TMM Lab 01: Demand Modelling

RE.Wilson@bristol.ac.uk

26 September 2025

Getting started with geography

There are some basic capabilities you need to build up before we can plough into various computations. Because they are basic building blocks, not ends in themselves, we are not too bothered how you build up these things, e.g., from

- Built-in Matlab functions (which seem to mushroom in each new version)
- Matlab central file exchange / similar
- ChatGPT / internet snippets
- Write yourself, from the ground up (slow, but the best for your personal development)

1. Basic wants:

   (a) Code that finds the distance between two points given as lat, long coordinates. (NB this is about the only occasion ever I am ok with you using degrees.) Even better if your code is vectorised and can (for example) take e.g., lists of such points and return a matrix of the distances between them. Google (for example) the Haversine formula.

   (b) Code that converts lat, long into 12 digit Ordnance Survey grid references — which makes the distance computations easy. NB 12 digits is 6 for Easting and 6 for Northing — which thus form $x - y$ coordinates with units of metres with an origin in the far South West of the UK. Again — vectorised solutions would be nice.

   (c) Wouldn't it be nice if sets of coordinates etc. could be plotted nicely over a map of the UK so we knew where on Earth (literally) they were? Shurely someone has written some contrib code for that / is it in a Matlab toolbox?

   (d) If given a large set of lat-long coordinates, compute and plot a Voronoi tesselation of the map with these coordinates as centroids. Build some kind of logical (graph) structure — e.g., an adjacency matrix which says which of the tesselated areas are neighbours with each other. Note that we have no plans to use *shapefiles* in this unit, so such adjacency might not faithfully replicate whether two localities truly share a boundary. (*)

   (e) (**) Find about shapefiles. How to deal with them in Matlab / other software e.g., ArcGIS or similar. Download and view shapefiles for MSOAs, e.g., search for them at either

      * `https://www.data.gov.uk/`
      * or the ONS Open Geography portal `https://geoportal.statistics.gov.uk/`

Basics of the census data

We are just going to look at the basic version of the census data exactly as processed in Anders's videos. See the file `UK_census_msoa_stats.csv` which Anders has prepared and which is attached on Blackboard. Again — this is about testing basic building blocks and developing / testing some more advanced coding skills, as well as looking at properties of the data itself.

2. Load in `UK_census_msoa_stats.csv`. We recommend you use Matlab's data import tool (get fluent with it). Check it allocates sensible variable types to each field (column). You might want to consider whether some variables should be categorical (learn it if you don't know what that means) and maybe integer variables should be cast as integer types such as `uint32` etc. We recommend you use a `table` to collect the imported data (in which case you might need to learn some of the syntax for working with tables in Matlab) — but other solutions are possible. Get used to saving the code from the data import tool and inspecting / modifying it.

   (a) Verify the linear correlation between total origin data (What Anders calls $O(i)$) and population in the respective MSOA. Use e.g., simple scatter plots initially. Then use one of Matlab's built in linear regression tools.

   (b) Relationships between $O(i)$ and other variables provided for MSOAs, and between themselves — compute, discuss, including errors and significance. E.g., numbers of households, area (hectares), Townsend deprivation score (lookup *indices of (multiple) deprivation*), and the destination data $D(i)$. (*) If you've done the Eng Maths Applied Statistics unit, you should know about (linear) regression with multiple input variables — if you haven't, aim to learn a bit of this stuff for overall life toolkit.

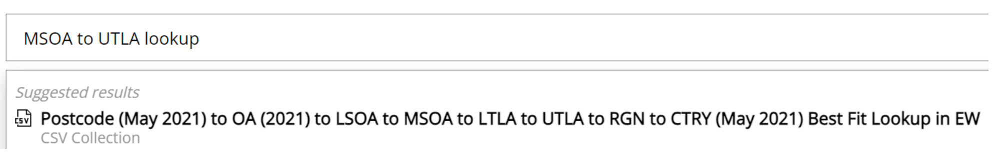
   (c) Demonstrate that $D(i)$ really isn't well predicted by any of the other variables here.

   (d) Try to aggregate the $O(i)$ and $D(i)$ data to show totals of origin and destination data for larger geographies. Some possible ways of going about this:

      (i) Is the sort-of string part of the MSOA name the name of the UTLA? (For example, at the top of the file, is *Barking and Dagenham* a UTLA or some other useful geography?). Obviously, this will need some string processing fun. Alternatively, can the MSOA codes be processed or understood in some way?

      (ii) ** (Maybe hard.) Attempt some kind of clustering more like what Anders does — either some kind of lumping of near neighbours (there must be countless ways of doing this) or maybe even crudely just according to large square grid cells applied over the whole of the map.

      (iii) * Systematically — use lookup tables from the Open Geography Portal.



      looks like a big file that might be useful? And a joy in its own right to play with. Note that there may be some issues as I am not completely clear which census the Anders's $O(i)$ and $D(i)$ data file in this question is from. Boundaries do change — although I think things are quite stable at the MSOA level and up.

   (e) Lighter activities if you are getting a bit tired:

      (i) Investigate / invent fun ways of getting some of the above results onto map plots.

      (ii) Have a hunt around the Open Geography portal and `data.gov.uk` for other fun administrative data. There is a lot out there. Check out `https://www.adruk.org/`. For your wider transport knowledge — one thing you should definitely check out is the National

Travel Survey (NTS)
`https://www.gov.uk/government/collections/national-travel-survey-statistics`

Trip data

We are now onto the main point: Working with the full Origin-Destination data which actually gives the numbers of people that go from A to B for all combinations A and B.

