Flood Tool

Team Frome

Oct 25, 2019

CONTENTS

This package implements a flood risk prediction tool.

CHAPTER

ONE

GEODETIC TRANSFORMATIONS

For historical reasons, multiple coordinate systems exist in British mapping. The Ordnance Survey has been mapping the British Isles since the 18th Century and the last major retriangulation from 1936-1962 produced the Ordance Survey National Grid (or **OSGB36**), which defined latitude and longitude across the island of Great Britain[1]. For convenience, a standard Transverse Mercator projection[2] was also defined, producing a notionally flat gridded surface, with gradations called eastings and westings. The scale for these gradations was identified with metres.

The OSGB36 datum is based on the Airy Ellipsoid of 1830, which defines semimajor axes for its model of the earth, a and b, a scaling factor F_0 and ellipsoid height, H.

$$\begin{aligned} a &= 6377563.396, \\ b &= 6356256.910, \\ F_0 &= 0.9996012717, \\ H &= 24.7. \end{aligned}$$

The point of origin for the transverse Mercator projection is defined in the Ordnance Survey longitude-latitude and easting-northing coordinates as

$$\begin{aligned} \phi_0^{OS} &= 49^\circ \text{ north}, \\ \lambda_0^{OS} &= 2^\circ \text{ west}, \\ E_0^{OS} &= 400000m, \\ N_0^{OS} &= -100000m. \end{aligned}$$

More recently, the world has gravitated towards the use of Satellite based GPS equipment, which uses the (globally more appropriate) World Geodetic System 1984 (or **WGS84**). This datum uses a different ellipsoid, which offers a better fit for a global coordinate system. Its key properties are:

$$\begin{aligned} a_{WGS} &= 6378137, , \\ b_{WGS} &= 6356752.314, \\ F_0 &= 0.9996. \end{aligned}$$

For a given point on the WGS84 ellipsoid, an approximate mapping to the OSGB36 datum can be found using a Helmert transformation[3],

$$\mathbf{x}^{OS} = \mathbf{t} + \mathbf{M}\mathbf{x}^{WGS}.$$

[1] A guide to coordinate systems in Great Britain, Ordnance Survey

[2] Map projections - A Working Manual, John P. Snyder, https://doi.org/10.3133/pp1395

[3] Computing Helmert transformations, G Watson, http://www.maths.dundee.ac.uk/gawatson/helmertrev.pdf

Here $\mathbf{x}$ denotes a coordinate in Cartesian space (i.e in 3D) as given by the (invertible) transformation

$$\begin{aligned}
e^2 &= \frac{a^2 - b^2}{a^2} \\
n &= \frac{a-b}{a+b} \\
\nu &= \frac{aF_0}{\sqrt{1 - e^2\sin^2(\phi)}} \\
x &= (\nu + H)\sin(\lambda)\cos(\phi) \\
y &= (\nu + H)\cos(\lambda)\cos(\phi) \\
z &= ((1-e^2)\nu + H)\sin(\phi)
\end{aligned}$$

and the transformation parameters are

$$\begin{aligned}
\mathbf{t} &= \begin{pmatrix} -446.448 \\ 125.157 \\ -542.060 \end{pmatrix}, \\
\mathbf{M} &= \begin{bmatrix} 1+s & -r_3 & r_2 \\ r_3 & 1+s & -r_1 \\ -r_2 & r_1 & 1+s \end{bmatrix}, \\
s &= 20.4894 \times 10^{-6}, \\
\mathbf{r} &= [0.1502'', 0.2470'', 0.8421''].
\end{aligned}$$

Given a latitude, ϕ^{OS} and longitude, λ^{OS} in the OSGB36 datum, easting and northing coordinates, E^{OS} & N^{OS} can then be calculated using the following formulae:

