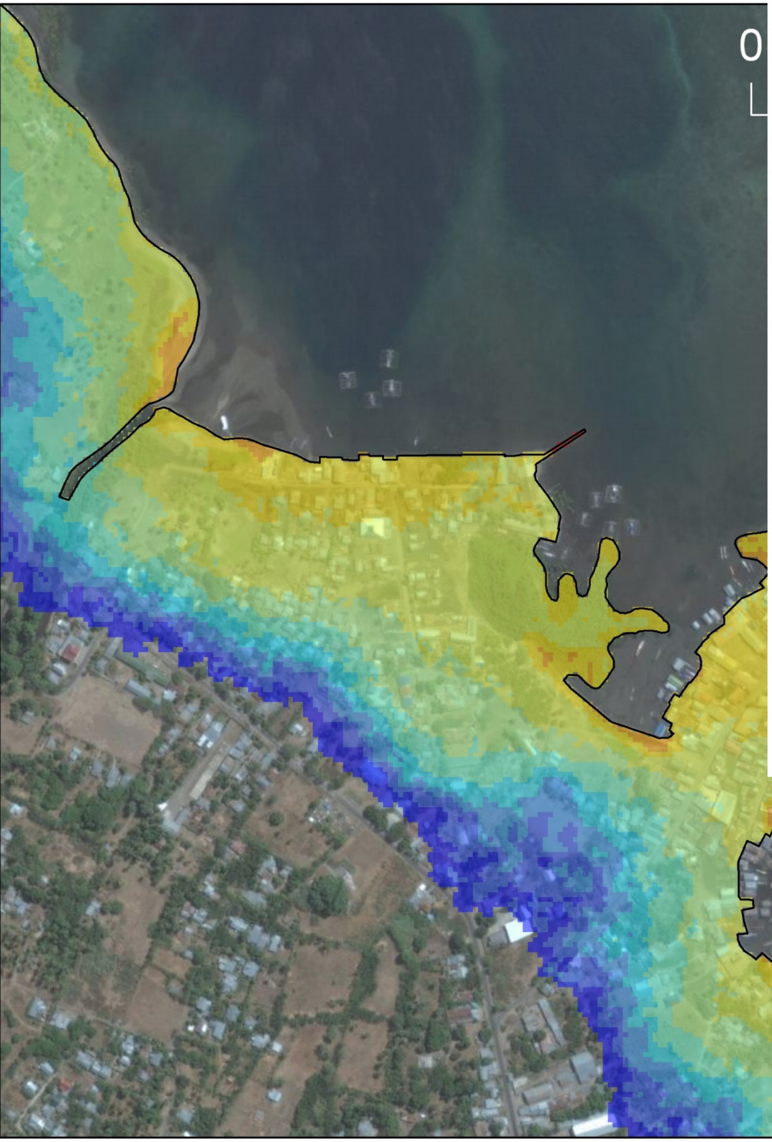
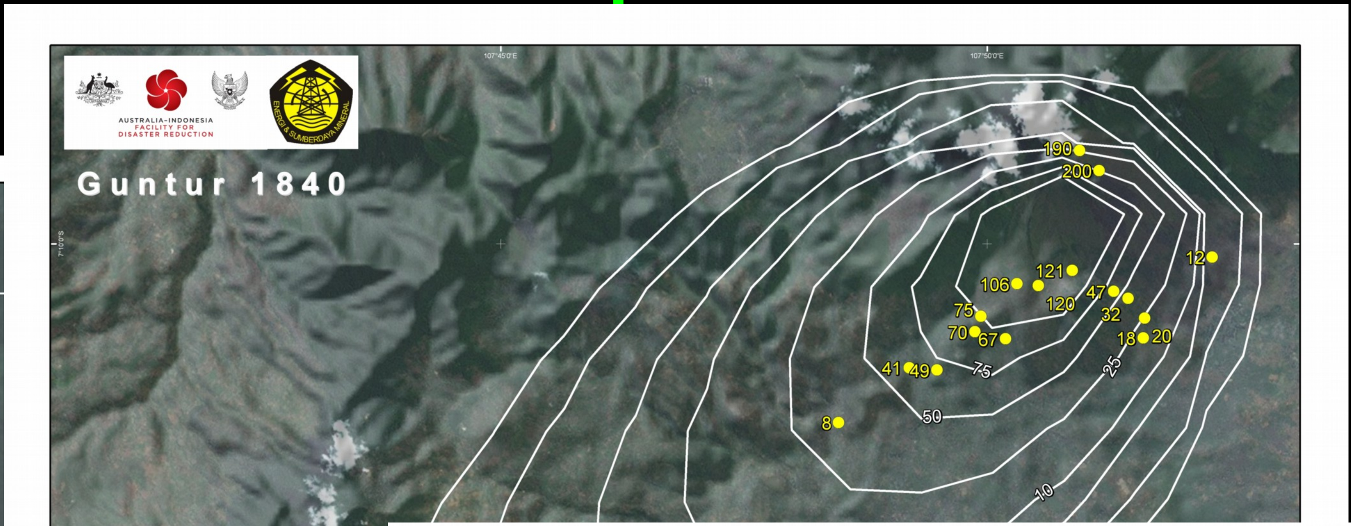
