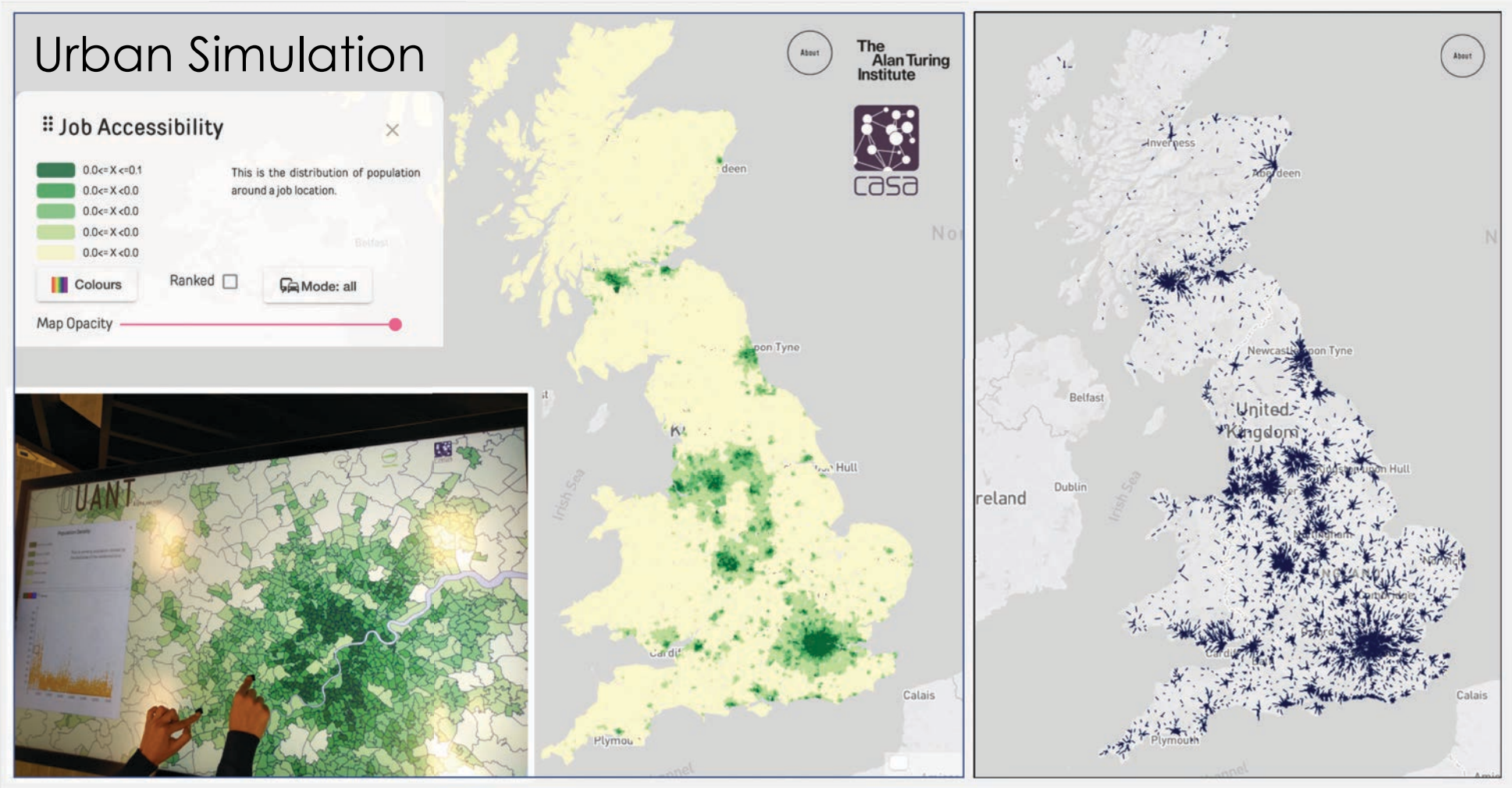
CASA0002: Urban Simulation (2024)



Michael Batty Simulation ● **Elsa Arcaute** Networks

Urban Simulation 2

Spatial Interaction: Basic Concepts

Michael Batty

m.batty@ucl.ac.uk
X@jmichaelbatty

Monday, 15th January 2024

http://www.spatialcomplexity.info/

Outline of Today's Lecture

Revision and Corrections From the First Lecture

- The Matrix Notation : Origins and Destinations
- The Basic Gravitational Model
- Different Distance Functions: Scaling
- Potential and Accessibility
- Ensuring the Model Meets Basic Constraints
- Explaining the Basic Computer Program
- Population Density: The 1- Dimensional Model
- Intervening Opportunities Models

Run out of time – dealt with in the next lecture

- The Family of Spatial Interaction Models
- Urban Models: Coupled Spatial Interaction

Lecture 1 Last Week: Corrections and Revision

The simplest linear relationship is

$$y = a + bx$$

this leads to what is clearly a straight line plot. If we have more than one dependent variable the model becomes

$$y = a + b_1x_1 + b_2x_2 + +b_3x_3 + \cdots$$

$$= \ a + \sum_{k=1}^{N} b_k x_k$$

The simplest nonlinear relationship is

$$y = ax^b \text{ and the log transform is}$$

$$\log y = \log a + b \log x$$

leading to a curvilinear plot and with the transform to a linear plot. If more than one independent variable

$$y = a{x_1}^{b_1}{x_2}^{b_2}{x_3}^{b_3}\ldots.$$

$$y = a\prod_{k=1}^{N}{x_k}^{b_k} = a{x_1}^{b_1}{x_2}^{b_2}{x_3}^{b_3}\ldots$$

$$\log y = \log a + \sum_{k=1}^{N} b_k \log x_k$$

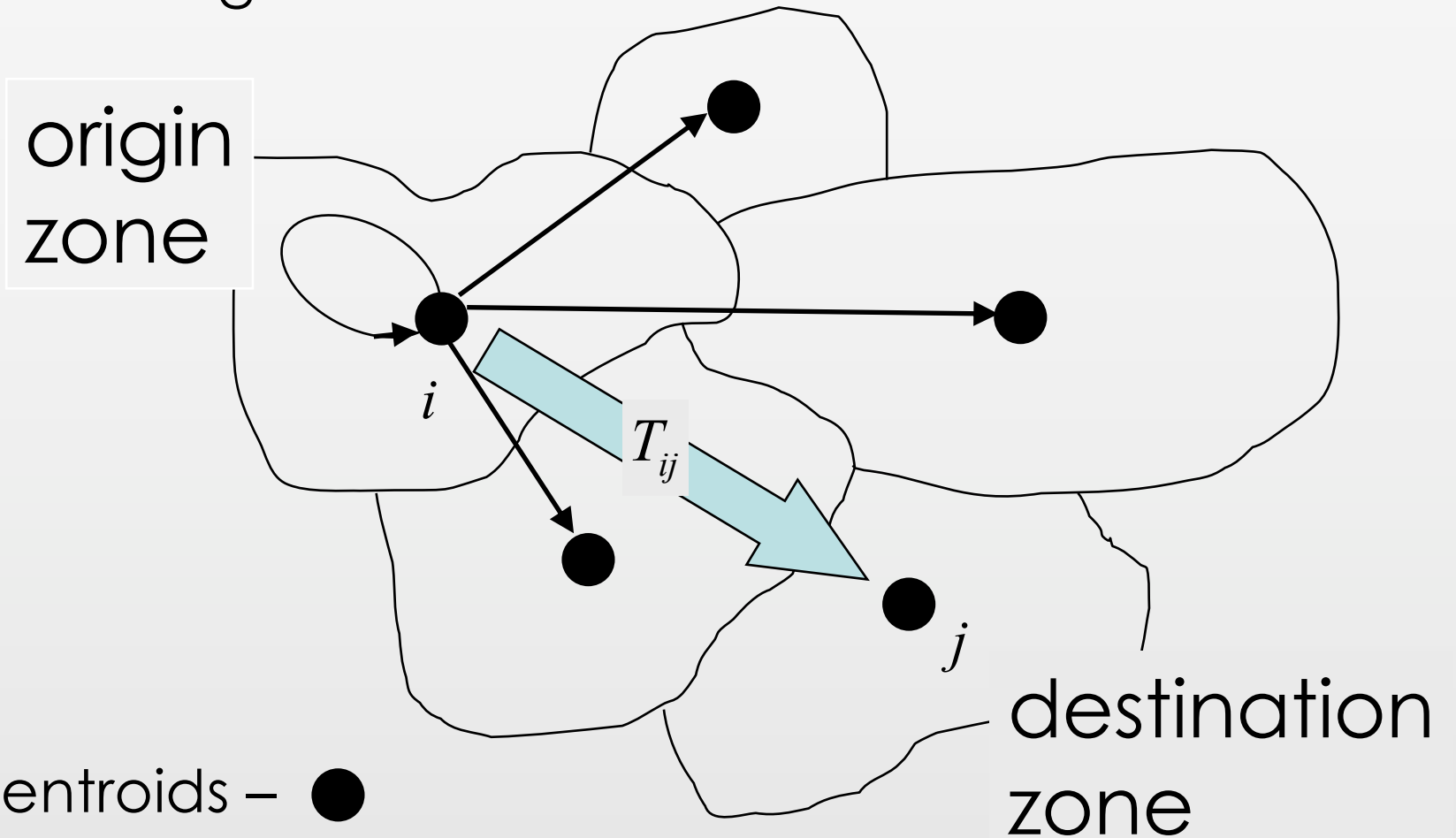
Moving From Theoretical Continuous Models to Discrete