$$
\begin{aligned}
\rho &= \frac{aF_0(1-e^2)}{\left(1-e^2\sin^2(\phi^{OS})\right)^{\frac{3}{2}}}\\
\eta &= \sqrt{\frac{\nu}{\rho}-1}\\
M &= bF_0\left[\left(1+n+\frac{5}{4}n^2+\frac{5}{4}n^3\right)(\phi^{OS}-\phi^{OS}_0)\right.\\
&\quad-\left(3n+3n^2+\frac{21}{8}n^3\right)\sin(\phi^{OS}-\phi^{OS}_0)\cos(\phi^{OS}+\phi^{OS}_0)\\
&\quad+\left(\frac{15}{8}n^2+\frac{15}{8}n^3\right)\sin(2(\phi^{OS}-\phi^{OS}_0))\cos(2(\phi^{OS}+\phi^{OS}_0))\\
&\quad\left.-\frac{35}{24}n^3\sin(3(\phi^{OS}-\phi^{OS}_0))\cos(3(\phi^{OS}+\phi^{OS}_0))\right]\\
I &= M+N^{OS}_0\\
II &= \frac{\nu}{2}\sin(\phi^{OS})\cos(\phi^{OS})\\
III &= \frac{\nu}{24}\sin(\phi^{OS})cos^3(\phi^{OS})(5-\tan^2(phi^{OS})+9\eta^2)\\
IIIA &= \frac{\nu}{720}\sin(\phi^{OS})cos^5(\phi^{OS})(61-58\tan^2(\phi^{OS})+\tan^4(\phi^{OS}))\\
IV &= \nu\cos(\phi^{OS})\\
V &= \frac{\nu}{6}\cos^3(\phi^{OS})\left(\frac{\nu}{\rho}-\tan^2(\phi^{OS})\right)\\
VI &= \frac{\nu}{120}\cos^5(\phi^{OS})(5-18\tan^2(\phi^{OS})+\tan^4(\phi^{OS})\\
&\quad+14\eta^2-58\tan^2(\phi^{OS})\eta^2)\\
E^{OS} &= E^{OS}_0+IV(\lambda^{OS}-\lambda^{OS}_0)+V(\lambda^{OS}-\lambda^{OS}_0)^3+VI(\lambda^{OS}-\lambda^{OS}_0)^5\\
N^{OS} &= I+II(\lambda^{OS}-\lambda^{OS}_0)^2+III(\lambda^{OS}-\lambda^{OS}_0)^4+IIIA(\lambda^{OS}-\lambda^{OS}_0)^6
\end{aligned}
$$

CHAPTER

TWO

ALGORITHM CHOICE

For the core function `get_easting_northing_flood_probability` we decided to use the following idea:

1. Create columns in our risk data for `x_min := x - radius`, `x_max := x + radius` and similarly `y_min` and `y_max`, and `radius_sqaured := radius * radius`.

2. **For each point - an easting and northing, `(e,n)` we:** 2.1 Check that it's within the 'square' of a flood risk zone's centroid.

   This step vastly reduces the number of potential zones that the point may lie in, with minimal computation because we just need to keep rows of flood risk zones where the northings and eastings of the point are within the `x_min,x_max,y_min,y_max` that we calculated prior.

   2.2 Then calculate the distance squared between the point and the zone's centroid, `d2` (we use squares to avoid calculating any square-roots), and check whether it's smaller than the `radius_squared` of the zone (which was calculated in step 1).

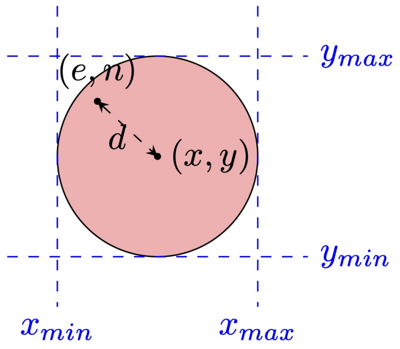

Fig. 1: Checking whether a point is actually in the flood risk zone.

   2.3 If it is, then find out the flood risk of the zone it's in, and appropriately set the risk of the point to that of the zone **if** it's not already in an existing higher risk zone.

CHAPTER

THREE

LIVE APIS

For the first extension task, we created a map in Folium that takes in real time data and alerts us to potential severity of flooding at certain postcodes. Red means a dangerous level of severity, orange means high severity, pink means medium severity, and so on. By inputting postcodes and using the get_lat_long function from our tool file, we attain the latitudes and longitudes of the inputted postcodes. We then put a range of 0.1 degrees around the latitude and longitude values of each postcode. 0.1 degrees comes out to be about 11km. The diameter of the circle around eachpostcode is therefore about 20km. We chose this value because the average width of a thunderstorm is around 20km.Afterwards, we found all the stations within this range and calculated the mean rainfall value. We then equated this to the rainfall value of the postcode. To forecast the severity of flooding, we looked at the difference between real time rainfall values and chosen threshold values. For example, for the highest risk areas, we chose a threshold value of 0, and the dangerous alert would come up when the difference comes up to be equal to or above 1mm/15minutes. This decision was based on case studies published by the Met Office; for example, in June 2012, the Southern Part of England experienced severe flooding from rainfall values over 1mm/15minutes. Many other examples can be seen in Met Office reports published on the internet. The output of the code can be seen here. At one point in time (yesterday 8pm) there was no risk of flooding (light green) in the area near Sandwich

