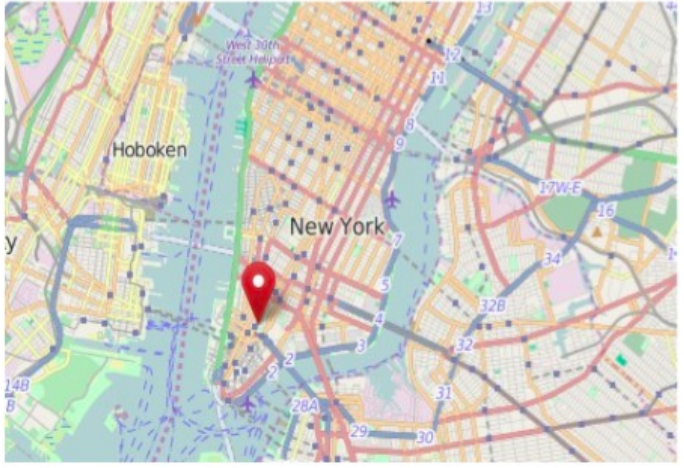
## Pl/Python basic Geocoder function

```
CREATE OR REPLACE FUNCTION geoname(toponym text)
                    RETURNS geometry(Point,4326)
AS $$

    from geopy import geocoders
    g = geocoders.GeoNames(username="YOUR_USERNAME")

    try:
        place, (lat, lng) = g.geocode(toponym)

        result = plpy.execute(
        "SELECT 'SRID=4326;POINT(%s %s)'::geometry(Point, 4326) AS geom"
        % (lng, lat), 1)

        return result[0]["geom"]

    except:
        plpy.warning('Geocoding Error')
        return None


$$ LANGUAGE plpython3u;
```

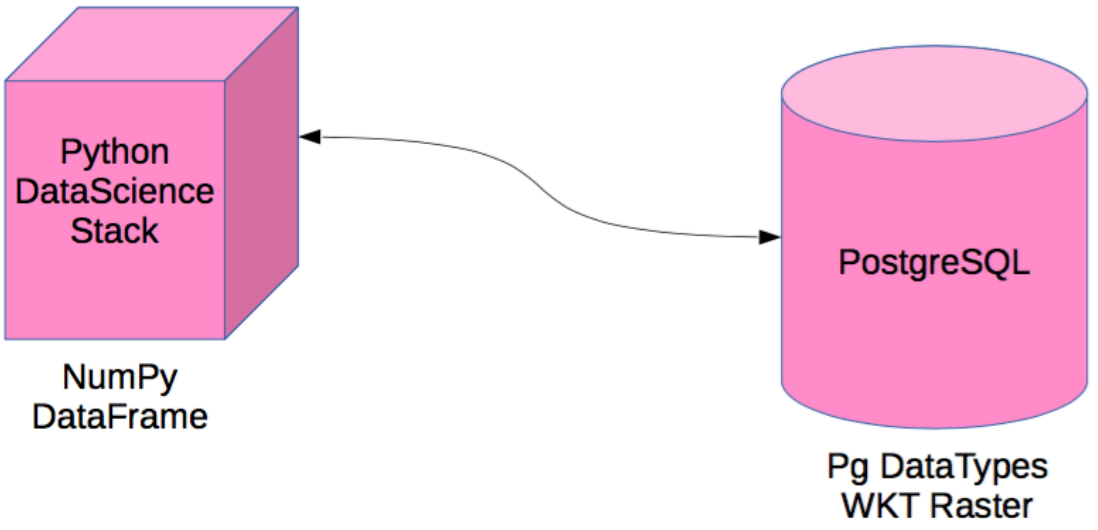
Check it :

```
psql db -c
"SELECT ST_AsGeoJSON(geoname('New York, NY 10022'))"

{"type":"Point","coordinates":[-74.00597,40.71427]}
```

http://www.openstreetmap.org/?mlon=-74.00597&mlat=40.71427&zoom=12



And now how to still go further ?