Our models are first written without relating them to the actual system where the basic unit is the zone or spatial unit and what we will do is specify models with respect to their zones based on origins and destinations $i \; and \; j$

$$T_{ij} = T_{ij} \sim \frac{P_i^{\alpha} P_j^{\gamma}}{d_{ij}^{\beta}} = P_i^{\alpha} P_j^{\gamma} d_{ij}^{-\beta}$$

Note the way we represents powers and also the nature of the relationships as a proportion or equality

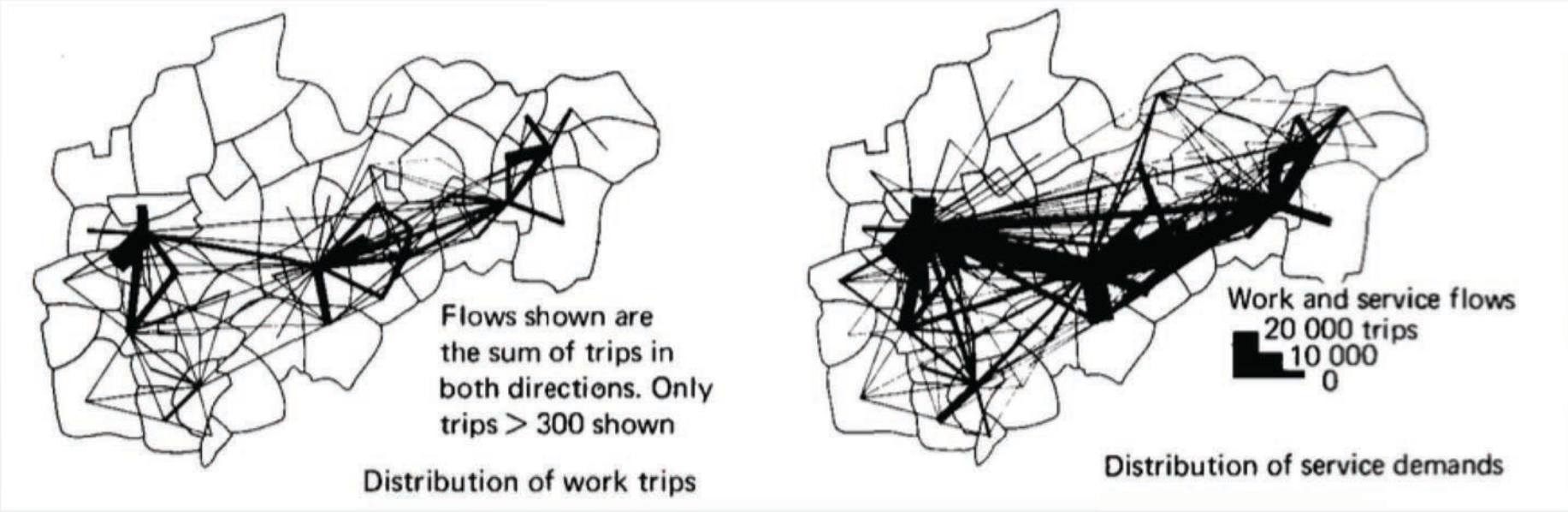
Here is a simple picture of what we are defining and modelling

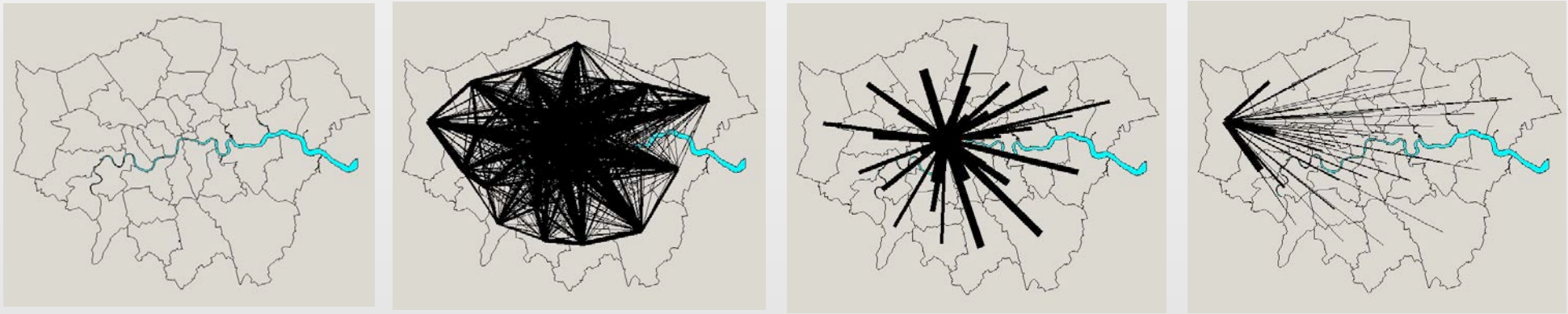


centroids – ●

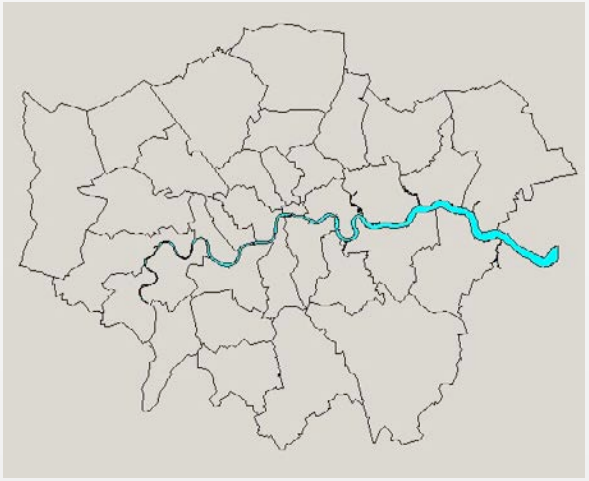
centre of gravity of the zone – approximating the zone by a point

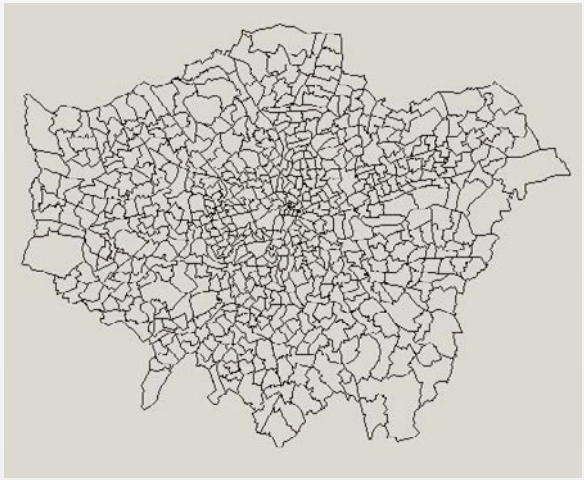
Early examples : Centre and NE Lancashire: London

