Geospatial Data Analytics for Business 3

Giorgio Chiovelli, Sebastian Hohmann

20/03/2020

Table of contents

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Which factors do we have to take into account?

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Which factors do we have to take into account?

- Demand: where are my customers?

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Which factors do we have to take into account?

- Demand: where are my customers?
  - Key demographics?

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Which factors do we have to take into account?

- Demand: where are my customers?
    - Key demographics?
    - Incomes?

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Which factors do we have to take into account?

- Demand: where are my customers?
  - Key demographics?
  - Incomes?
- Where are my comptetitors?

Site selection

- Suppose you run a coffeeshop business.
- You want to open a branch in central London.
- You have a set of potential locations in mind.
- How do you choose the best one?

Which factors do we have to take into account?

- Demand: where are my customers?
  - Key demographics?
  - Incomes?
- Where are my comptetitors?
- Can we integrate customer demand and competitors into one analysis?

Introduction

The plan for today

1) ORS tools

- Installing the tool
- Signing up and getting an API key
- Configuring the tool with API key

2) Demand analysis with isochrones

- Demand based on total number of residents
- Refinement 1: key demographic
- Refinement 2: income-weighted population
- Refinement 3: Tube station passenger numbers

3) Gravity-model of demand

- Taking into account my own demand...
- and my competitors...
- based on distance/time weights!

- In the top menu-bar, navigate to `Plugins` → `Manage and Install Plugins...`
- Search for "ORS Tools"
- This tool allows you to use most of the functionalities of openrouteservice.org, based on OpenStreetMap, right from within QGIS
- Install the plugin
- Before we can use the plugin, we have to create a free account and get an API key

- Steps for how sign up are described here: `https://github.com/nilsnolde/orstools-qgis-plugin/wiki/ORS-Tools-Help`
- Specifically, go to `https://openrouteservice.org/sign-up/` and create an account
- Sign in to the account, go to Dashboard → `Request a token` → `Token type:` Free, `Token name:` whatever
- copy the API key

ORS Tools

Configuring the tool

- In the top menu bar, go to `View` → `Toolbars` and make sure that `Web toolbar` is checked
- Click on to start ORS tools
- Make sure that “openrouteservice” is selected as `Provider`
- Click on the gear icon , if necessary, click on next to **openrouteservice** to open the possibility to enter the API key, enter your API key and make sure that `https://api.openrouteservice.org` is selected as base URL.
- Once you have entered your API key, click `Close`.

Defining the inputs

- On the google drive you will find two candidate store locations in central London. They are stored in the in the shapefile `candidate_shops.shp`
- We have also given you population numbers (and other information) for LSOA areas based on the 2011 census. These are stored in the shapefile `_demand_points_census.shp`
- The latter are point features – they correspond to the centroids of each LSOA area in central London

Demand analysis with isochrones

Demand based on total number of residents

Running demand analysis with a total number of individuals

- Open the Graphical Processing Modeller ( )
- Open the model `_isochrones_demand_analysis.model3` that we have provided for you
- This model performs a bunch of tasks (we will go through it together)
  1. It finds "isochrones" (areas around a point with common travel time) for the two candidate shops.
  2. It spatially joins the isochrones to the demand points.
  3. Drops a bunch of fields.
  4. Changes field types so we can sum total demand.
  5. Creates a unique store-isochrone ID.
  6. Calculates total demand inside each store-isochrone.
  7. Deletes duplicate store-isochrones.
  8. Keeps only the fields relevant for our decision.
- We are left with the total demand inside each store-isochrone and can now compare the store locations

Demand analysis with isochrones

Inspecting and refining the isochrone analysis

Inspecting the output

- zoom to full extent (clicking on )
- You see isochrones overlapping each other as well as a bunch of demand points. Uncheck the ones with larger numbers to see what is going on
- Inpect the attribute table of `totals` (to get intuition of how the intersection works) and `final` (final output with one row of total demand per shop-isochrone)

This was demand based on total resident population

- Can we refine the analysis?
  - Population in key demographic?
  - Total income?
  - Tube station total daily passengers as alternative demand proxy?
- To make this easier, perhaps we can do it with code. $\rightarrow$ python!

Python

A sales pitch

Even if you don't end up using GIS, python may be for you

- Fully fledged programming language, very easy to learn
- Jupyter for seamlessly combining beautifully formatted text and code: reproducible research! (can do symbolic math, so even for theorists!)

  `http://jupyter.org/`
- The "scientific stack": *numpy, pandas, scipy, matplotlib, Scikit-learn,...*
- Very promising (still in alpha, Mac users will have easier time trying it out than Win users:) *pydatatable*
  - `https://www.youtube.com/watch_popup?v=1yTHSxJ4KL8`
  - `https://github.com/h2oai/datatable`

QGIS comes with python

- If you have installed QGIS, you already have python.