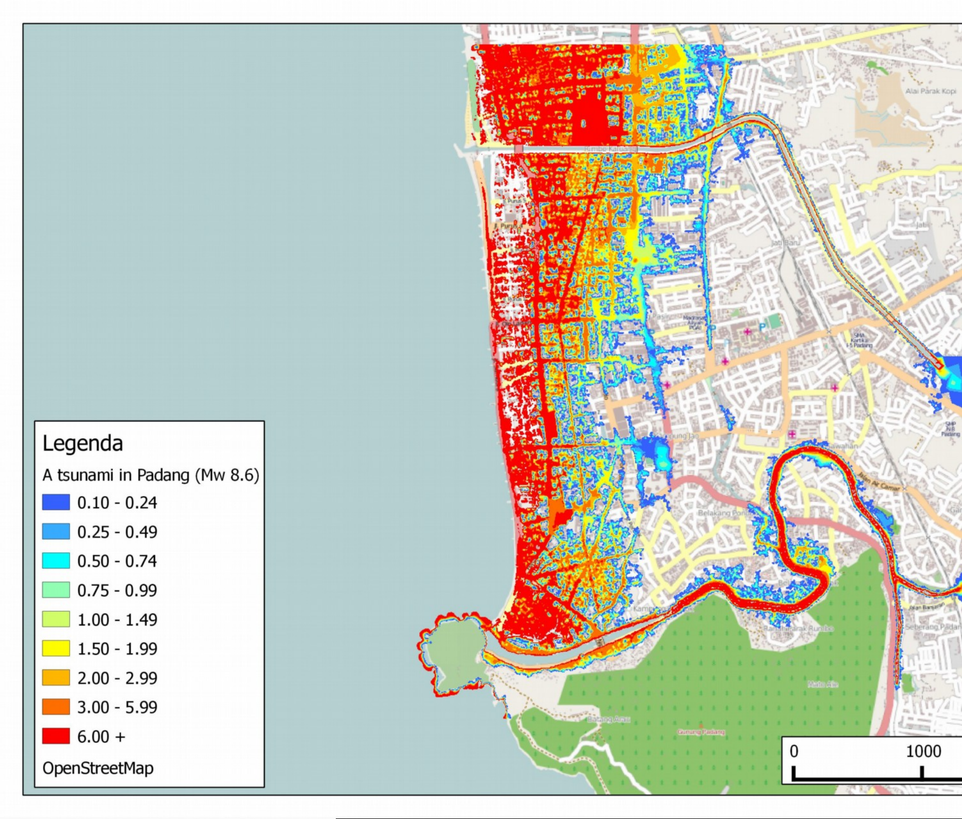
Putting stuff on maps 101