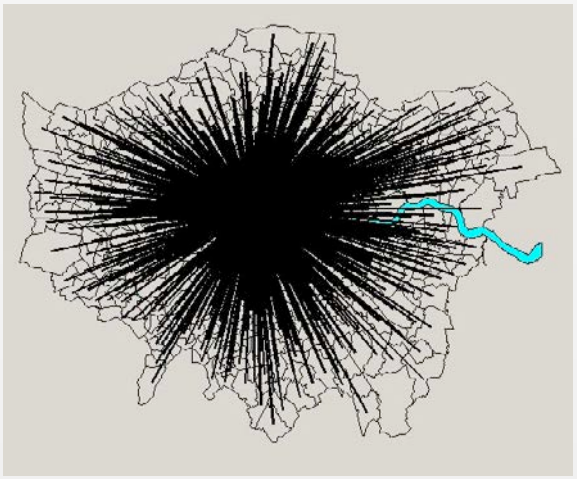


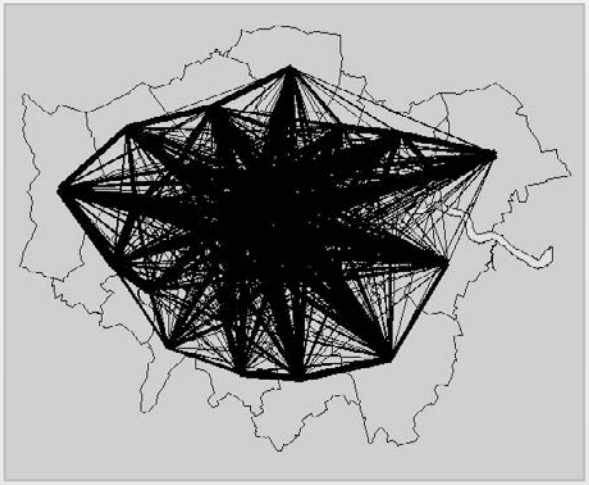


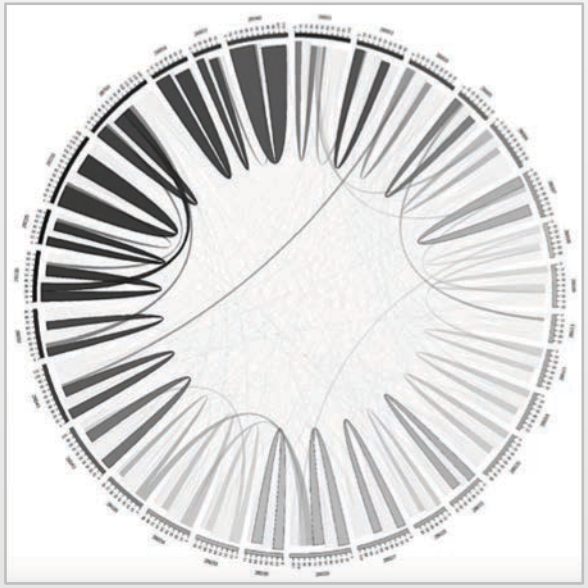
$N^2=33^2=1089$, not so big but hard to visualise