The official source

- `https://www.python.org/downloads/`

A nice distribution

- Anaconda (choose the lightweight miniconda)
- `https://docs.conda.io/en/latest/miniconda.html`
- Has most packages you will need if you want to use python for research (with miniconda, type:
  `conda install packagename`
  from the command line)

2.x.y or 3.x.y ?

- The two versions are maintained and widely used in parallel.
- For the basic functionalities we will use, the differences are minimal.
- QGIS 3.12 works in python 3.7 so that is what we will use.

Case sensitive

- `a = 2` is different from `A = 2`

Indentation is syntactically significant

- You don't need to enclose blocks in { } as in C, Java, or terminate blocks with a statement like `end` in MATLAB.
- Indent to start a block, dedent to end it.
- Statements that should be followed by indentation start with a colon ":".

Path names

- Use frontslashes or double backslashes or raw strings (a backslash inside a raw string is just a backslash, otherwise backslash is used to escape special characters).
- `'C:/the/path/to/your/folder'`
- `'C:\\the\\path\\to\\your\\folder'`
- `r'C:\the\path\to\your\folder'`

Windows

- Start → Search → “cmd” → Enter
- Type `python` and hit Enter

Mac

- Search for “Terminal” and open it
- Type `python` and hit Enter

Closing python

- Type `quit()`

We will now switch over to python. The examples we will run are in the file *python_intro.py* on google drive.

Introduction to Python

Interactive mode and normal mode

Interactive mode

- Open python
- You see a prompt >>>
- Type the examples in section 1) - 7) from *python_intro.py* into the command line / terminal.
- In interactive mode, you get immediate feedback for each statement. Previously run statements (such as variable assignments) are kept in memory.

Normal mode

- Write a python script and save it with the ending *.py* in some directory.
- Inside the command line / terminal, type `pwd` to see your current working directory.
- Change to the directory containing the *.py* file by typing `cd path/to/directory`.
- Run the script by typing `python scriptname.py`
- All the commands in the script are executed
- Alternatively, you can use QGIS's **Python Console**: Plugins → Python Console (`Ctrl+Alt+P`).
- Open the code editor with and run the script with

Agricultural suitability in U.S. counties example

Saving Model Builder model to python script

Exporting to Python

- Inside the QGIS Graphical Modeler, click Export as Script Algorithm ( )
- This opens a window of neatly formatted code. Copy all of it, in the main QGIS window open a python console via Plugins → Python console, show the editor with , paste the code and save it to some file name ending in *.py*.
- This will give you a raw python script, which you can edit further to make it more readable and make it run smoothly.

Try to run the script

- Clicking to run our python script produces no output.*
- Let's go into the script to fix that.

*If you run in the command line or in QGIS Python Console. If you run in in the Processing Script Editor, you get the exact same result as the graphical modeller

Agricultural suitability in U.S. counties example

Understanding and cleaning up the script, part 1

What QGIS has produced and what we want

- QGIS gives us the geoprocessing script organized as a **Class**, with all the geoprocessing happening inside the `processAlgorithm` **method**
- It also imports a bunch of modules
- We will make this simpler (forget about OOP and classes) and just keep the core geoprocessing steps and only the module imports necessary

The header

- Remove all the imports, class, and method definitions before the first `alg_params` dictionary
- Before getting to the definition of local variables, enter (adjust the path name to fit your directory structure):

```
wdir = 'C:/Users/se.4537/Dropbox/PoliteconGIS/LBS_2020/MBA/lecture_3/_workflow_2'
```

The other paths then follow as in the script if you unzipped the `gis_data` into a folder of the same name.

Agricultural suitability in U.S. counties example

Understanding and cleaning up the script, part 2

Intermediate outputs

- `qgis:` and `native:` (latter are written in C++, so faster) algorithms can store outputs in memory if we specify `'OUTPUT': 'memory:'`
- other algorithms, such as those of GDAL, require storage of outputs. Here we create a “junk” folder that we can delete after the program has run
- inside our simple program, both types of output are stored in simple variables (the automatically generated script created a hash table to store the outputs but that is unnecessary complexity)

Algorithm parameters are specified via dictionaries

- The basic syntax for running an algorithm is `processing.run('provider:algoname', parameters)`
- The parameters are specified in dictionaries. Useful, since the order of arguments doesn’t matter
- Optional parameters don’t have to be specified in the dictionary; default values are then used

Writing the output

- Tools that produce output with an attribute table can be given a path to a `.csv`-file as the `OUTPUT` parameter.
- Typically, at the end of a script, use this produce final tabular output of a geoprocessing worklfow.