Python
DataScience
Stack
NumPy
DataFrame
PostgreSQL
Pg DataTypes
WKT Raster

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

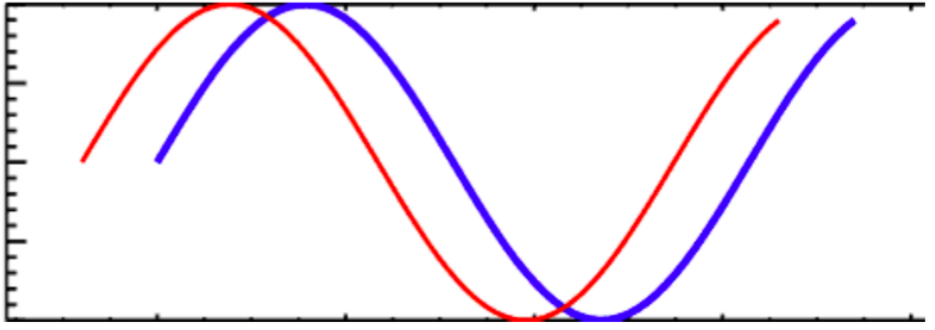
SciPy.org Sponsored By ENTHOUGHT

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;
```

How convert PostGIS types to Python type ?

## WKB Raster

Read WKB rasters to Numpy arrays.

## Docs

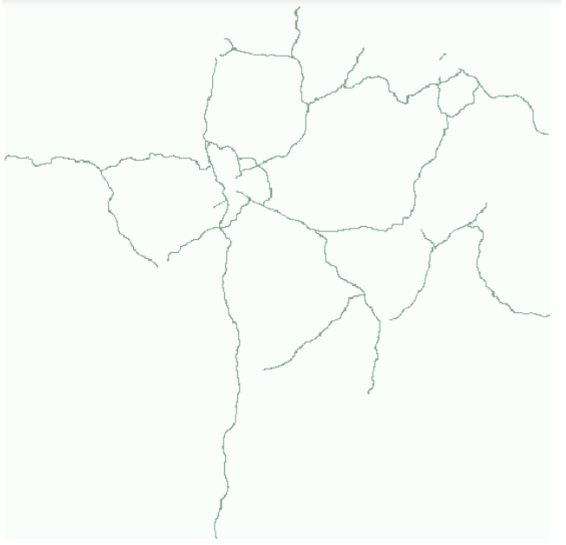
```
wkb_raster.read_wkb_raster(wkb)
```

Parameters

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

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

```
from wkb_raster import read_wkb_raster
from io import BytesIO

f, ax = plt.subplots(1, figsize=(10, 10))
ax.set_axis_off()

rasts = pg.execute("""

WITH h AS (

SELECT ST_Collect(way) AS geom
FROM planet_osm_line
WHERE highway = 'motorway'

)

SELECT ST_AsBinary(ST_AsRaster(geom, 500, 500)) as bin FROM h

""")

for rast in rasts:
    plt.imshow(read_wkb_raster(BytesIO(rast['bin']))['bands'][0]['ndarray'], alpha=0.50, cmap='Greens')
```



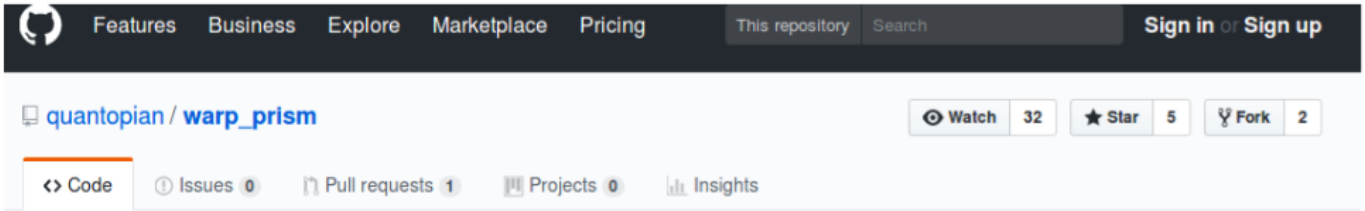
In [5]:
```
from sqlalchemy import create_engine
import pandas as pd

pg = create_engine(pg_uri)
```

In [6]:
```
df = pd.read_sql_query("""
    SELECT  "Value" v,
            "Timestamp" ts
    FROM iot
    ORDER BY ts
""", pg)

print(df)
```

```
             v                   ts
0         24.0 2016-10-11 17:10:00
1        137.0 2016-10-11 17:10:00
2        372.0 2016-10-11 17:10:00
3         30.0 2016-10-11 17:10:00
4        251.0 2016-10-11 17:10:00
5        284.0 2016-10-11 17:10:00
6         46.0 2016-10-11 17:10:00
```

In [7]:
```
np = df.values
print(np)
```

```
[[24.0 Timestamp('2016-10-11 17:10:00')]
 [137.0 Timestamp('2016-10-11 17:10:00')]
 [372.0 Timestamp('2016-10-11 17:10:00')]
 ...,
 [3.0 Timestamp('2017-03-07 11:40:00')]
 [0.0 Timestamp('2017-03-07 11:40:00')]
 [0.0 Timestamp('2017-03-07 11:40:00')]]
```




# warp_prism

Quickly move data from postgres to numpy or pandas.

## API

**`to_arrays(query, *, bind=None)`**

```
Run the query returning a the results as np.ndarrays.
```

**`to_dataframe(query, *, bind=None, null_values=None)`**

```
Run the query returning a the results as a pd.DataFrame.
```

https://github.com/quantopian/warp_prism

In [10]:
```
!psql -c 'CREATE TABLE foo AS (SELECT "Value" v, "Timestamp" ts FROM iot)' db
```