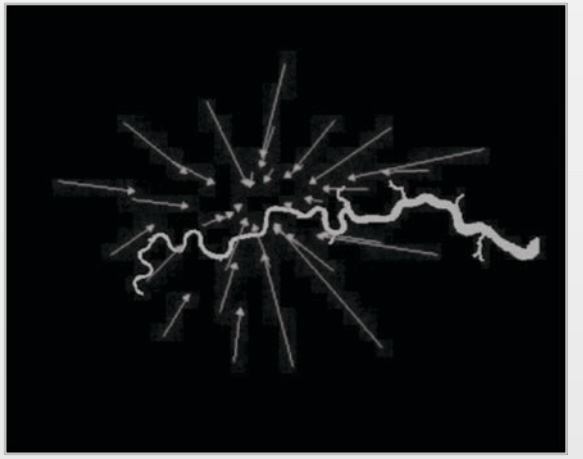






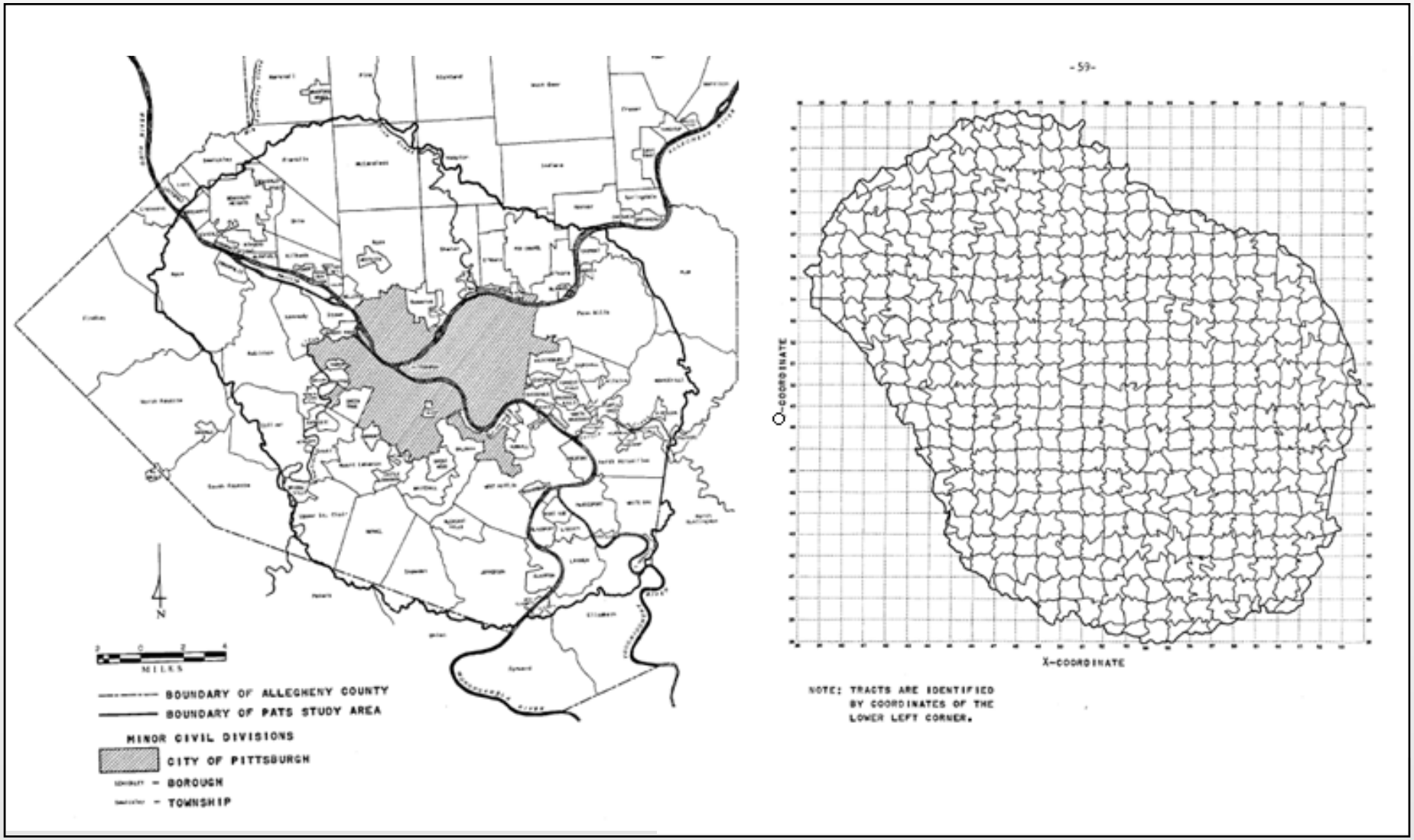




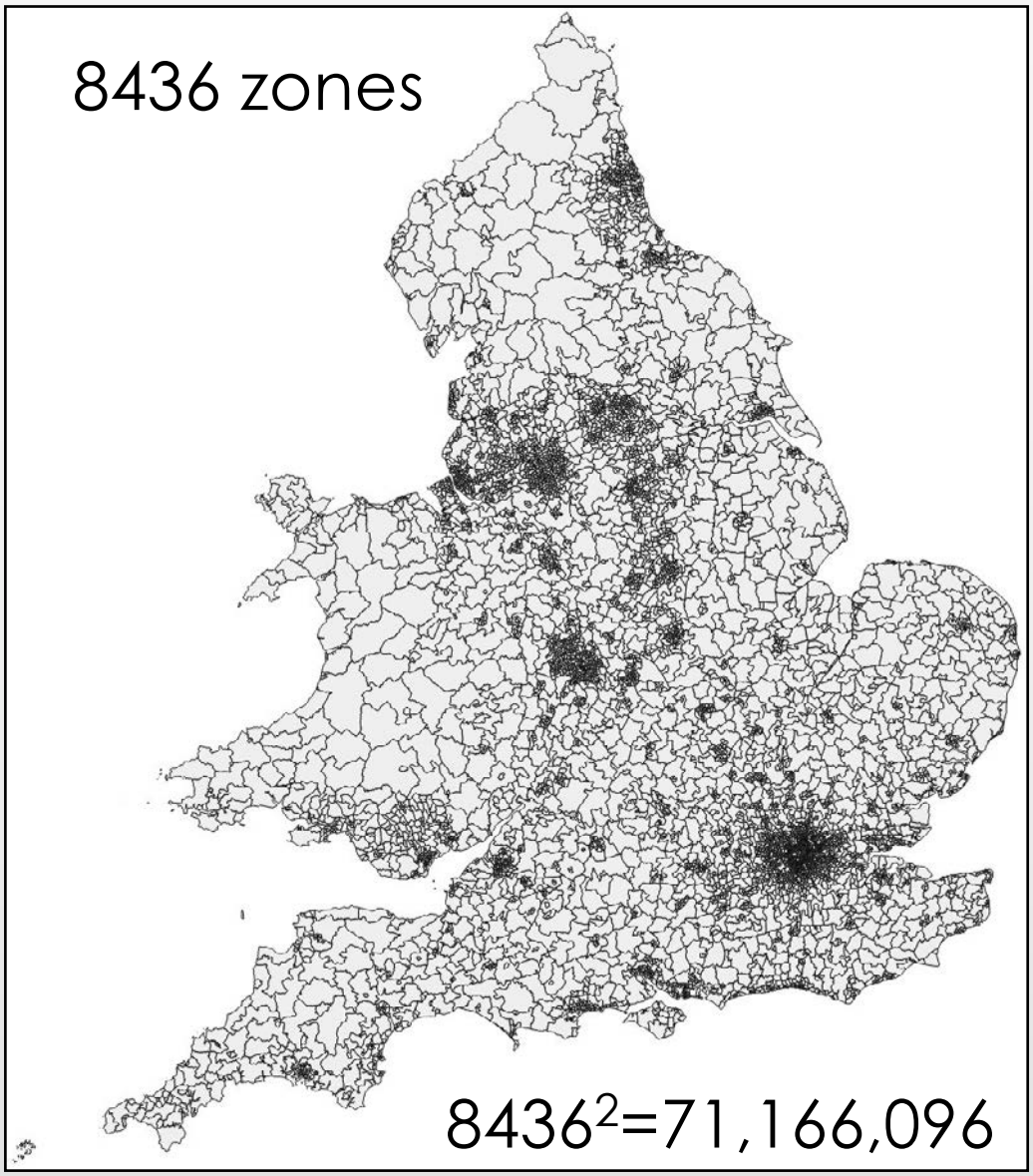


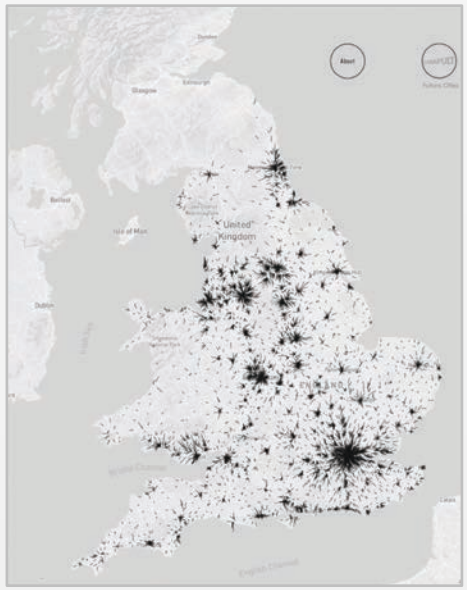
Vectors of flows positioned as average directions of all trips flowing into different destinations

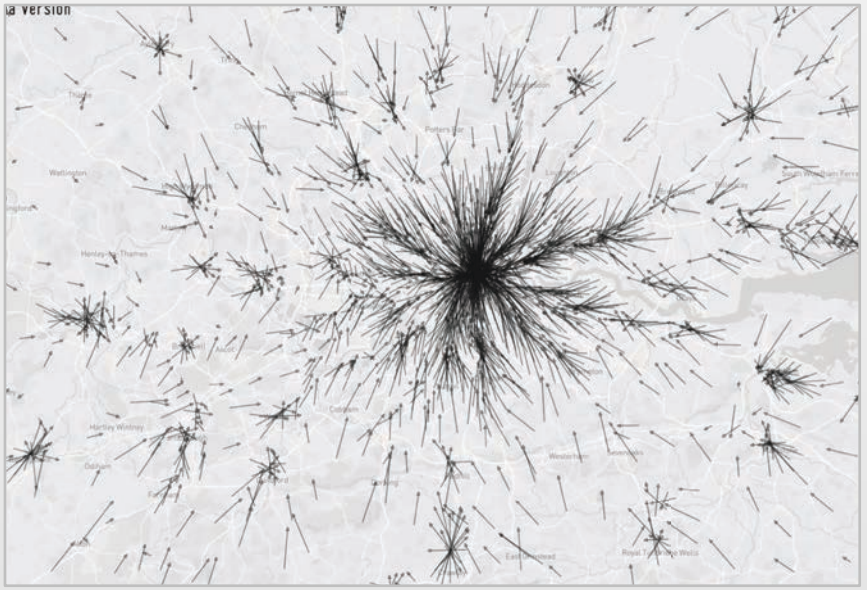
$n^2=633^2=400,689$, bigger but impossible to visualise



Pittsburgh – Lowry's 1964 Model of Metropolis

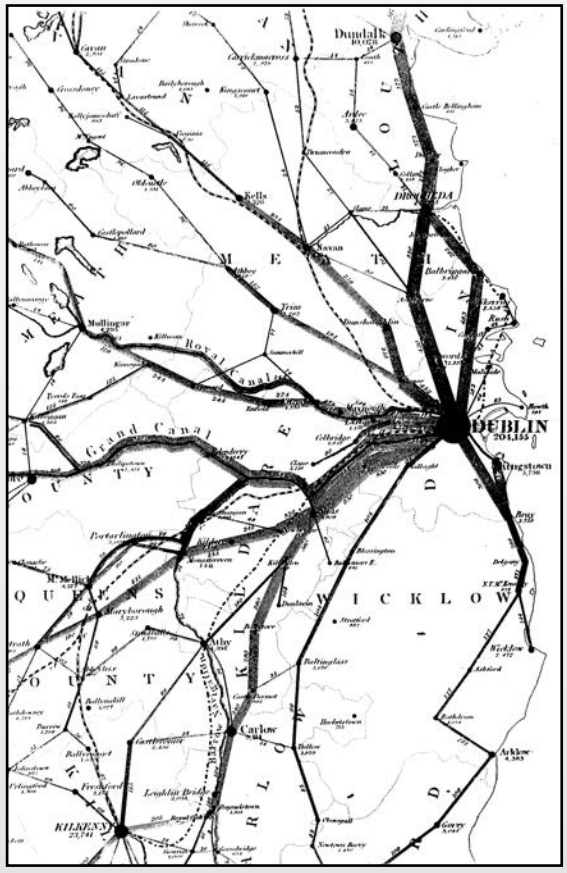




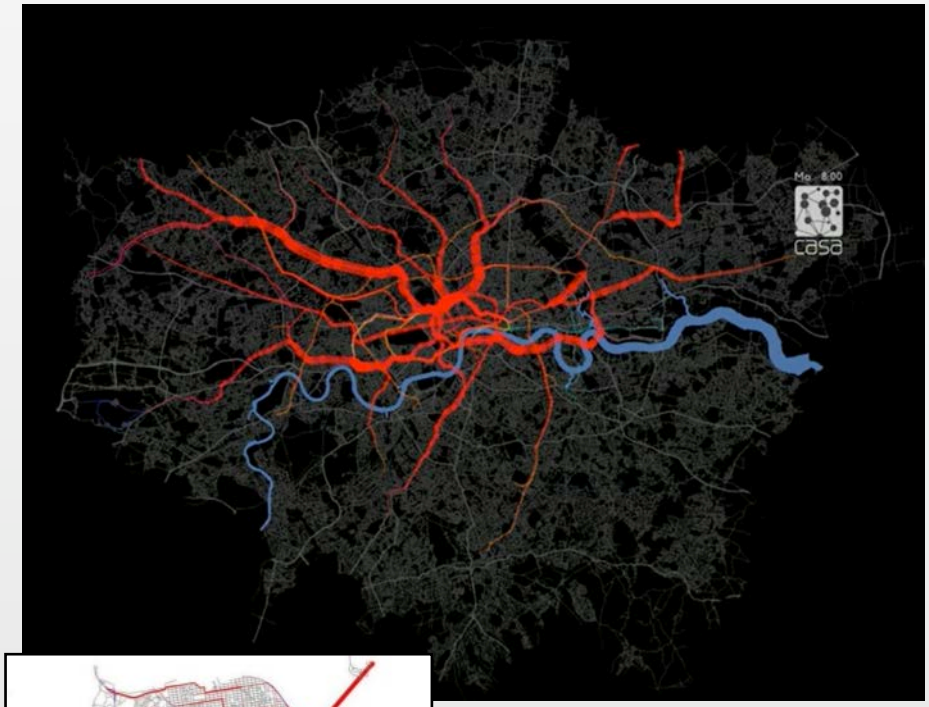


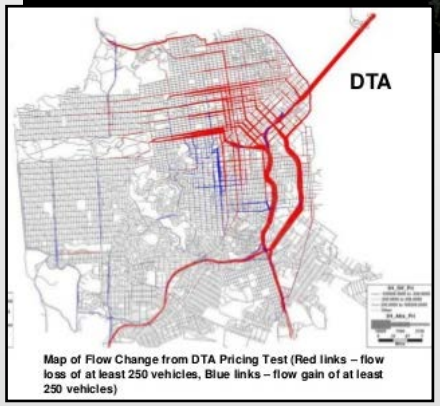
Vectors of flows positioned as average directions of all trips flowing into different destinations

In fact we often assign the flows to a network – early examples you can find in my book **The New Science of Cities** but here are typical examples



Traffic in 1837 in Dublin





Tube flows above
This is almost an analogue like the flow of blood

San Francisco left

The Matrix Notation: Dimensional Models based on Origins and Destinations

Let me begin with the algebra of spatial interaction – we denote origins by the subscript i and destinations with the subscript j . Then the flow from i to j is T_{ij} .
Ok now we usually arrange these flows in a matrix

$$[\boldsymbol{T}]=\begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & ..... & T_{1n} \\ T_{21} & T_{22} & T_{23} & T_{24} & T_{25} & ..... & T_{2n} \\ T_{31} & T_{32} & T_{33} & T_{34} & T_{35} & ..... & T_{13n} \\ . & & & & & & \\ & & & T_{ij}. & & & \\ . & & & & & & \\ T_{n1} & T_{n2} & T_{n3} & T_{n4} & T_{n5} & ..... & T_{nn} \end{bmatrix} \qquad \sum_j T_{ij} = O_i$$

$$\sum_i T_{ij} = D_j$$

Now the usual operations are adding up over the rows or the columns as we show here left