3. Get hold of `ODWP01EW_MSOA` from Blackboard and import it into Matlab. This is a very big file and might cause your computer to die. Sorry. If this really is causing issues, I can provide you the equivalent data for a coarser geography (e.g., local authorities) — or you can be resourceful and get it from the ONS
`https://www.ons.gov.uk/census/aboutcensus/censusproducts/origindestinationflowdata`.
Screenshot of the first few lines viewed in Excel:

A	B	C	D	E	F	G
Middle layer S	Middle layer Super C	MSOA of workplac	MSOA of workplace label	Place of work indicator	Place of work indicator (4 categories) label	Count
E02000001	City of London 001	-8	Does not apply	-8	Does not apply	2653
E02000001	City of London 001	999999999	Workplace is outside the UK	2	Other (including offshore installation, working outside	35
E02000001	City of London 001	E02000001	City of London 001	1	Mainly working at or from home, No fixed place	3871
E02000001	City of London 001	E02000001	City of London 001	3	Working in the UK but not working at or from home	436
E02000001	City of London 001	E02000016	Barking and Dagenham 015	3	Working in the UK but not working at or from home	2
E02000001	City of London 001	E02000024	Barnet 001	3	Working in the UK but not working at or from home	3
E02000001	City of London 001	E02000027	Barnet 004	3	Working in the UK but not working at or from home	1
E02000001	City of London 001	E02000055	Barnet 032	3	Working in the UK but not working at or from home	1

The pair of columns on the left describe the origin MSOA and the next pair of columns describe the destination MSOA — for morning commuting trips. The column on the far right is the number of trips in units of people. You will note there is not a row for every MSOA–MSOA combination. This is because many combinations have zero trips between them and so are not reported.
Wish list of tasks — (a–d) essential.

(a) Get the OD counts actually into an OD matrix (which will be several thousand rows by several thousand columns). Because so many of the entries are zero, you might want to use Matlab's *sparse* matrix type to do this.

(b) Make sure you keep a lookup table so you know the correspondence between row or column index and the name / code for the MSOA.

(c) Compute the row and column totals (ie the total origin and destination flows for each MSOA) and keep those safe. NB they might not be the same as in the file we were working on Q2 as I am not sure that that is for the same census or about other features of how Anders built it.

(d) Compute a (symmetric) matrix of the distance between each OD pair, or strictly speaking, the distance between the centroids of those respective geographies.

(e) Compute distributions (over the whole population) of numbers of people for different commuting distances — although note short commutes will be poorly represented because the centroid of the origin and destination MSOA will not accurately represent where the trip actually started/ended. Also note there will be quite a lot of trips that start and end within the same MSOA. (As we go forward in the course, we will look at ways of synthesising more realistic trips.) Compare with the National Travel Survey.

(f) Redo all of the above for coarser geography eg local authority, but note that gives terrible resolution for short trips.

(g) Can you make plots which resemble the interactive tool on datashine?
`https://commute.datashine.org.uk/`

Modelling trip data

If you have completed Q3 — you should have:

- A matrix of actual realised trip flows T_{ij} from place i to place j.
- A matrix of distances d_{ij} from place i to place j.
- A vector of flows O_i that total all the trips with origin at place i — which is the row sums of the matrix T_{ij}.
- A vector of flows D_j that total all the trips with destination at place j — which is the column sums of the matrix T_{ij}.

The point now is to model the observed T_{ij} in terms of O_i, D_j, and d_{ij}. The basic form of (unconstrained) model takes the form

$$T_{ij} \simeq cO_iD_jf(d_{ij}),$$

where c is a constant and f is a decreasing function.

4. Seek fits of the classic unconstrained gravity model with $f(d) = 1/d^2$. Hint: build an $n^2 \times 2$ matrix which unravels T_{ij} and compares with O_iD_j/d_{ij}^2. Do a scatter plot of one column against the other. Seek a straight line fit which will yield c. Note your methods might need to have special ways of putting less weight on the very many zero or small values in T_{ij}.

5. Try fitting with different inverse powers of d, e.g., $1/d$, see Anders's video.

6. For the classical gravity model $f(d) = 1/d^2$ and for best fit c:

   (a) Compute the predicted flows cO_iD_j/d_{ij}^2, and verify that the row sums do not correctly add up to O_i nor do the column sums correctly add up to D_j.

   (b) Hence by normalising each row by a constant α_i so that the row sum gives the correct value O_i, compute a singly-constrained model OD matrix. Verify that the column sums do not add up to the correct values D_j. (You could also attempt this question the other way round, by constraining the column sums with constants β_j and noting that the row sums do not give the correct values.)

7. For the classical gravity model, attempt to construct the doubly constrained model for the OD matrix. That is, see constants α_i (that multiply rows) and β_j (that multiply columns) so that both row and column sums add to the correct quantities. Note therefore that each individual entry i, j is multiplied by $\alpha_i\beta_j$. To do this:

   (a) Formulate a nonlinear system of equations to solve for the $2n$ constants. Solve it either using your favourite Matlab built-in solver, or research hand-crafted iterative methods in the internet. (Note: there is a very similar piece of mathematics involved in correcting sample biases in market research / surveys —– google RIM weighting.)

   (b) Verify that the scaled entries now have the correct row and column sums.

   (c) Investigate how well the modelled OD data fits T_{ij}.

8. Investigate fits of other models — e.g., Intervening Opportunities, Radiation model etc. (I will be supplying a sheet of mathematics to underpin this exercise.)