Gradient of a spatial point process along a linear feature

Miklós Koren[1]
[1]Central European University

dx.doi.org/10.17504/protocols.io.ubhesj6

Miklós Koren
Central European University

In devel.
Oct 04, 2018

ABSTRACT

How we measure the gradient of he density of 19th century post offices along North American rivers in our paper Armenter, Roc, Miklós Koren and Dávid K Nagy. “Bridges”

Method relies on maximum likelihood formula in Zhao, M., and M. Xie. 1996. “On Maximum Likelihood Estimation for a General Non-Homogeneous Poisson Process.” Scandinavian Journal of Statistics, Theory and Applications 23 (4): 597–607.

THIS PROTOCOL ACCOMPANIES THE FOLLOWING PUBLICATION

Armenter, Roc, Miklós Koren and Dávid K Nagy. 2018. “Bridges.” Working paper.

PROTOCOL STATUS

In development
We are still developing and optimizing this protocol

Convert all features to a distance-invariant projection

1 Most geospatial data are in WGS84, which is not distance-invariant. Convert everything (points, lines) to a distance-invariant projection such as Lambert Conformal Conic. Make sure (x,y) coordinates are in meters.

```
import pyproj
from shapely.ops import transform
from shapely.geometry.base import BaseGeometry
from shapely.geometry import asShape

from shapely.geometry import Point

PROJECTIONS = dict(
    wgs84 = pyproj.Proj("+init=EPSG:4326"),
    lcc = pyproj.Proj(proj="lcc"),
    nhgis = pyproj.Proj("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs"),
    epsg3975 = pyproj.Proj("+proj=cea +lon_0=0 +lat_ts=30 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m
+no_defs"),
    albers = pyproj.Proj("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +datum=NAD83
+units=m +no_defs "),
    eov = pyproj.Prof("+proj=somerc +lat_0=47.14439372222222 +lon_0=19.04857177777778 +k_0=0.99993
+x_0=650000 +y_0=200000 +ellps=GRS67 +towgs84=52.17,-71.82,-14.9,0,0,0,0 +units=m +no_defs ")
)

def project_geometry(geometry, tfrom, tto):
    return transform(lambda x, y: pyproj.transform(PROJECTIONS[tfrom], PROJECTIONS[tto], x, y), geometry)

class ProjectedFeature(object):
    def __init__(self, geometry, projection):
        '''
        Every incoming geometry (point, linestring, polygon) is stored in wgs84.
        '''
        if not isinstance(geometry, BaseGeometry):
            # if not shapely geometry, treat as geojson dictionary
            geometry = asShape(geometry)
        self.geometry = project_geometry(geometry, projection, 'wgs84')

    def __getattr__(self, projection):
        '''
        We can refer to projected features as an attribute, e.g.

            feature.wgs84
            feature.lcc
        '''
        if projection in PROJECTIONS:
            return project_geometry(self.geometry, 'wgs84', projection)
        else:
            raise AttributeError

if __name__ == '__main__':
    union_sq = Point((1827011,574994))
    union_sq_geojson = {
        "type": "Point",
        "coordinates": [1827011, 574994]
    }

    Union_Sq = ProjectedFeature(union_sq, 'nhgis')
    print Union_Sq.wgs84
    print Union_Sq.lcc

    Union_Sq = ProjectedFeature(union_sq_geojson, 'nhgis')
    print Union_Sq.wgs84
```

Select relevant sample of points close to linear feature

2 Create a polygon around your linear feature by extending it a D distance either circularly or by parallel shift

```
from project_geojson import ProjectedFeature
import json
from shapely.geometry.geo import mapping
from shapely.geometry import LineString, Point, Polygon, asShape
import sys

def bank(river, side, width_km=10):
 bank = river.parallel_offset(width_km*1000.0, side)
 river_coords = list(river.coords)
 bank_coords = list(bank.coords)
 if side=='left':
  bank_coords = bank_coords[::-1]
 linestring = river_coords+bank_coords
 return Polygon(linestring)

if __name__ == '__main__':
 which_bank = sys.argv[1]
 width_km = 10

 river_json = json.load(sys.stdin)
 river = ProjectedFeature(river_json['features'][0]['geometry'], 'wgs84').lcc
 bank = ProjectedFeature(bank(river, which_bank, width_km), 'lcc').wgs84.buffer(0)
 meta = river_json['features'][0]['properties']
 meta['bank'] = which_bank
 meta['width_km'] = 10.0
```

```
print json.dumps(dict(type="Feature", geometry=mapping(bank), properties=meta))
```

Select the intersection of this polygon with the points of interest.

Project points on linear feature and convert to linear coordinates

3 Make sure you know where the start and end of your linear feature is. Measure coordinates as distance along the feature to the start.

```
bridge_rivermile = river.project(bridge)
postoffice_rivermile = river.project(postoffice)
```

For each point, measure distance to nearest crossing point

4

```
def distance_to_nearest_bridge(river_mile, bridges):
    return min([abs(bridge['river_mile']-river_mile) for bridge in bridges])
for postoffice in postoffices:
    postoffice['distance_to_bridge'] = distance_to_nearest_bridge(postoffice['river_mile'], bridges)
```

Given the distribution of distances, estimate gradient parameters by maximum likelihood

5 Given a distance function $f()$,

$$\lambda(z) = e^{a_j - bf(|z-x_j|)}.$$

The log likelihood is then

$$\ln \mathcal{L}(x,y,\theta) = n(T)\ln M(T) - \ln[n(T)!] - 2M(T) + n(T)a_j - b\sum_{i=1}^{N(T)} f(|y_i - x_j|).$$