Let me repeat this so we can ponder over it

$$[\boldsymbol{T}] = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & ..... & T_{1n} \\ T_{21} & T_{22} & T_{23} & T_{24} & T_{25} & ..... & T_{2n} \\ T_{31} & T_{32} & T_{33} & T_{34} & T_{35} & ..... & T_{13n} \\ . & & & & & & \\ & & & T_{ij}. & & & \\ & & & & & & \\ . & & & & & & \\ T_{n1} & T_{n2} & T_{n3} & T_{n4} & T_{n5} & ..... & T_{nn} \end{bmatrix} \qquad \sum_j T_{ij} = O_i$$

$$\sum_i T_{ij} = D_j$$

The full matrix $\boldsymbol{T}$ is written in **bold**, note that the number of rows need not be the same as the number of columns i.e. *n* rows and *m* columns say – here I have n x n; and when we want to add up a matrix in this way we can use the following matrix operations

So first let us write $\boldsymbol{n}$ the rows sums as the origin vector

$$\boldsymbol{O} = [O_1, O_2, O_3, \dots, O_n]$$

And the $\boldsymbol{m}$ columns sums as the destination vector

$$\boldsymbol{D} = [D_1, D_2, D_3, \dots, D_m]$$

Then we can compute these by multiplying the basic matrix $\boldsymbol{T}$ by the unit vector $\mathbf{1} = [1,1,1, \dots, 1]$ as follows

$$\boldsymbol{O} = \boldsymbol{T}\mathbf{1}'$$

$$\boldsymbol{O} = [O_1, O_2, \dots, O_n] = \begin{bmatrix} T_{11}, T_{12}, \dots, T_{1m} \\ T_{21}, T_{22}, \dots, T_{2m} \\ \dots \\ T_{n1}, T_{n2}, \dots, T_{nm} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

and $\boldsymbol{D} = \boldsymbol{1T}$.

$$\boldsymbol{D} = \begin{bmatrix} D_1 \\ D_2 \\ \cdots \\ D_m \end{bmatrix} = [1, 1, \ldots, 1] \begin{bmatrix} T_{11}, T_{12}, \cdots, T_{1m} \\ T_{21}, T_{22}, \cdots, T_{2m} \\ \cdots \\ T_{n1}, T_{n2}, \cdots, T_{nm} \end{bmatrix}$$

And that is all there is to it – all we are doing in spatial interaction models is adding things up so we can make sure the accounting is right. We don't need 3-dimensional matrices but some of our models are 1-dimensional – as if we envisage the city as splayed out on a line of distance from the CBD – does everyone know what the CBD is ?

The Basic Gravitational Model

By now you know the form of the basic gravitational model, originally proposed in the 18thC in analogy to Newton's 2nd Law of Motion which from Wikipedia

Newton's law of universal gravitation

From Wikipedia, the free encyclopedia

Newton's law of universal gravitation is usually stated as that every particle attracts every other particle in the universe with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between their centers.[note 1] The publication of the theory has become known as the "first great unification", as it marked the unification of the previously described phenomena of gravity on Earth with known astronomical behaviors.[1][2][3]

And here is the model again with its variables

$$T_{ij} \propto \frac{P_i P_j}{d_{ij}^2} = K \frac{P_i P_j}{d_{ij}^2}$$

$$T_{ij}, P_i, P_j, d_{ij}^2, \text{and } K$$

Mass or Population
- the attractor

Scaling
constant

Flow
(or Force)

$$T_{ij} \propto \frac{P_i P_j}{d_{ij}^2} = K \frac{P_i P_j}{d_{ij}^2}$$

The deterrence

The parameter

Note that we will henceforth label origins and destinations not as

P_i and P_j

but as

O_i and D_j

Let me now write the model in full scaling form which I believe is the model that is introduced in the practical

$$T_{ij} = K\frac{O_i^{\alpha} D_j^{\gamma}}{d_{ij}^{\beta}} = K O_i^{\alpha} D_j^{\gamma} d_{ij}^{-\beta}$$

And in logarithmic linear form it is

$$\log T_{ij} = \log K + \alpha \log O_i + \gamma \log D_j - \beta log\, d_{ij}$$

Different Distance Functions: Scaling

We can modify this general model but showing how we can use different variables and alter its parameters. First we can define a generalised deterrence as a travel cost which might be some combination of distance travelled, cost incurred and time taken. We define this as c_{ij} and we can raise this to a power β. In fact we use the index notation to turn it into an inverse power law as $c_{ij}^{-\beta} = 1/c_{ij}^{\beta}$.

Now we can redefine our attractors as some measure of the origin and destination populations as O_i and D_j And the generalised model becomes

$$T_{ij} \propto \frac{O_i^{\alpha} D_j^{\gamma}}{c_{ij}^{\beta}} = K \frac{O_i^{\alpha} D_j^{\gamma}}{c_{ij}^{\beta}} = K O_i^{\alpha} \; D_j^{\gamma} c_{ij}^{-\beta}$$

We also use other functions of generalised travel cost, that is other than the inverse power form $c_{ij}^{-\beta}$ and the usual one is the negative exponential which is very similar but written as $e^{-\beta c_{ij}} = \exp(-\beta c_{ij}) = 1/e^{\beta c_{ij}}$. The model can now be written as

$$T_{ij} = K O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})$$

Note that we have parameterised the origin and destination attractor terms as explicit power functions

$$T_{ij} = K O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})$$

Potential and Accessibility

In the 1940, the astronomer John Stewart suggested that a measure of potential could be produced from the gravity model that was an overall measure of the force of an object on all others. He defined this from the basic Gravity Model equation (which uses P_i and P_j) as potential V_i or potential per capita v_i

$$V_i = \sum_j T_{ij} \propto P_i \sum_j \frac{P_j}{d_{ij}^2}$$

$$v_i \propto \frac{V_i}{P_i} = \sum_j \frac{P_j}{d_{ij}^2}$$

This is essentially accessibility or nearness and it was first used as the basis for a simple urban model by

Walter Hanson in the late 1950s in a paper called "How Accessibility Shapes Land Use". There he said that the residential development in a place was a simple function of accessibility i.e.

$$R_i \propto \frac{V_i}{P_i} = \sum_j \frac{P_j}{d_{ij}^2}$$

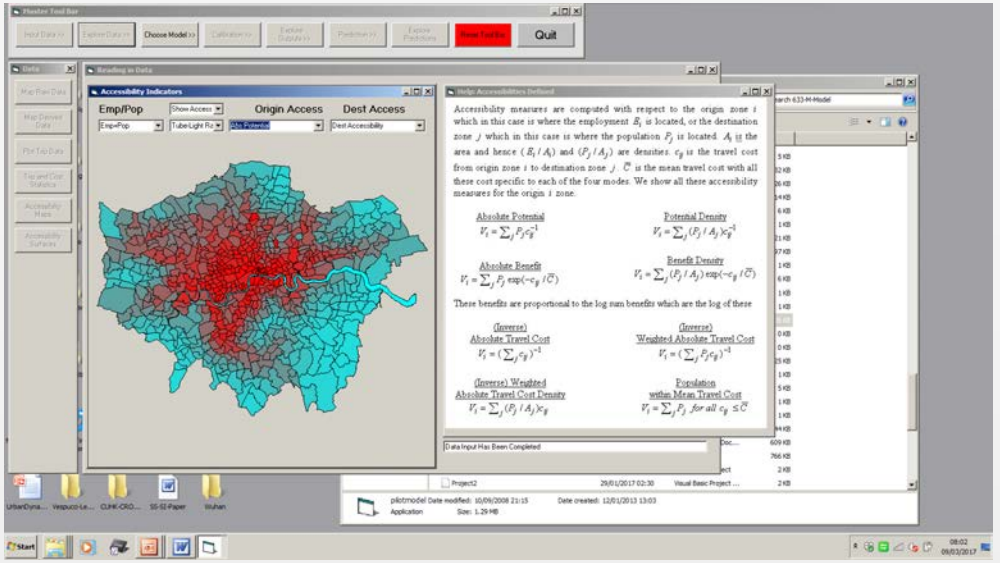
In fact if total residential development is R, then the equation can be written as

$$R_i = R \frac{(V_i / P_i)}{\sum_k (V_k / P_k)}$$

And this is our first operational land use model, the simplest

Walter G. Hansen, W. G. (1959) How Accessibility Shapes Land Use, **Journal of the American Institute of Planners**, Volume **25**, Pages 73-76 http://dx.doi.org/10.1080/01944365908978307

There are many definitions of accessibility but essentially they are all composite measures of attraction and deterrence from any one place to all others. The model I am using for London has many of these