```
SELECT 2198006
```

In [11]:
```
import warp_prism as wp
from odo import resource

np = wp.to_arrays(resource(pg_uri + '::foo'))

print(np)
```

```
{'v': (array([  7.,   8.,  13., ...,  88.,  43.,  13.]), array([ True,  True,  True, ...,  True,  True,  True], dtype=bool)), 'ts': (array(['2016-11-22T07:20:00.000000', '2016-11-22T07:20:00.000000',
       '2016-11-22T06:40:00.000000', ..., '2017-01-27T00:20:00.000000',
       '2017-01-27T00:20:00.000000', '2017-01-27T00:20:00.000000'], dtype='datetime64[us]'), array([ True,  True,  True, ...,  True,  True,  True], dtype=bool))}
```

In [12]:
```
import warp_prism as wp
from odo import resource

df = wp.to_dataframe(resource(pg_uri + '::foo'))
print(df)
```

```
             v                   ts
0          7.0 2016-11-22 07:20:00
1          8.0 2016-11-22 07:20:00
2         13.0 2016-11-22 06:40:00
3         10.0 2016-11-22 07:20:00
4         16.0 2016-11-22 06:40:00
5         59.0 2016-11-22 06:40:00
6         25.0 2016-11-22 06:40:00
```

In [1]:

```
import sqlalchemy as sa
import geopandas as gpd

pg = sa.create_engine('postgresql://o:xxx@127.0.0.1:5432/osm')

lines = gpd.GeoDataFrame.from_postgis("""

SELECT way AS geom
FROM planet_osm_line
WHERE highway = 'motorway'

""", pg)

lines.head()
```

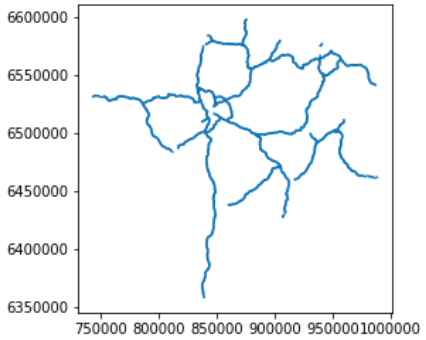
Out[1]:

| | geom |
|---|---|
| 0 | LINESTRING (837826.76 6367169.34, 837845.45 63... |
| 1 | LINESTRING (838497.59 6363225.84, 838485.18000... |
| 2 | LINESTRING (837701.14 6368190.28, 837706.21 63... |
| 3 | LINESTRING (837687.84 6368398.91, 837691.53 63... |
| 4 | LINESTRING (837686.62 6368847.87, 837683.63 63... |

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
lines.plot()
```

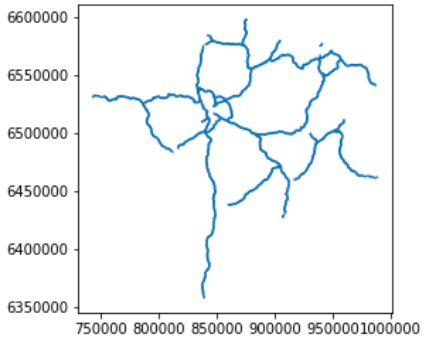
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f895e090d30>
```



```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
lines.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f895e090d30>
```




```
f, ax = plt.subplots(1, figsize=(8, 8))
ax.set_axis_off()
lines.plot(ax=ax, alpha=0.80, color='pink')
```

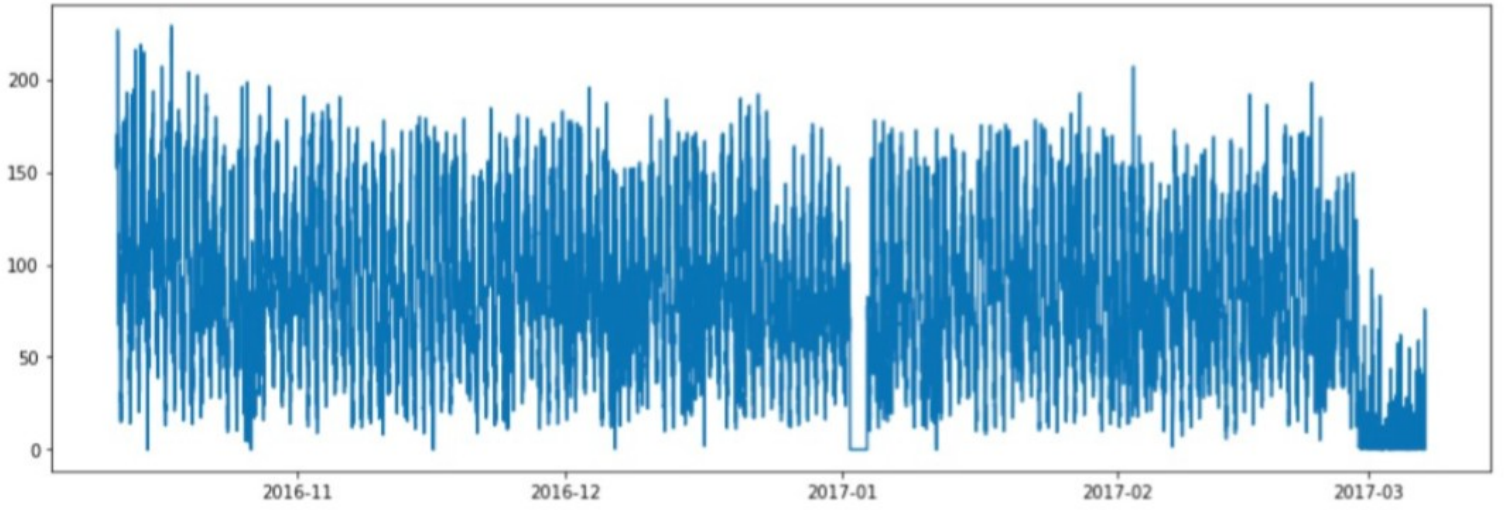
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f895a2292b0>
```