Math vs Maps

Spatial Capabilities

Numerical Algorithms

http://www.qgis.org

Prerequisites

- Familiarity with Python and Numpy
- Familiarity with Linux
- Code is platform independent, but tutorial designed for a Debian/Ubuntu/Mint system.
- Dependencies are: python, python-numpy, python-gdal and qgis

Outline

- Learn to Read and Write (spatial data)
- View and overlay spatial data
- Geoprocessing exercises

Note for this tutorial:

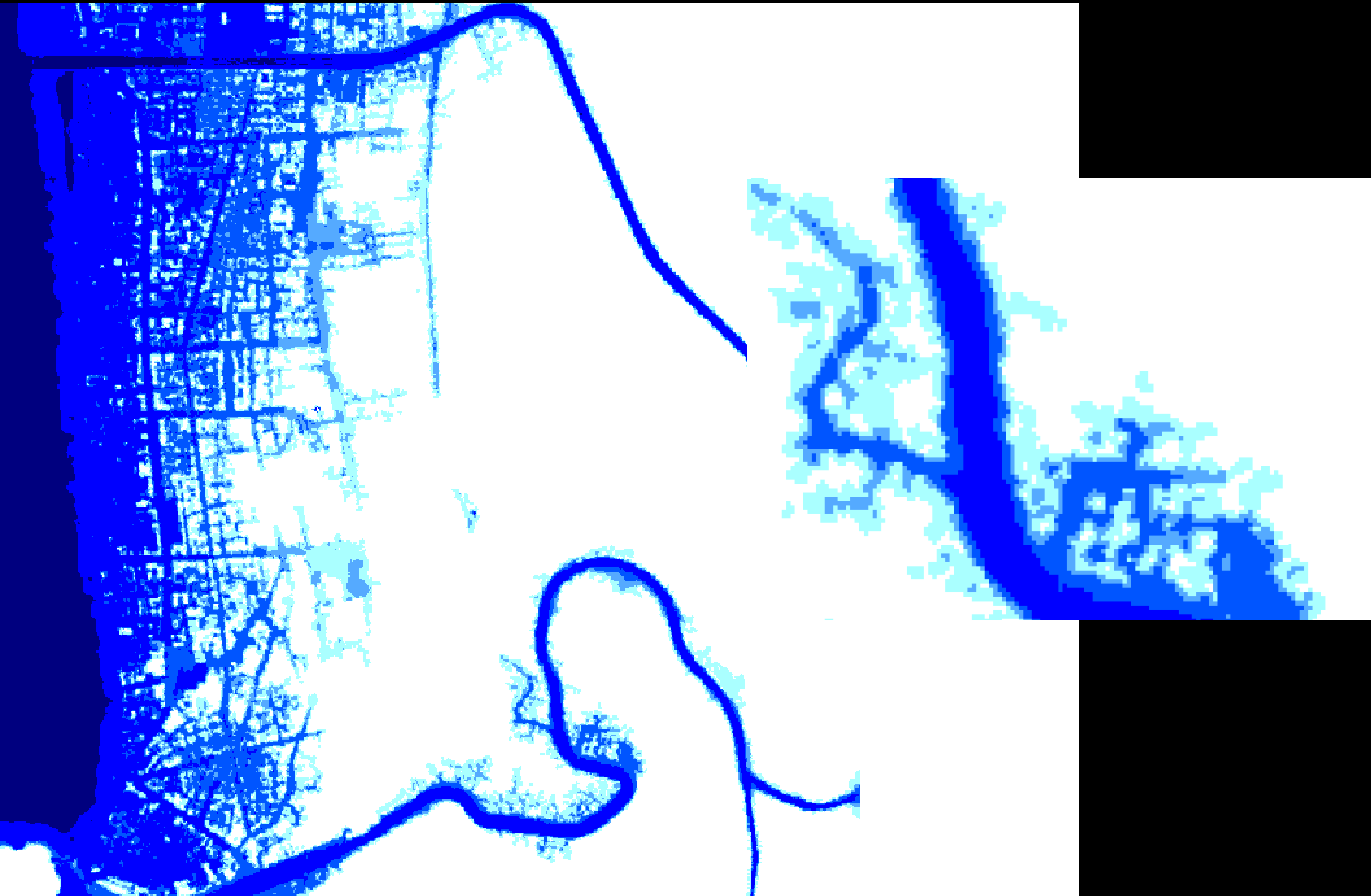
All coordinates in geographic coordinates in datum WGS84.

Tools will work with any spatial reference.