Note that we could also scale the accessibility measures as $V_i \propto \sum_j T_{ij} \propto O_i^{\alpha} \sum_j D_j^{\gamma} / d_{ij}^{\beta}$. There are many variants like this

http://www.casa.ucl.ac.uk/urbanmodelling/

Ensuring the Model Meets Basic Constraints

Now in the basic model we have two 'scaling' parameters K and β. We will distinguish between these calling K the constant of proportionality and β the friction of distances parameter. This parameter K can be worked out by noting it adjusts the trips to make sure that they 'add up' to the total number of trips T

$$T = \sum_{i=1}^{n}\sum_{j=1}^{m} T_{ij} = \sum_{i}\sum_{j} T_{ij}$$

Now note how we can miss off the *n* and *m* and the range of the summation is obvious by context

Now the key thing in this sort of model is that we substitute the model into the constraint equation shown above. So let us do this and simplify the substitutions and thence derive the value of the parameter K . So here we go. You can also see the working in the handout for the computer program we are about to develop and demonstrate. Then substitute the model. $T_{ij} = KO_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})$ into the constraint equations on total trips

$$T = \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij}$$

And we then get the following derivation

$$T = \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij} = K \sum \sum O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})$$

and we get $$K = \frac{T}{\sum \sum O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})}$$

The model can be written out in full a

$$T_{ij} = T \frac{O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})}{\sum \sum O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})}$$

and we can think of this as defining probabilities of trips between $i \; and \; j$

$$T_{ij} = T p_{ij};$$

$$\text{where } p_{ij} = \frac{O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})}{\sum \sum O_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})}; \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} = 1$$

Explaining the First Computer Program

I am going to go through the handout quite quickly and then run the model. Note the following

- This is the ***unconstrained model*** just introduced *but without scaling parameters on the mass/attactors* – in fact there is an overall constraint on it so the trips add up to T but we call it this because other models in the family of spatial interaction models have different degrees of constraint
- In the computer program, we define some data on origins, destinations, trips and distances arbitrarily and hypothetically – we don't use real data but we do need some data to calibrate our model with.

- We thus divide our program into getting the data sorted out. You are allowed to fix the size of a grid by specifying the number of units of horizontal or vertical of a square grid of zones – and we recommend *n= 5*, or *n=7*, or *n*=9, or *n = 11* but if you select more, then program graphics is tiny and it makes more sense to keep this number small. Note that if you specify *n= 5*, you still have n^2*=25 zones*
- Running the model – we note that the model works out K and we have to specify β. In the practical I think you have to calibrate – i.e. fine tune β – but here we just do it once. If you want to run the model for different β you can do so.

- In fact the model produces scatter plots of the predicted origins and destinations and trips against the observed (which we have said are the hypothetical/ made-up), and we can see how good the correlation is
- We also plot scatter diagrams of the grid of observed and predicted origins and destinations and the differences. These are scatter graphs we we plot the size of the origins and destinations observed v predicted by the size of the scatter dots
- You need to have a look at the program and if you have learnt Python, then think about the elements in the program.

- Ok let me look at the handout and restate it

The Simplest Urban Models: The Basic Gravity Model

The basic model simulates flows of any kind of activity in a city – often called trips or spatial interactions. These are physical flows of transport between different locations which are referred to as origins and destinations. We subscript any variable at an origin by i and at a destination by j and any flow between an origin and destination ij by trips T_{ij}. The standard model of such a flow is based on gravitational principles meaning that as the deterrence measured by distance, travel cost or travel time between an origin and a destination, called henceforth d_{ij}, increases, the number of trips declines.

This model is usually written as

$$T_{ij} = K\frac{O_i D_j}{d_{ij}^{\beta}} = KO_i D_j d_{ij}^{-\beta} \quad . \qquad (1)$$

Note that the model has two parameters – a scaling constant K which adjust the 'scale' of the model to ensure that we get it in the right units and β which is sometimes called the 'friction of distance' parameter and which adjust the deterrence function $d_{ij}^{-\beta}$ to the best possible simulation we can get of the real situation. We therefore use this parameter as a way to calibrate or fine tune the model.

The model also includes the size of the origin which we call O_i and the size of destination as D_j and as these get larger, then the number of trips also gets larger in proportion to these. So you can think of such a model as a balance between the size of two places – sometimes as in physics called the mass, that is $O_i D_j$ – and the deterrence of distance between these masses d_{ij}. The model was first proposed by Isaac Newton in the late 17th century where he wrote down the gravitational force or attraction between the planets as $F_{ij} = GP_i P_j d_{ij}^{-2}$ where the constant G is the gravitational constant, distance is assumed to be the inverse square law, and P_i and P_j are the size of the planets measured by their mass.

Now to estimate this model, we need to find β and K in the model above which ensure that the difference between the observed trips T_{ij}^{obs} and the model T_{ij} – predicted – trips are at a minimum of some sort. In fact we need to make sure the model generates no more or less than a total of T trips,

$$\sum_i \sum_j T_{ij} = \sum_i \sum_j T_{ij}^{obs} = T \quad . \qquad (2)$$

Now it is easy to show that if we put the model in equation (1) above into equation (2), we get the following

$$\sum_i \sum_j T_{ij}^{obs} = \sum_i \sum_j KO_i D_j d_{ij}^{-\beta} = K\sum_i \sum_j O_i D_j d_{ij}^{-\beta} = T \qquad (3)$$

and from this we can calculate K as

$$K = T / \sum_i \sum_j O_i D_j d_{ij}^{-\beta} \quad , \qquad (4)$$

and the full model is thus

$$T_{ij} = T\frac{O_i D_j d_{ij}^{-\beta}}{\sum_i \sum_j O_i D_j d_{ij}^{-\beta}} \quad . \qquad (5)$$

To finally calibrate this model the best way we can do this at this point is to try different values of β and then choose one from the set of values that maximises the goodness of fit. A simple measure is the correlation coefficient between T_{ij} and T_{ij}^{obs}. I will not define the correlation other than say that this gives the percent of the model that explains the data and we want the correlation to be as large as possible.

Now the computer program we have to explore this model does five things in the following order. This is a set of workflows which are:

1. We set up an hypothetical set of zones arranged on a grid. The program asks the user to specify a number of zones on one side of the grid which might be N and it then creates a grid of N^2 zones. Try $N = 5$ or $N = 10$. More than this you can explore but as you go bigger, it takes exponentially longer to compute and if you put in 100, then this would give you 10,000 zones !

2. It then creates some hypothetical data for the grid you specify. In fact it produces an observed trip matrix

   $$T_{ij}^{obs} = 1000\frac{d_{ij}^{-0.1-rnd(1)}}{\sum_i \sum_j d_{ij}^{-0.1-rnd(1)}}$$

   Now this is roughly similar to the model but it does generate hypothetical data that we can fit and is not the same as the model. We also get the observed origins and destinations activity from this equation, that is

   $$\sum_j T_{ij}^{obs} = O_i^{obs}; \sum_i T_{ij}^{obs} = D_j^{obs}; \sum_i \sum_j T_{ij}^{obs} = \sum_i O_i^{obs} = \sum_j D_j^{obs} = T$$

3. The next stage is to compute the model in equation (1) or in fact in equation (5). You have to input a value for β and because we use a negative exponential function in the program, I advise putting in a value of 0.5. From the model, we then predict values of

   $$T_{ij},\ \sum_j T_{ij} = O_i \ , \text{and } \sum_i T_{ij} = D_j \quad .$$

   We can now compare these with the observed totals. First we compute three scatter graphs and their correlations, namely $O_i \ v\ O_i^{obs}$, $D_j \ v\ D_j^{obs}$ and $T_{ij} \ v\ T_{ij}^{obs}$

4. Then we can compare the grid maps of the same variables with each other. First $O_i \ v\ O_i^{obs}$ on the N^2 grid and then $D_j \ v\ D_j^{obs}$ on the N^2 grid.

5. Then we work out differences $(O_i - O_i^{obs})$ and $D(_j - D_j^{obs})$ and plot these.

There is a wealth of data here to compare the 'observed' (but hypothetical) data with the predicted and I will point out some issues when we discuss these in the class.