# A GeoSpatial TimeSeries example


https://opendata.paris.fr/explore/dataset/place-de-la-nation-points-de-mesure-flux/map/?location=18,48.84852,2.39599
PARISDATA
MAIRIE DE PARIS
Les données
L'API
La licence
La démarche
Cartographe
66 enregistrements
Aucun filtre actif.
Filtres
Rechercher...
type
turnstile 50
area 16
classes 2
ped 47
vehicle 17
bike 2
classes 3
bike 21
ped 1
Place de la Nation - Points de mesure flux
Informations
Tableau
Carte
Analyse
Export
API
id
7 118
nom
Cercle interieur - niveau Fabre d'Eglantine
metrics
[{"id": "direction_1", "name": "Sens inverse"}, {"id": "direction_2", "name": "Sens de circulation"}]
classes
all
type
turnstile
classes 2
vehicle
classes 3
bike
Boulevard Diderot
Avenue de Taillebourg
Avenue du Trône
Lycée général et technologique Arago
Nation
30 m

```
df = pd.read_sql_query("""

 WITH a AS (SELECT "Value" v, "Timestamp" ts, "Host" host FROM iot)

    SELECT avg(v), ts
    FROM a
    WHERE host IN ('7062', '6196', '7118')
    GROUP BY ts
    ORDER BY ts
    """, pg)

data = df.values
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 5))
plt.plot(data[:,1], data[:,0])
plt.show()
```



```
when = pd.read_sql_query("""

WITH a AS (

  SELECT "Timestamp" ts, "Value" v, "Host" host
  FROM iot

),

b AS (
  SELECT ts, v, host,
   sum(v) OVER (PARTITION by host ORDER BY ts
                ROWS BETWEEN CURRENT ROW AND 20 FOLLOWING) as cv
  FROM a
  WHERE host = '7062'
  ORDER BY ts
)

SELECT ts, v, host
FROM b
WHERE cv = 0
ORDER BY ts
LIMIT 1

 """, pg)

print(when.values)
```

```
[[Timestamp('2017-01-01 22:30:00') 0.0 7062]]
```

# Prophet: Automatic Forecasting Procedure

Prophet is a procedure for forecasting time series data. It is based on an additive model where non-linear trends are fit with yearly and weekly seasonality, plus holidays. It works best with daily periodicity data with at least one year of historical data. Prophet is robust to missing data, shifts in the trend, and large outliers.