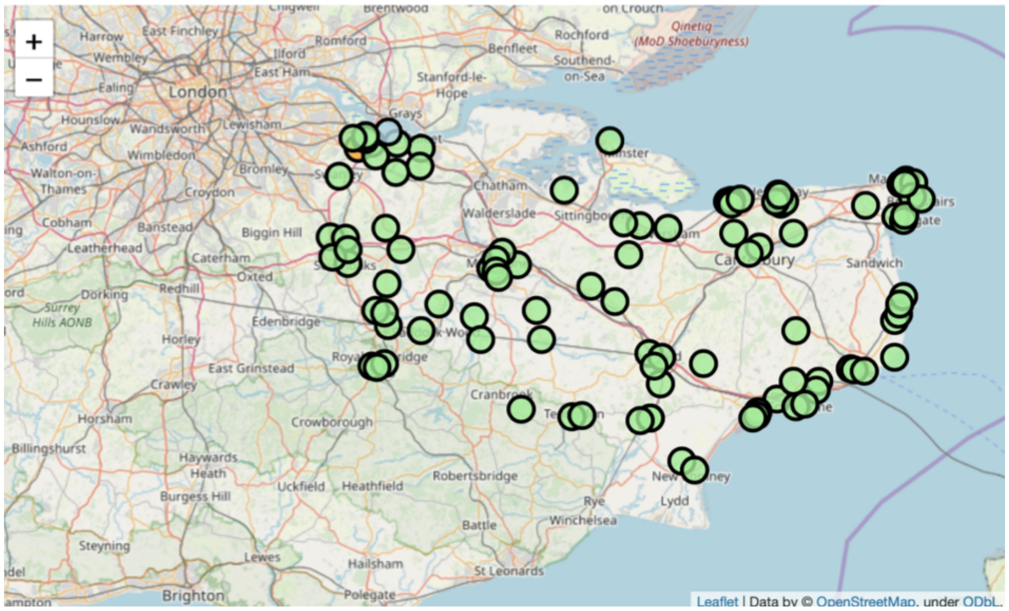

Fig. 1: Checking whether a point is actually in the flood risk zone.

For the second extension task, we wrote the code to categorize the daily rainfall values for areas in the whole of the UK. The lowest threshold was the average of the daily rainfall as calculated by the Met Office in the 1981-2010 period. The 2nd highest and highest thresholds are the 95th and 99th percentiles of daily rainfall as calculated by the

Royal Meteorological Office in the 1961-2010 period. The figure shows values on the 8th of October, 2019. The red indicates areas that had values higher than the 99th percentile of 13.4mm, whilst the light green indicates areas below the average of 2.34mm. Orange and blue indicates values above the 95th and average values respectively.

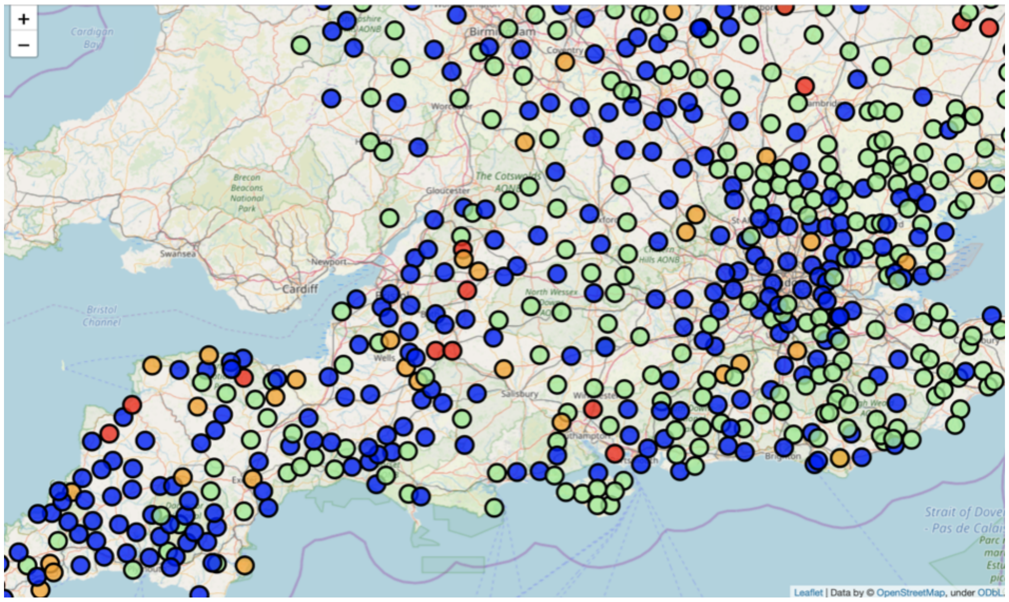

Fig. 2: Checking whether a point is actually in the flood risk zone.