- Now I will look at the program

BasicGravityModel

January 17, 2021

[2]:
```
%matplotlib inline
import numpy as np
import math
from matplotlib import pyplot as plt
from numpy import random
import numpy as np
plt.style.use('seaborn-whitegrid')

#Defined Functions: for Plotting observed and Predicted Model Results on the␣
 ↪Grid,
#the Friction of Distance Parameter, and the Location-Interactions Plots for␣
 ↪Correlations

def friction(parameter,dis):
    frict=math.exp(parameter*dis)
    return frict

def locintplot(obs,pred,activities):
    maxo=np.max(obs); maxop=np.max(pred)
    if maxo>maxop:
        maxv=maxo
    else:
        maxv=maxop
    maxv=maxv*(1.1)
    plt.axis([0,maxv,0,maxv])
    plt.xlabel("Observed" + activities)
    plt.ylabel("Predicted" + activities)
    plt.title(activities + " Activity")
    plt.scatter(pred, obs, s=10,c='black')
    r0 = np.corrcoef(pred, obs)
    print("Correlation ","\t","{:.3f}".format(r0[0,1]))
    return

def scattergraph(soutput,dataname,colours):
    plt.show()
    sizes=soutput**2
    plt.axis([0,n,0,n])
```



```
    plt.xlabel("x coordinate")
    plt.ylabel("y coordinate")
    plt.title(dataname + ' Activity')
    plt.scatter(xcoord, ycoord, c=sizes, s=sizes, alpha=0.75, cmap=colours)
    return

#Defining the Hypothetical Spatial System

xcoord=np.array([])
ycoord=np.array([])

n=input("Enter the Grid Size: The Number of Squares Along One Side of a Square␣
 ↪Grid, 5 or 10, say --- ")
n=int(n); N=n*n
print("The Size of the Hypothetical Spatial System is",n, "Zones by", n,"Zones,␣
 ↪Making", N,"in All")
print()

distance=np.full((N,N), 1.0)

n=n+1
for y in range(1,n):
    for x in range(1,n):
        xcoord = np.append(xcoord,[x])
        ycoord = np.append(ycoord,[y])

#for i in range(0,N):
#        print(i, xcoord[i], ycoord[i])

input()
ij=0
for i in range (0,N):
    xi=xcoord[i]
    yi=ycoord[i]
    for j in range (0,N):
        ij=ij+1
        xj=xcoord[j]
        yj=ycoord[j]
        dis=math.sqrt((((xi-xj)**2)+((yi-yj)**2)))
        distance[i][j]=dis
        if distance[i][j]==0:
            distance[i][j]=0.5
        #print(i+1,j+1,distance[i][j])

#Defining the Hypothetical Trip, Origin and Destination Data

tobs=np.full((N,N), 1.0)
```

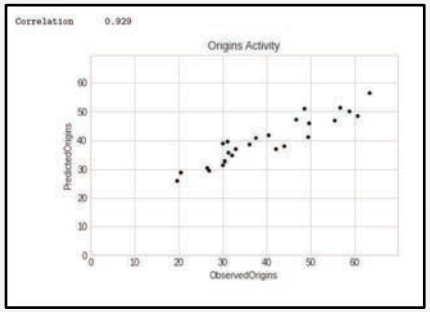
```
origins=np.full((N),1.0)
destinations=np.full((N),1.0)
differences=np.full((N),1.0)

for i in range (0,N):
    for j in range (0,N):
        tobs[i][j]=1.0/(distance[i][j]*(0.1+random.rand()))
Tobs = np.sum(tobs)
for i in range (0,N):
    for j in range(0,N):
        ij=ij+1
        tobs[i,j]=1000*(tobs[i][j]/Tobs)

origins = np.sum(tobs, axis = 1)
destinations = np.sum(tobs, axis = 0)

To=np.sum(origins)
Td=np.sum(destinations)
Tobs = np.sum(tobs)

#Defining and Running th Unconstrained Gravity Model

beta=input("Enter the Parameter Value on Distance - beta - that You Think Best␣
 ↪Fits the Data: It should be greater than 0 and less than 1 ---- ")
beta=float(beta)
print()
trips=np.full((N,N),1.0)
OPred=np.full((N),1.0)
DPred=np.full((N), 1.0)
trips1=np.full((N,N),1.0)
tobs1=np.full((N,N),1.0)

total=1000
for i in range(0,N):
    for j in range(0,N):
        trips[i][j]=origins[i]*destinations[j]/(friction(beta, distance[i][j]))
Ttrip=np.sum(trips)
for i in range(0,N):
    for j in range(0,N):
        trips[i][j]=total*((origins[i]*destinations[j])/(friction(beta,␣
 ↪distance[i][j])))/Ttrip

#Printing the Predictions

OPred = np.sum(trips, axis = 1)
DPred = np.sum(trips, axis = 0)
```

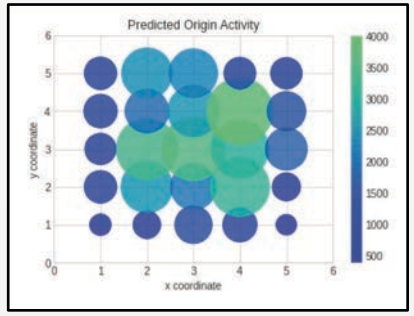


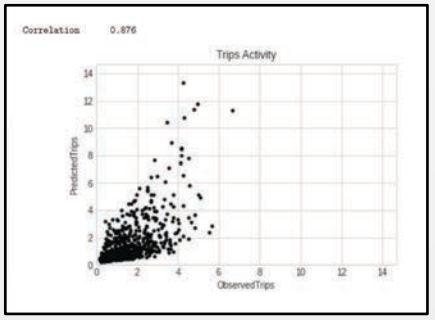
```
print("Zone","\t","ObsO","\t","ObsD","\t","PredO","\t","PredD")
print()
for i in range(0,N):
    print(i+1, "\t","{:.2f}".format(origins[i]), "\t","{:.2f}".
 ↪format(destinations[i]),"\t","{:.2f}".format(OPred[i]),"\t","{:.2f}".
 ↪format(DPred[i]))

#Comparing Observed with Predicted Origin Activity Using Scattergraphs

input()
locintplot(origins, OPred, 'Origins')
plt.savefig('OriginOutputs.png', dpi=300, bbox_inches='tight')
plt.show()

input()
locintplot(destinations, DPred, 'Destinations')
plt.savefig('DestOutputs.png', dpi=300, bbox_inches='tight')
plt.show()

input()
locintplot(tobs, trips, 'Trips')
plt.savefig('TripOutputs.png', dpi=300, bbox_inches='tight')
plt.show()

rng = np.random.RandomState(0)
colors = rng.rand(N)

#Plotting Observed and Predicted Locations on the Hypothetical Grid

input()
scattergraph(origins, 'Observed Origin','winter')
plt.savefig('ObsOrigins.png', dpi=300, bbox_inches='tight')
plt.colorbar()
plt.show()

input()
scattergraph(OPred, 'Predicted Origin','winter')
plt.savefig('PredOrigins.png', dpi=300, bbox_inches='tight')
plt.colorbar()
plt.show()

input()
scattergraph(destinations, 'Observed Destination','winter')
plt.savefig('ObsDestinations.png', dpi=300, bbox_inches='tight')
plt.colorbar()
plt.show()
```

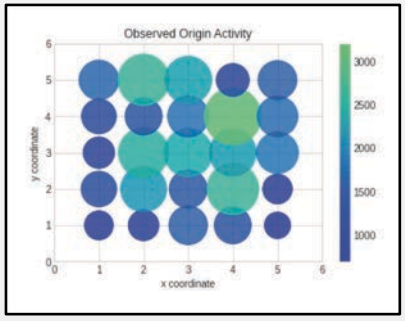
```
input()
scattergraph(DPred, 'Predicted Destination','winter')
plt.savefig('PredDestinations.png', dpi=300, bbox_inches='tight')
plt.colorbar()
plt.show()

#Measuring the Differences Between Predictions and Observations

for i in range(0,N):
    differences[i]=(origins[i]-OPred[i])*6
input()

scattergraph(differences, 'Differences in Origin ','autumn')
plt.savefig('Odifferences.png', dpi=300, bbox_inches='tight')
plt.colorbar()
plt.show()

for i in range(0,N):
    differences[i]=(destinations[i]-DPred[i])*6
input()
scattergraph(differences, 'Differences in Destination ','autumn')
plt.savefig('Ddifferences.png', dpi=300, bbox_inches='tight')
plt.colorbar()
plt.show()

print("The model and its outputs are now complete")
print()
```

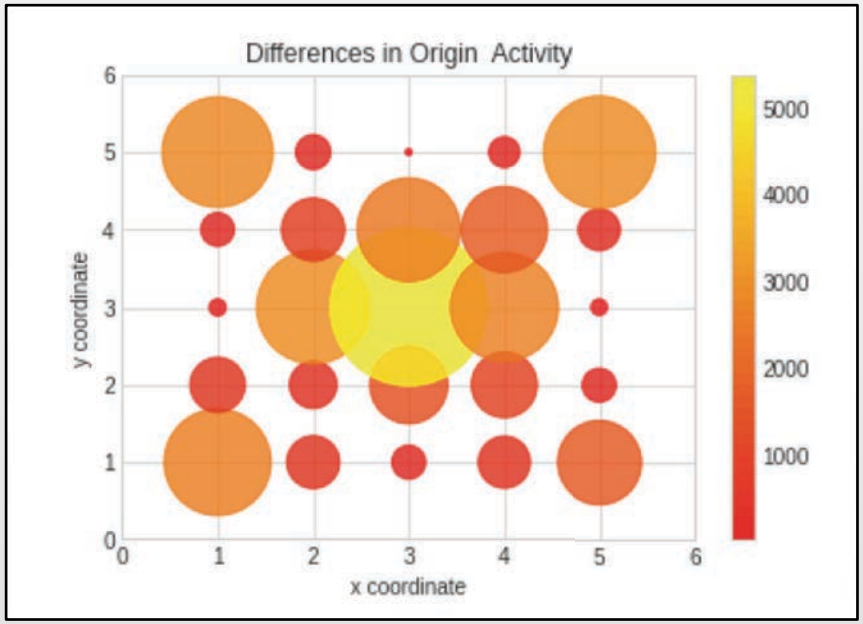
```
Enter the Grid Size: The Number of Squares Along One Side of a Square Grid, 5 or
10, say ---  5

The Size of the Hypothetical Spatial System is 5 Zones by 5 Zones, Making 25 in
All


Enter the Parameter Value on Distance - beta - that You Think Best Fits the
Data: It should be greater than 0 and less than 1 ----  0.5
```