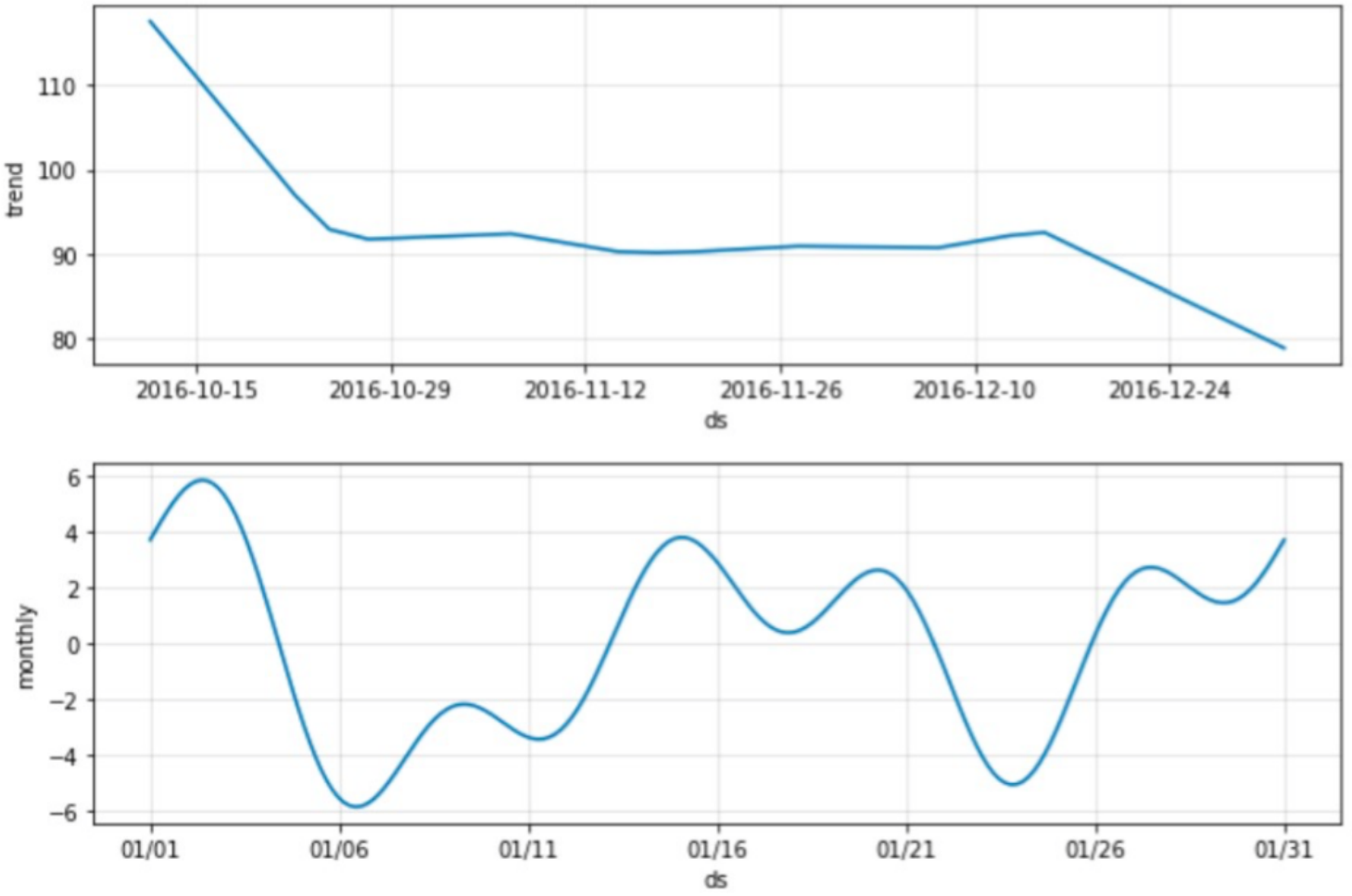
Prophet is open source software released by Facebook's Core Data Science team. It is available for download on CRAN and PyPI.