CHAPTER

FOUR

FUNCTION APIS

class `flood_tool.`**`Tool`**(*postcode_file=None*, *risk_file=None*, *values_file=None*)

Class to interact with a postcode database file.

Reads postcode and flood risk files and provides a postcode locator service.

Parameters

- **`postcode_file`** (*`str, optional`*) – Filename of a .csv file containing geographic location data for postcodes.
- **`risk_file`** (*`str, optional`*) – Filename of a .csv file containing flood risk data.
- **`values_file`** (*`str, optional`*) – Filename of a .csv file containing property value data for postcodes.

`get_annual_flood_risk`(*postcodes*, *probability_bands*)

Get an array of estimated annual flood risk in pounds sterling per year of a flood event from a sequence of postcodes and flood probabilities. :param postcodes: Ordered collection of postcodes :type postcodes: sequence of strs :param probability_bands: Ordered collection of flood probabilities :type probability_bands: sequence of strs

Returns array of floats for the annual flood risk in pounds sterling for the input postcodes. Invalid postcodes return *numpy.nan*.

Return type numpy.ndarray

`get` $_easting\_northing\_flood\_probability$**(*`easting`*, *`northing`*)**

numpy array of flood probability bands corresponding to input locations.

Return type numpy.ndarray of strs

`get_flood_cost`(*postcodes*)

Get an array of estimated cost of a flood event from a sequence of postcodes. :param postcodes: Ordered collection of postcodes :type postcodes: sequence of strs

Returns array of floats for the pound sterling cost for the input postcodes. Invalid postcodes return *numpy.nan*.

Return type numpy.ndarray of floats

`get_lat_long`(*postcodes*)

Get an array of WGS84 (latitude, longitude) pairs from a list of postcodes.

Parameters **`postcodes`** (*`sequence of strs`*) – Ordered sequence of N postcode strings

Returns Array of Nx_2 (latitude, longitdue) pairs for the input postcodes. Invalid postcodes return [*numpy.nan*, *numpy.nan*].

Return type ndarray

get_sorted_annual *flood_risk* (***postcodes***)

postcodes (*sequence of strs*) – Ordered sequence of postcodes

Returns Dataframe of flood risks indexed by (normalized) postcode and ordered by risk, then by lexagraphic (dictionary) order on the postcode. The index is named *Postcode* and the data column *Flood Risk*. Invalid postcodes and duplicates are removed.

Return type pandas.DataFrame

get_sorted *flood_probability* (***postcodes***)

Dataframe of flood probabilities indexed by postcode and ordered from *High* to *Zero*, then by lexagraphic (dictionary) order on postcode. The index is named *Postcode*, the data column is named *Probability Band*. Invalid postcodes and duplicates are removed.

Return type pandas.DataFrame

`flood_tool.`**`WGS84toOSGB36`**(*latitude*, *longitude*, *radians=False*)

Wrapper to transform (latitude, longitude) pairs from GPS to OS datum.

`flood_tool.` **`get_easting_northing_from_lat_long`**(*latitude*, *longitude*, *radians=False*)

Convert GPS (latitude, longitude) to OS (easting, northing).

Parameters

- **latitude** (*sequence of floats*) – Latitudes to convert.
- **longitude** (*sequence of floats*) – Lonitudes to convert.
- **radians** (*bool, optional*) – Set to *True* if input is in radians. Otherwise degrees are assumed

Returns

- **easting** (*ndarray of floats*) – OS Eastings of input
- **northing** (*ndarray of floats*) – OS Northings of input

References

A guide to coordinate systems in Great Britain (https://webarchive.nationalarchives.gov.uk/20081023180830/http://www.ordnancesurvey.co.uk/oswebsite/gps/information/coordinatesystemsinfo/guidecontents/index.html)

References

PYTHON MODULE INDEX

INDEX