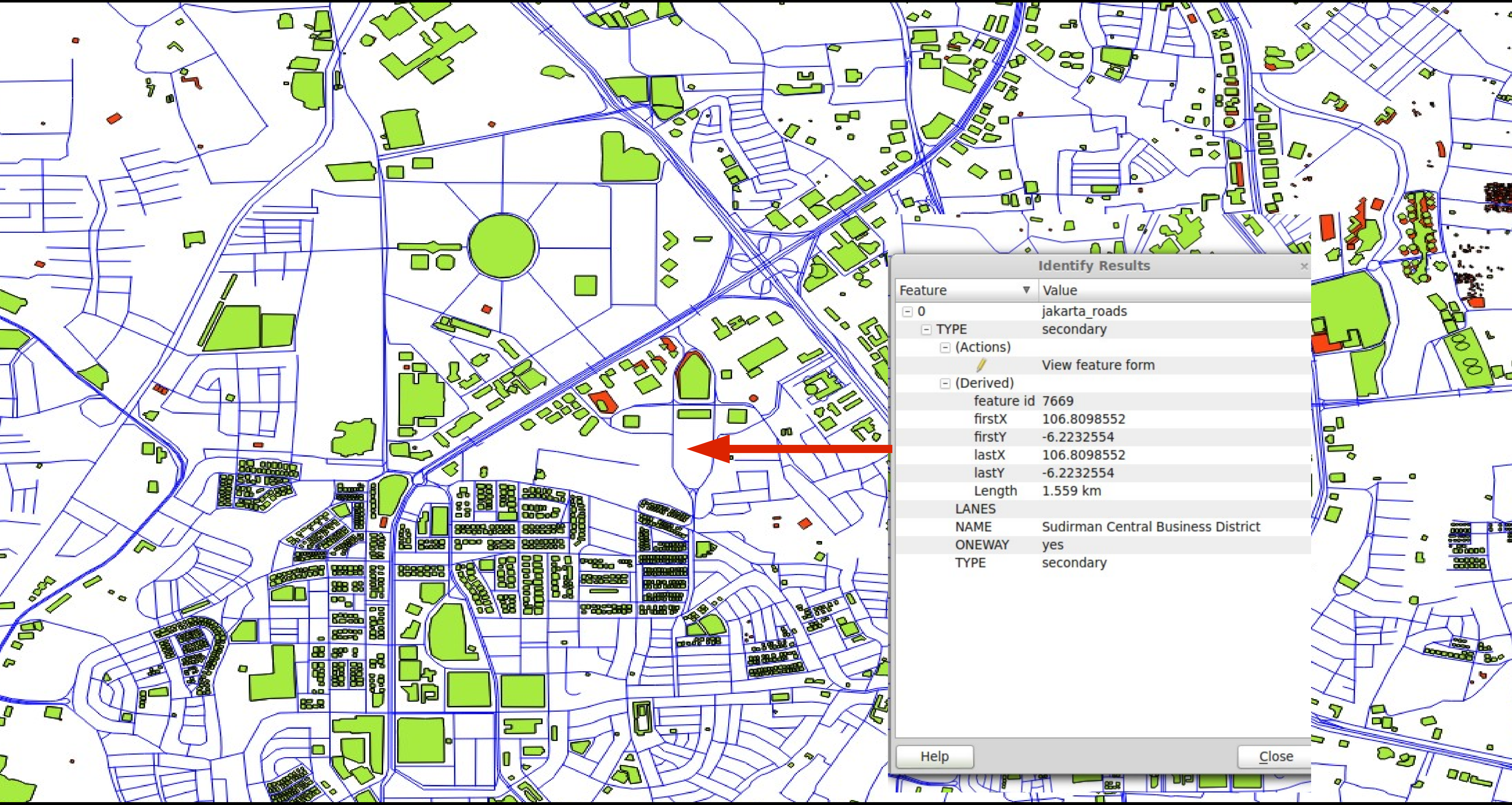
Spatial Information Formats

- Raster Data (Grids)
  - Ascii format
  - GeoTiff
- Vector Data (Points, Lines, Polygons)
  - Shape format
  - KML (Google Earth)

Raster Example



Vector Example



Some GIS tools

- Quantum GIS: Analysis and Visualisation
- GDAL (& Python Bindings): Geographic Data Abstraction Library (Frank Warmerdam)

GDAL doesn't really provide data in a Python/numpy friendly for, so we wrote wrappers around GDAL*:

https://github.com/AIFDR/inasafe/tree/master/safe/storage

* Also included in the tutorial material.

Writing spatial raster data

```
R = Raster(data=A, geotransform=G)
R.write_to_file(<filename>.tif)   # or .asc
```

where

A: 2D numpy array

G: GDAL geotransform (see next slide) defining
    where on earth the grid will be situated.

GDAL Geotransform

The affine geotransform consists of six coefficients which map grid cells into georeferenced space:

- Top left x coordinate
- W-E pixel resolution,
- Rotation (always 0 if north is up)
- Top left y coordinate
- Rotation (always 0 if north is up)
- N-S pixel resolution

Example (in geographic coordinates) with upper left corner at the IIT and a pixel resolution of 0.008333, 0.008333 (approx 1km x 1km):

[72.91645, 0.008333, 0, 19.12543, 0, -0.008333]

Reading spatial raster data

R = read_layer(<filename>.asc)   (or .tif)

A = R.get_data() # Numpy array

G = R.get_geotransform() # GDAL ref

Wrapper can also do

R.get_geometry(): The grid axes - latitudes and longitudes

R.get_resolution()

R.get_bounding_box()

R.get_nodata_value() # Often -9999

R.get_extrema() # Ignoring NODATA value

Writing spatial vector data

V = Vector(geometry, attributes)

V.write_to_file(<filename>.shp)   # or .kml

Geometry: List of points, lines or polygons

Attributes: List of dictionaries of attribute names and values

Exercise 1 will play with this

Reading spatial vector data

V = read_layer(<filename>.asc)   (or .tif)

A = V.get_data() # Attributes

G = V.get_geometry() # Point, line or polygon

Install dependencies

For Debian/Ubuntu/Mint etc:

```
sudo apt-get install qgis
python-numpy python-gdal
```