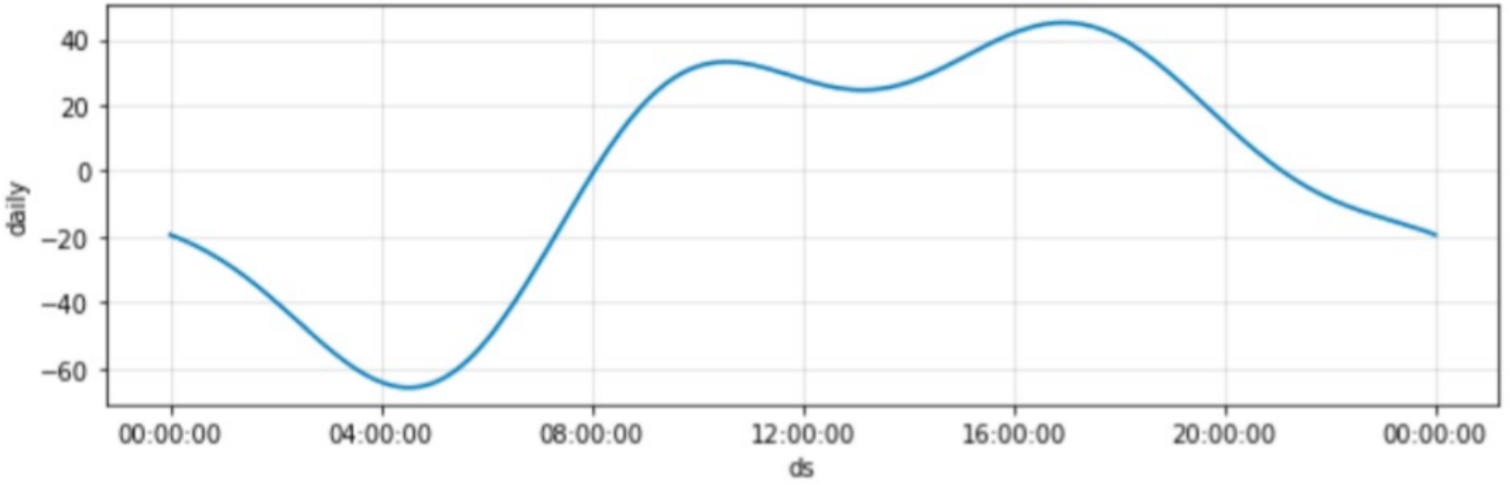
https://github.com/facebook/prophet

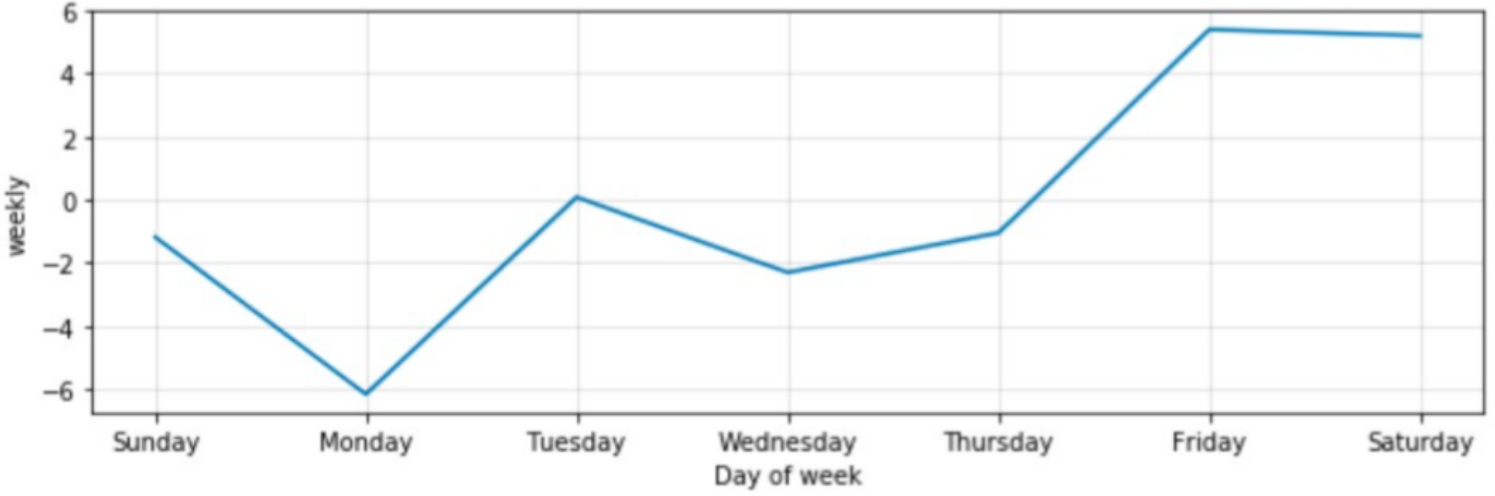
```
import pandas as pd

df = pd.DataFrame({'y': data[:,0], 'ds': data[:,1]})
```

```
from fbprophet import Prophet

m = Prophet()
m.add_seasonality(name='monthly', period=30, fourier_order=5)
m.fit(df)
future = m.make_future_dataframe(periods=30, freq='H')
forecast = m.predict(future)
```

```
m.plot_components(forecast)
```


trend
110
100
90
80
2016-10-15
2016-10-29
2016-11-12
2016-11-26
2016-12-10
2016-12-24
ds
monthly
6
4
2
0
−2
−4
−6
01/01
01/06
01/11
01/16
01/21
01/26
01/31
ds


40
20
0
−20
−40
−60
daily
00:00:00
04:00:00
08:00:00
12:00:00
16:00:00
20:00:00
00:00:00
ds