Agricultural suitability in U.S. counties example

Understanding and cleaning up the script, part 3

Processing tools

- It is a good idea to comment out all geoprocessing commands in the beginning, run the script; then (if it works), successively un-comment one geo-processing command after another.
- As geoprocessing scripts can run for a while, it is a good idea to insert messages using `print('xyz')`.
- The parameter-dictionary of geoprocessing tools translate the menu-structure of QGIS into python scripting language. Essentially, every field you fill out in the QGIS GUI corresponds to a key in the dictionary.
- To find out how to specify the parameters for a given tool, type the following:

```
for alg in QgsApplication.processingRegistry().algorithms():
    print(alg.id(), "->", alg.displayName())
```

- This gives the full list of algorithms: their IDs and display names. To get information on a particular one, type

```
processing.algorithmHelp("provider:algoname")
```

For a list of coordinate systems and their codes, see:
http://resources.arcgis.com/en/help/arcgis-rest-api/index.html#/Using_spatial_references/02r3000000qq000000/

Gravity model of demand

The basic idea

We want to take into account both customer demand and presence of competitors

- How can this be done in a non-arbitrary way?
- Answer: **Gravity Model**

Suppose we try to model some notion of "demand" for our shop

- Let's index our candidate shop with i
- Demand for the shop comes from all potential consumers (suppose there are n of them):

$$
\begin{aligned}
D_i &= d_i^1 + d_i^2 + \cdots + d_i^n \\
&= \sum_{j=1}^{n} d_i^j .
\end{aligned}
$$

Gravity model of demand

Developing the basic idea (1)

What does d_i^j depend on?

- Total expenditure of potential consumer j
- Distance to our shop
- ... relative to our competitors!

Let's stick with the first two for now. Letting τ_{ij} denote the distance between our shop i and consumer j and y_j the income of consumer j, we can write

$$
\begin{aligned}
D_i &= \frac{y^1}{\tau_{i1}} + \frac{y^2}{\tau_{i2}} + \cdots + \frac{y^n}{\tau_{in}} \\
&= \sum_{j=1}^{n} \frac{y^j}{\tau_{ij}}.
\end{aligned}
$$

Which assumptions are embedded here?

Gravity model of demand

Developing the basic idea (2)

What about our competitors?

- Once we have entered with our shop, every customer is faced by a choice between K (including our own shop) competitors
- Consumer j faces a "pull of attraction" to our shop equal to $\frac{1}{\tau_{ij}}$.
- Similarly, she faces a pull of $\frac{1}{\tau_{kj}}$ to shop k
- And a total pull of $\frac{1}{\tau_{1j}} + \frac{1}{\tau_{2j}} + \cdots + \frac{1}{\tau_{Kj}} = \sum_{k=1}^{K} \frac{1}{\tau_{kj}}$ to all K shops (including our own potential entrant)
- If she made coffee-buying decision probabilistically (a useful simplification), the fraction of demand of consumer j coming our way can written as $\frac{\frac{1}{\tau_{ij}}}{\sum_{k=1}^{K} \frac{1}{\tau_{kj}}}$
- So we can write demand of consumer j for our product as $\frac{\frac{y^j}{\tau_{ij}}}{\sum_{k=1}^{K} \frac{1}{\tau_{kj}}}$
- And finally total demand as $D_i = \sum_{j=1}^{n} \frac{\frac{y^j}{\tau_{ij}}}{\sum_{k=1}^{K} \frac{1}{\tau_{kj}}}$.

Gravity model of demand

Developing the basic idea (2)

What about our competitors?

- Once we have entered with our shop, every customer is faced by a choice between K (including our own shop) competitors
- Consumer j faces a "pull of attraction" to our shop equal to $\frac{1}{\tau_{ij}}$.
- Similarly, she faces a pull of $\frac{1}{\tau_{kj}}$ to shop k
- And a total pull of $\frac{1}{\tau_{1j}} + \frac{1}{\tau_{2j}} + \cdots + \frac{1}{\tau_{Kj}} = \sum_{k=1}^{K} \frac{1}{\tau_{kj}}$ to all K shops (including our own potential entrant)
- If she made coffee-buying decision probabilistically (a useful simplification), the fraction of demand of consumer j coming our way can written as $\frac{\frac{1}{\tau_{ij}}}{\sum_{k=1}^{K} \frac{1}{\tau_{kj}}}$
- So we can write demand of consumer j for our product as $\frac{\frac{y^j}{\tau_{ij}}}{\sum_{k=1}^{K} \frac{1}{\tau_{kj}}}$
- And finally total demand as $D_i = \sum_{j=1}^{n} \frac{\frac{y^j}{\tau_{ij}}}{\sum_{k=1}^{K} \frac{1}{\tau_{kj}}}$. PHEW!!!

Gravity model of demand

Computing the solution in python

Inspect the data

- Load the shape with candidate and Nero coffee shops (`shops_nero_candidates.shp`) and the tube station shapefile (`_demand_points_tube.shp`) to the canvas
- inspect the data visually and look at the attribute tables

Computing the gravity-solution in python

- Open QGIS's **Python Console**: Plugins → Python Console (`Ctrl+Alt+P`).
- Open the code editor with , load the script `gravity.py` we have provided ( ), adjust the path names to fit your directory structure, and run the script with