For Windows and Mac it works too, but I don't know the installation commands

Get The Source

- Open a terminal
- Download tarball from scipy website and unpack

Test the installation

- cd source
- python test_installation.py

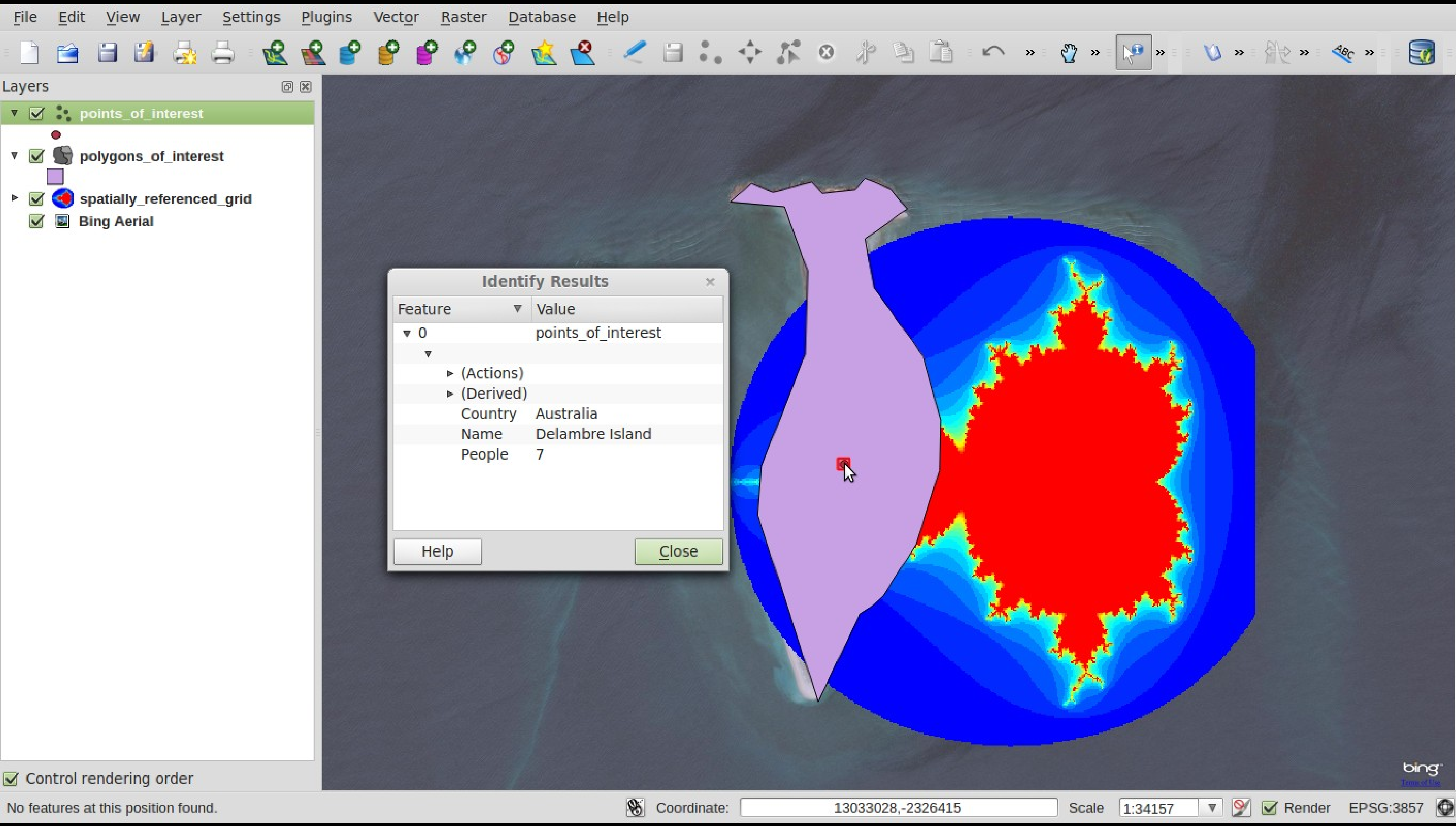
To run exercises (from tutorial root):

- export PYTHONPATH=. (Linux)
- Set PYTHONPATH=. (Windows)
- python exercises/exercise1a.py

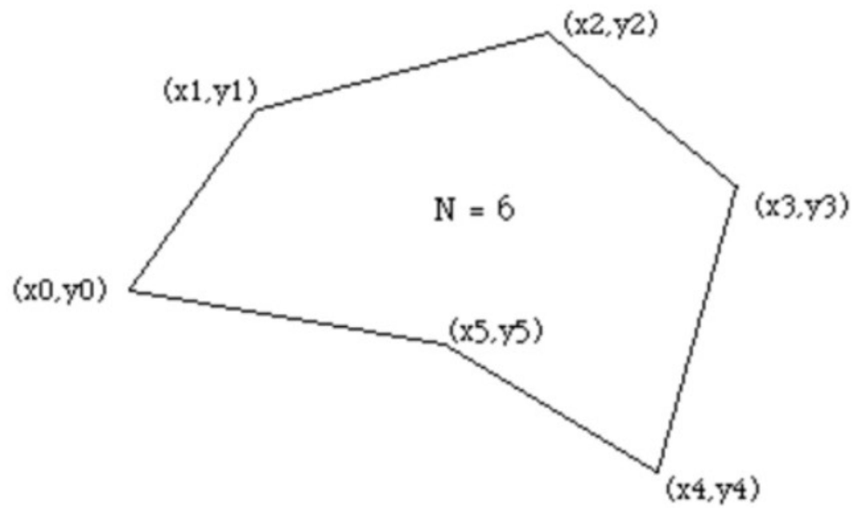
Exercise 1 (a,b,c)

- Read and write spatial data.
- Raster data represented as numpy 2d array
- Vector data represented as
  - List of attributes (on dictionary per feature)
  - List of geometries (point, lines or polygons)

QGIS Screenshot of exercise 1 Data



Exercise 2 – polygon area



The area is given by

$$A = \frac{1}{2} \sum_{i=0}^{N-1} (x_i \, y_{i+1} - x_{i+1} \, y_i)$$

Paul Bourke, 1988

Exercise 3 – use numpy

- If the loop is written in Python it'll be slow.
- Using numpy vector operations can speed things up several orders of magnitude.