6
4
2
0
−2
−4
−6
weekly
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


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

azavea

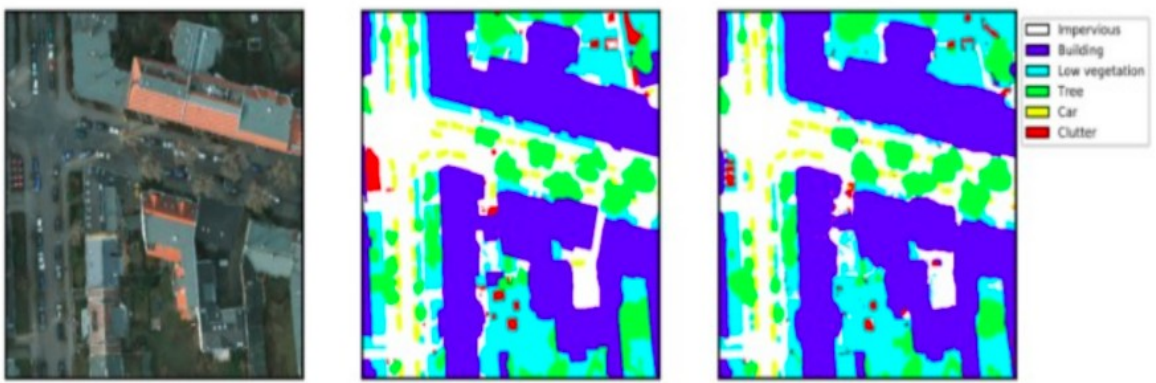
SOFTWARE DEVELOPMENT

# Deep Learning for Semantic Segmentation of Aerial Imagery

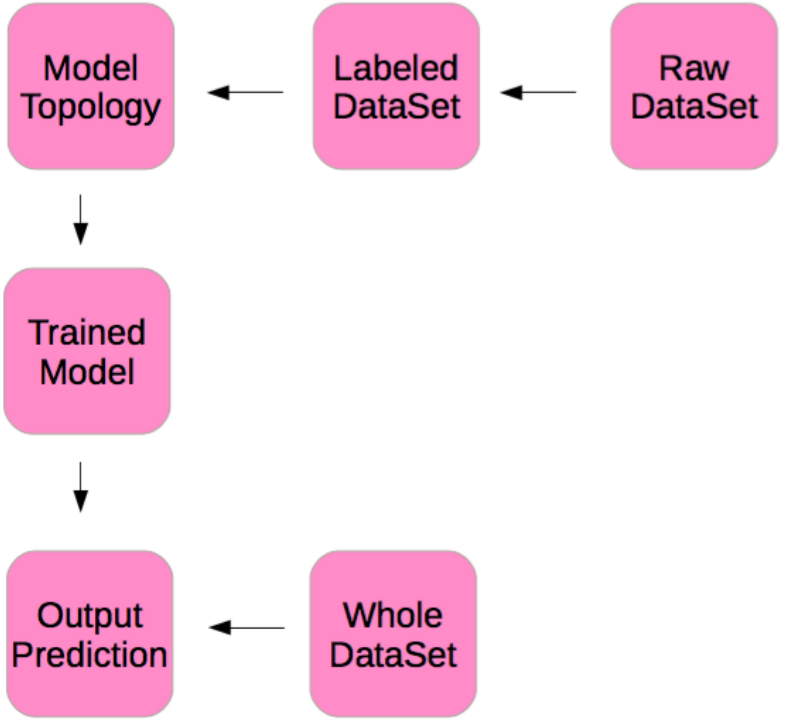
By Rob Emanuele on May 30th, 2017

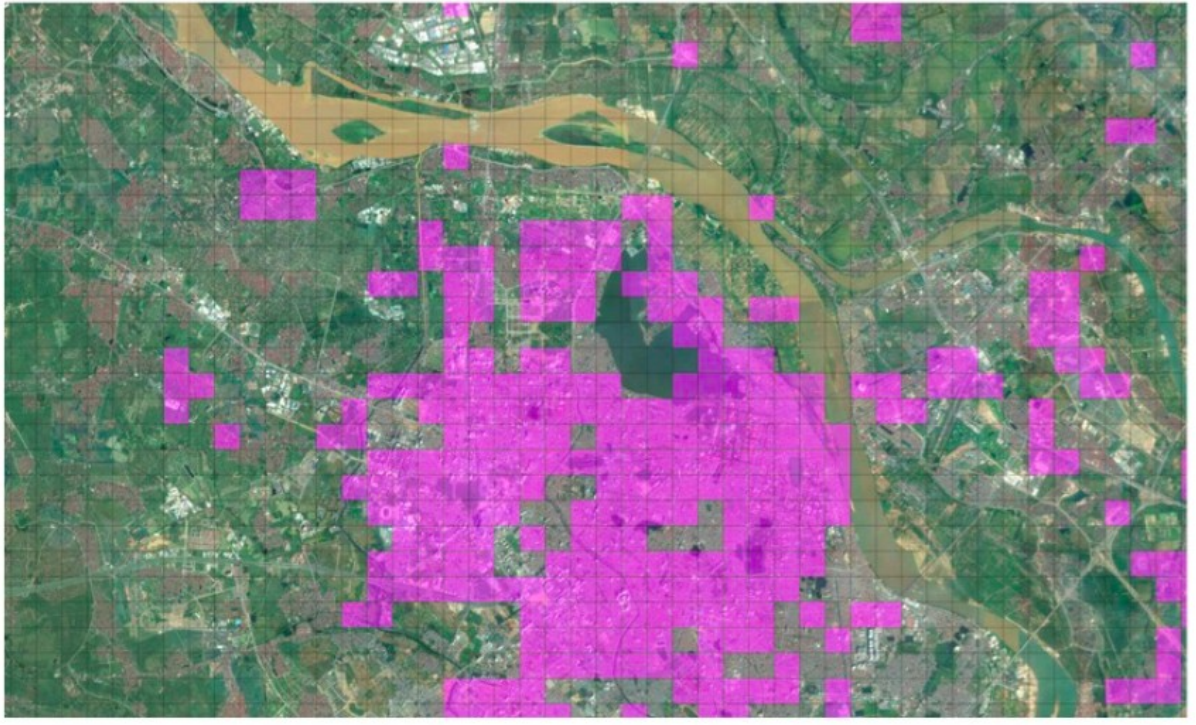
Pre-trained ResNet50 with ImageNet on IR-R-G

| | 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 |




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


Model Topology
Labeled DataSet
Raw DataSet
Trained Model
Output Prediction
Whole DataSet