Exercise 4 & 5 – Polygon Centroids

$$A = \frac{1}{2} \sum_{i=0}^{N-1} (x_i\, y_{i+1} - x_{i+1}\, y_i)$$

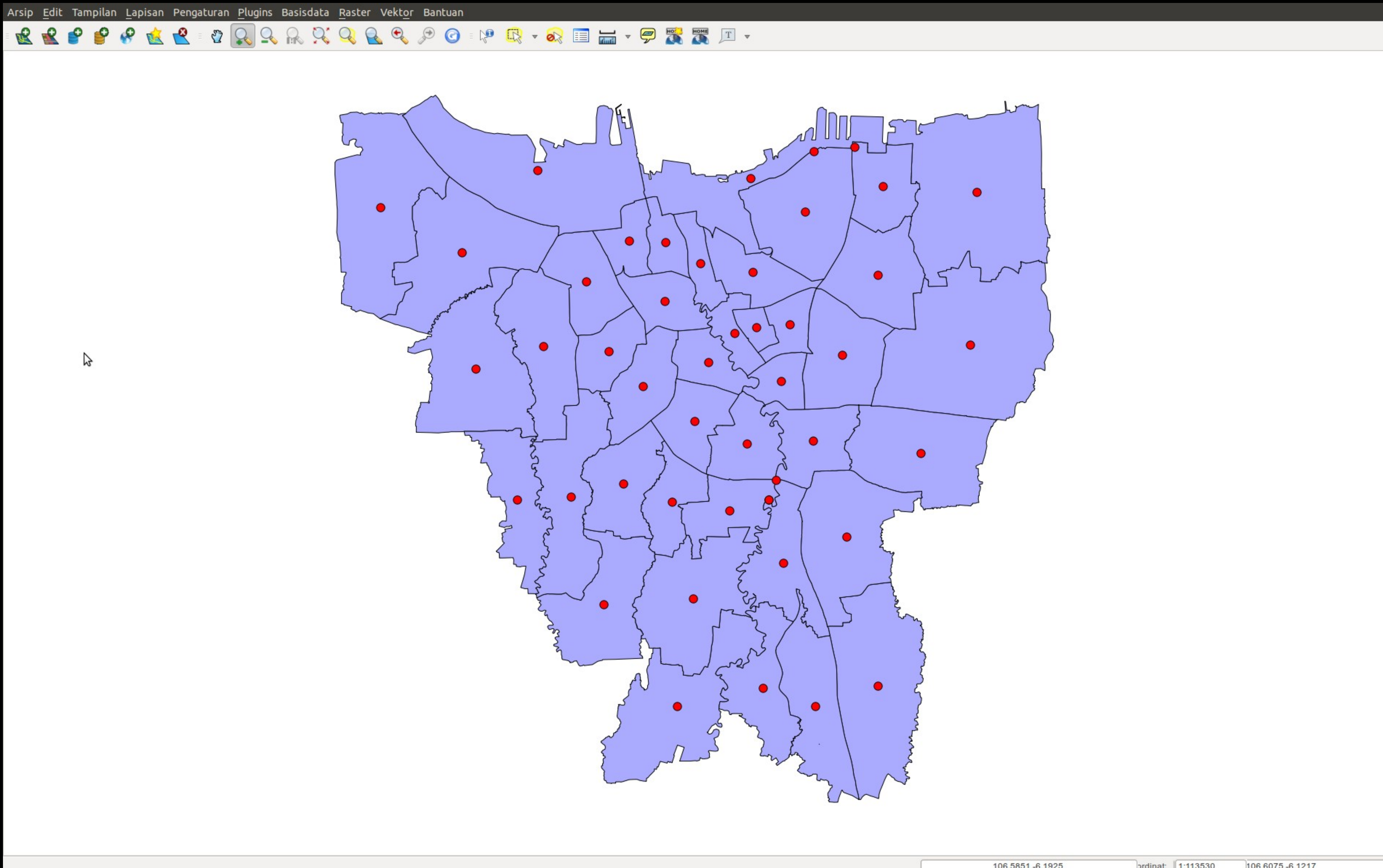
$$c_x = \frac{1}{6A} \sum_{i=0}^{N-1} (x_i + x_{i+1})(x_i\, y_{i+1} - x_{i+1}\, y_i)$$

$$c_y = \frac{1}{6A} \sum_{i=0}^{N-1} (y_i + y_{i+1})(x_i\, y_{i+1} - x_{i+1}\, y_i)$$

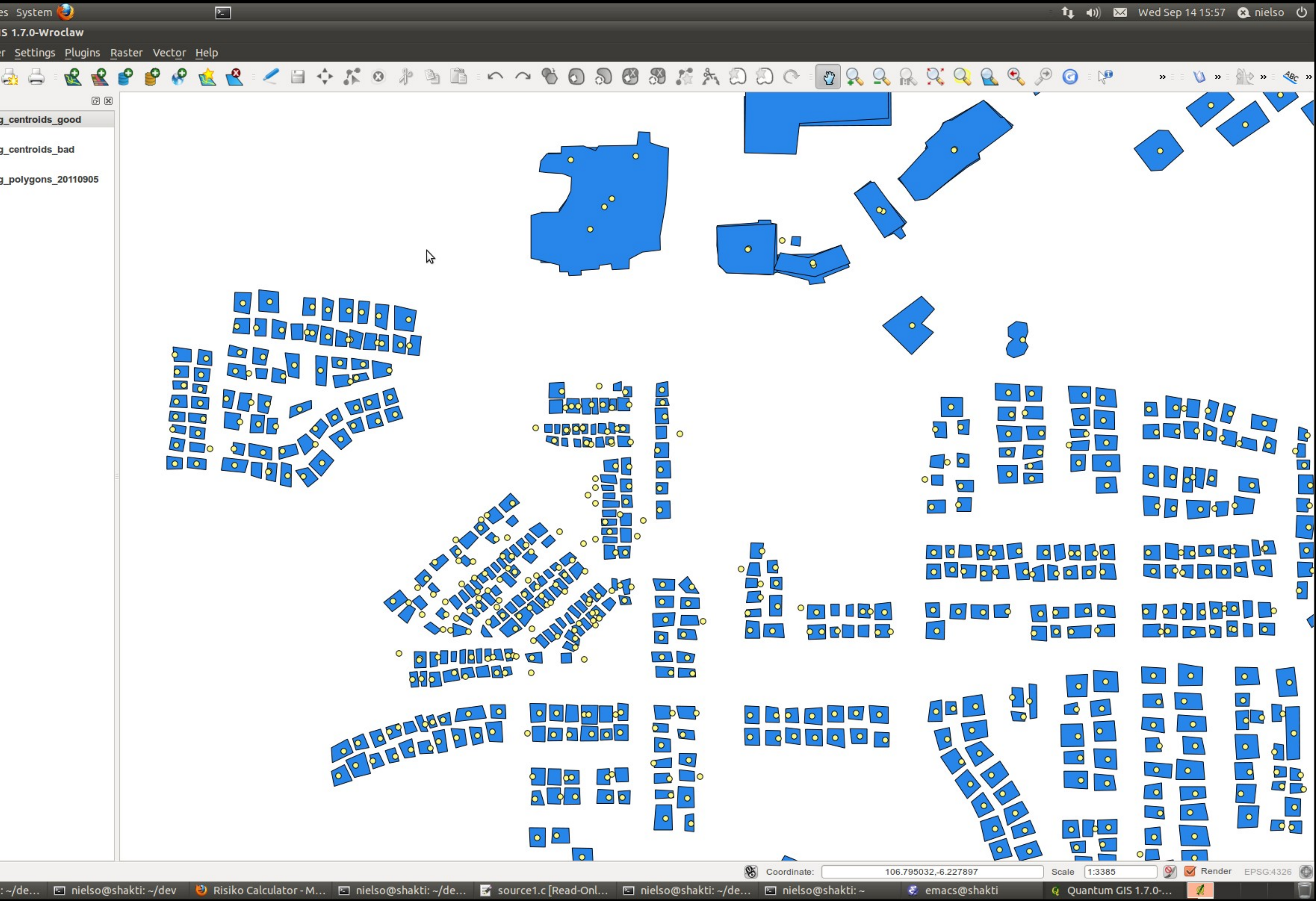
Exercise 5 result

```
Calculated centroids stored, please review with qgis:
qgis ../spatial_test_data/kecamatan_geo.shp
calculated_centroids_kecamatan_geo.shp
../spatial_test_data/kecamatan_geo_centroids.shp
Test 1 passed

Calculated centroids stored, please review with qgis:
qgis ../spatial_test_data/OSM_subset.shp
calculated_centroids_OSM_subset.shp
../spatial_test_data/OSM_subset_centroids.shp
Traceback (most recent call last):
  File "exercises/exercise5.py", line 179, in <module>
    assert numpy.allclose(c_geometry, r_geometry, rtol=1.0e-9), msg
AssertionError: Centroids of OSM_subset.shp were not correct
```