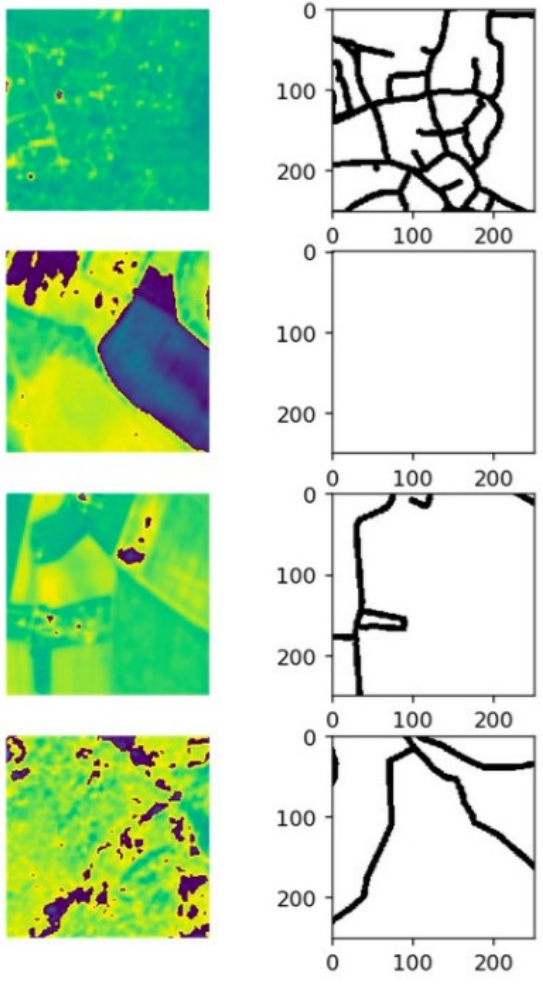


Purple building tile labels overlaid over Mapbox Satellite Imagery.

https://developmentseed.org/

```
WITH
origins AS (SELECT ('{{855878,6534055},{878721,6533022},{873294,6541341},{870027,6524893}}'::float[]) AS ul ),
tiles AS (
     SELECT row_number() OVER() as tid,
            ST_SetSRID(
                ST_MakeEnvelope(ul[i][1], ul[i][2], ul[i][1] + 1250, ul[i][2] + 1250)
                , 2154
            ) AS geom
     FROM origins, generate_subscripts((SELECT ul FROM origins), 1) AS i
  ),
tile_rast AS
(
    SELECT  tiles.tid,
            ST_AddBand(
                ST_SetSRID(
                    ST_MakeEmptyRaster(
                        250, 250,
                        ST_Xmin(tiles.geom)::float8,
                        ST_Ymax(tiles.geom)::float8,
                    2.5),
                2154),
            '8BUI') AS rast
    FROM tiles
),
images AS
(
    SELECT  tile_rast.tid,
            tile_rast.rast AS tile_rast,
            ST_MapAlgebra(
                ST_AddBand(tile_rast.rast, '8BUI'::text), 1,
                ST_Resample(ST_Grayscale(ST_Union(image.rast)), tile_rast.rast, 'bilinear'), 1,
                '[rast2] ', NULL, 'FIRST', '[rast2]'
            ) AS rast

    FROM tile_rast, LATERAL
    (
        SELECT rast
        FROM sat.s2
        WHERE ST_Intersects(s2.rast, tile_rast.rast)

    ) AS image

    GROUP BY tile_rast.rast, tile_rast.tid
  ),
labels AS (
    SELECT  tile_rast.tid,
            ST_MapAlgebra(
                 tile_rast.rast,
                 ST_AsRaster(label.geom, tile_rast.rast, '8BUI'),
                 '([rast2])::integer', NULL, 'FIRST', '([rast2])::integer'
             ) AS rast

    FROM tile_rast, LATERAL
    (
       SELECT ST_ClipByBox2D(ST_Buffer(ST_Union(osm.way), 10),
                             ST_Envelope(tile_rast.rast)) geom
       FROM planet_osm_line osm
       WHERE osm.highway IS NOT NULL AND (osm.route = 'road' OR osm.route IS NULL)
       AND ST_Intersects(osm.way, tile_rast.rast)

       GROUP BY tile_rast.rast, tile_rast.tid

    ) AS label
)
SELECT Box3D(images.rast) AS bbox,
       ST_AsBinary(images.rast) AS data,
       CASE WHEN labels.rast IS NULL
           THEN ST_AsBinary(images.tile_rast)
           ELSE ST_AsBinary(labels.rast)
       END AS label
FROM  labels RIGHT JOIN images ON images.tid = labels.tid
```

```
batch_size = 2
max_iter = 2

geo_iter = GeoIter(
    'postgresql://o:xxx@127.0.0.1:5433/osm_qa',
    (850000,6524040,890960,6565000), 2154, (256, 256), (10, 2.5),

    """

         SELECT ST_ClipByBox2D(ST_Buffer(ST_Union(osm.way), 6),
                               ST_Envelope(tile_rast.rast)) geom
         FROM planet_osm_line osm
         WHERE osm.highway IS NOT NULL AND (osm.route = 'road' OR osm.route IS NULL)
            AND ST_Intersects(osm.way, tile_rast.rast)

""",

    """

        SELECT ST_Grayscale(ST_Union(s2.rast)) AS rast
        FROM sat.s2
        WHERE ST_Intersects(s2.rast, tile_rast.rast)

    """,

    batch_size, max_iter)
```


0
100
200
0
100
200
0
100
200
0
100
200
0
100
200
0
100
200
0
100
200
0
100
200

# Conclusions


@data_pink

www.datapink.com