Arsip
Edit
Tampilan
Lapisan
Pengaturan
Plugins
Basisdata
Raster
Vektor
Bantuan
106.5851,-6.1925
ordinat:
1:113530
106.6075,-6.1217

Hower not good for smaller scales



Solution is Normalisation

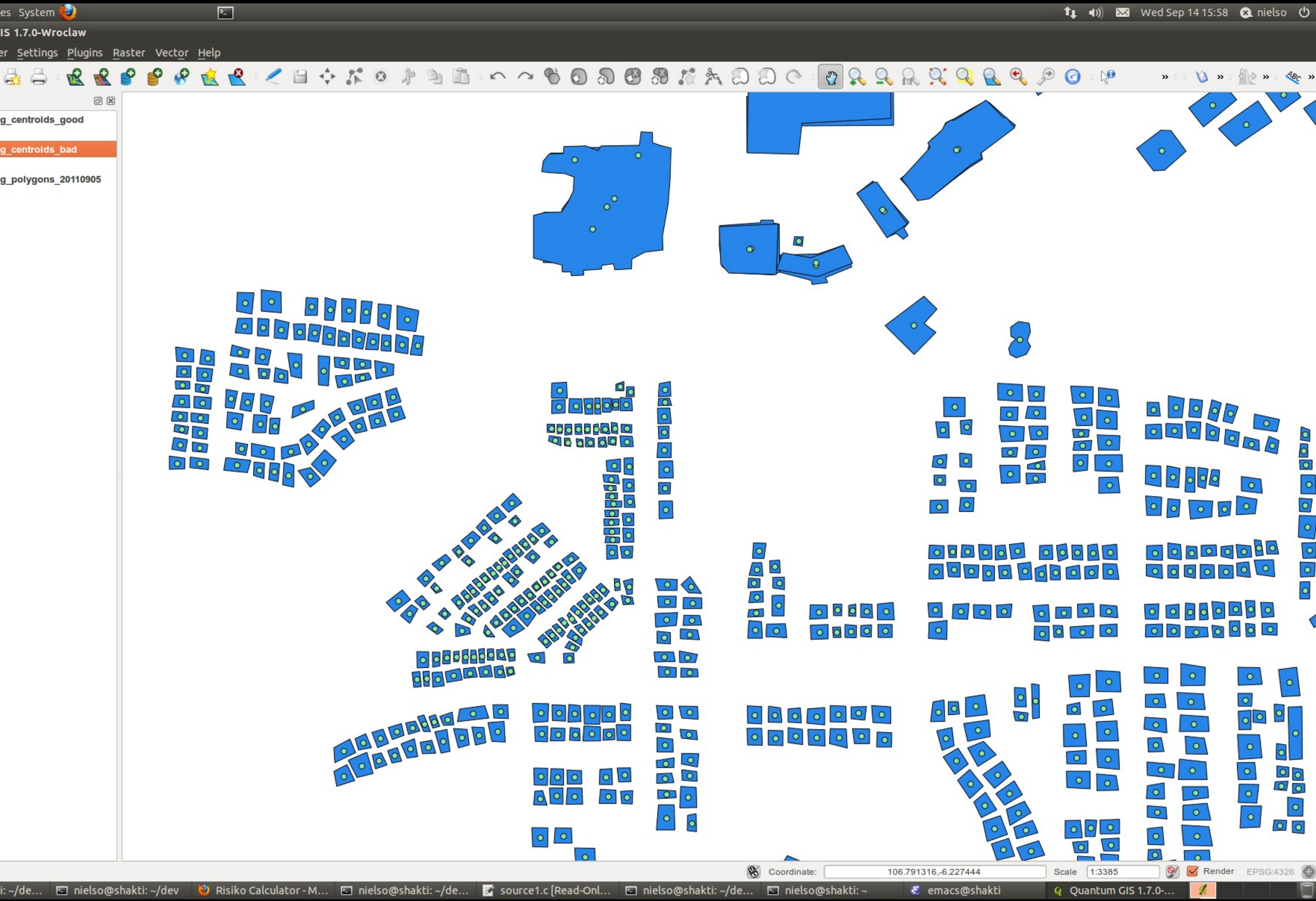
```
# Normalise to ensure numerical accurracy.
# This requirement in backed by tests in test_io.py and without it
# centroids at building footprint level may get shifted outside the
# polygon!
P_origin = numpy.amin(P, axis=0)
P = P - P_origin


# Calculate centroids as usual



# Translate back to real location
C = numpy.array([Cx, Cy]) + P_origin
return C
```

After Normalisation



Exercise 6 – just taking a look

- Numpy implementation of bi-linear interpolation
- Taking NaN into account

Thank you very much!

- The code you have seen was built for the InaSAFE project: www.inasafe.org
- Please have a look at all of it at:

  https://github.com/AIFDR/inasafe