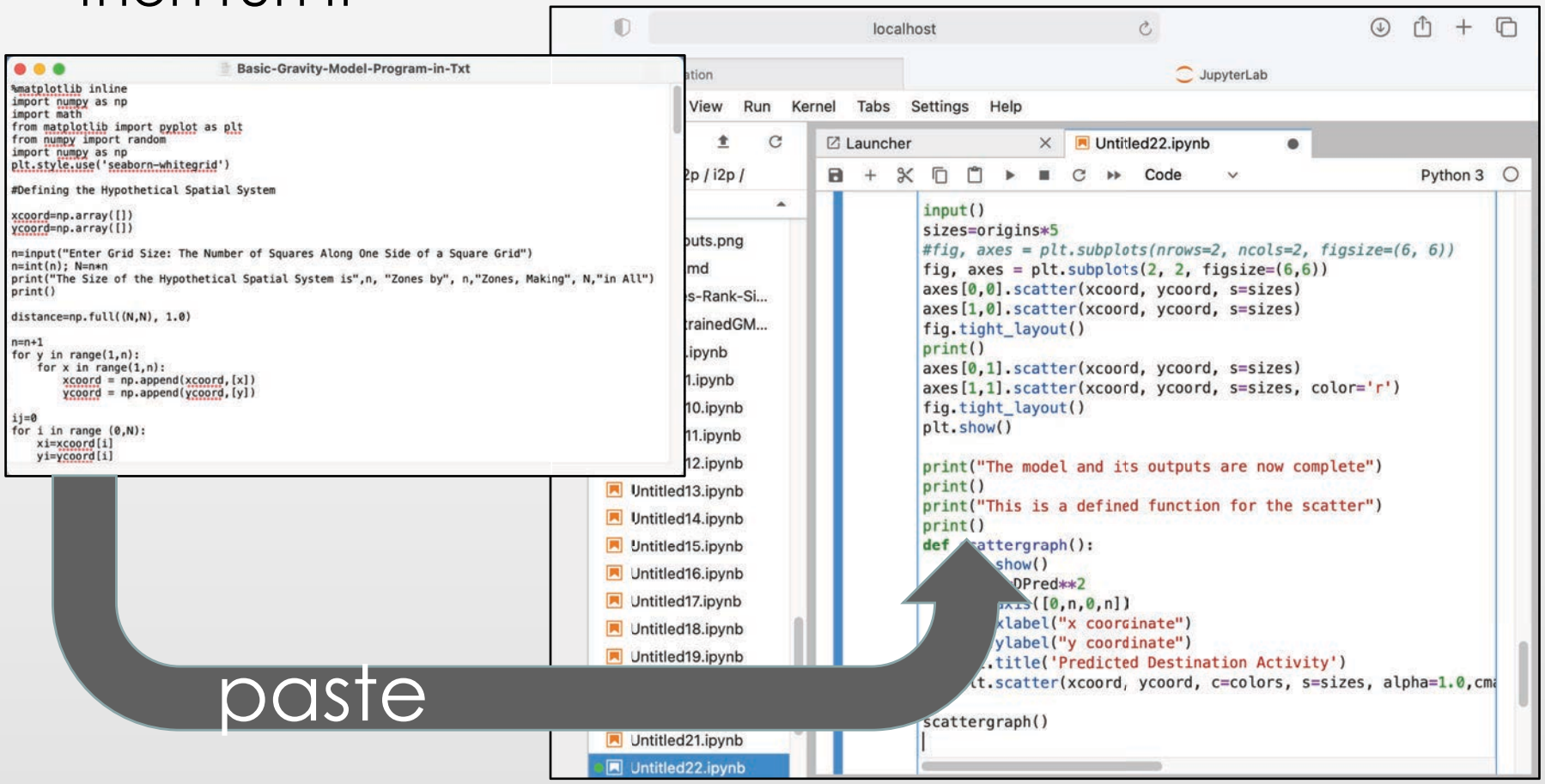






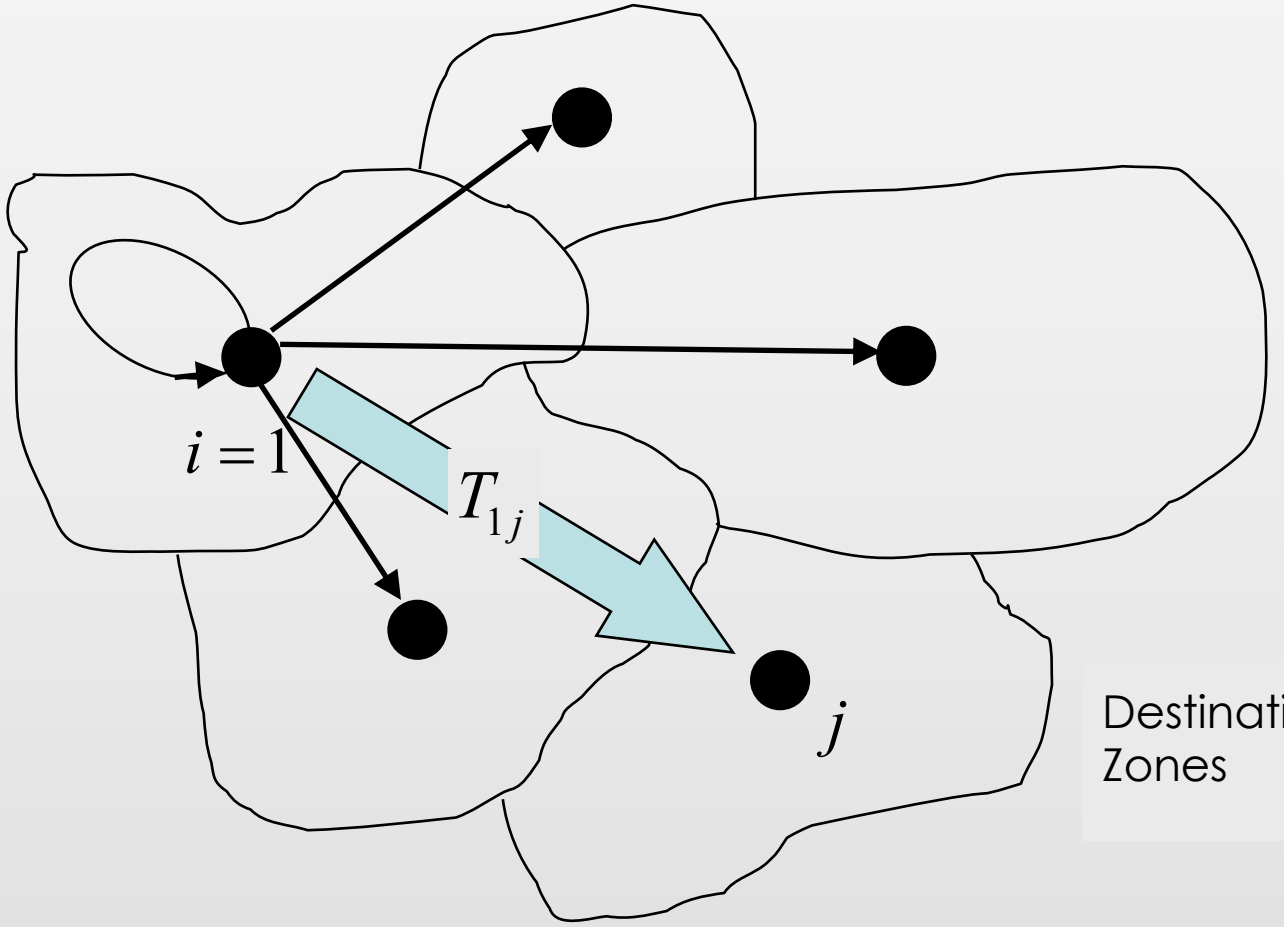
- Here are typical outputs from a model run

- Your have the program that is on Moodle called **Basic-Gravity-Model-Program-in-Txt** and you can open this and paste it into a Jupyter notebook, and then run it



Population Density: 1- Dimensional Models

Now imagine we only have **ONE** origin and **MANY** destinations – instead of flows to everywhere we have flows only from say the city centre, the CBD



Let us use our negative exponential form of gravity model and then the model becomes

$$T_{1j} = KO_1^{\alpha} D_j^{\gamma} \exp\left(-\beta c_{1j}\right)$$

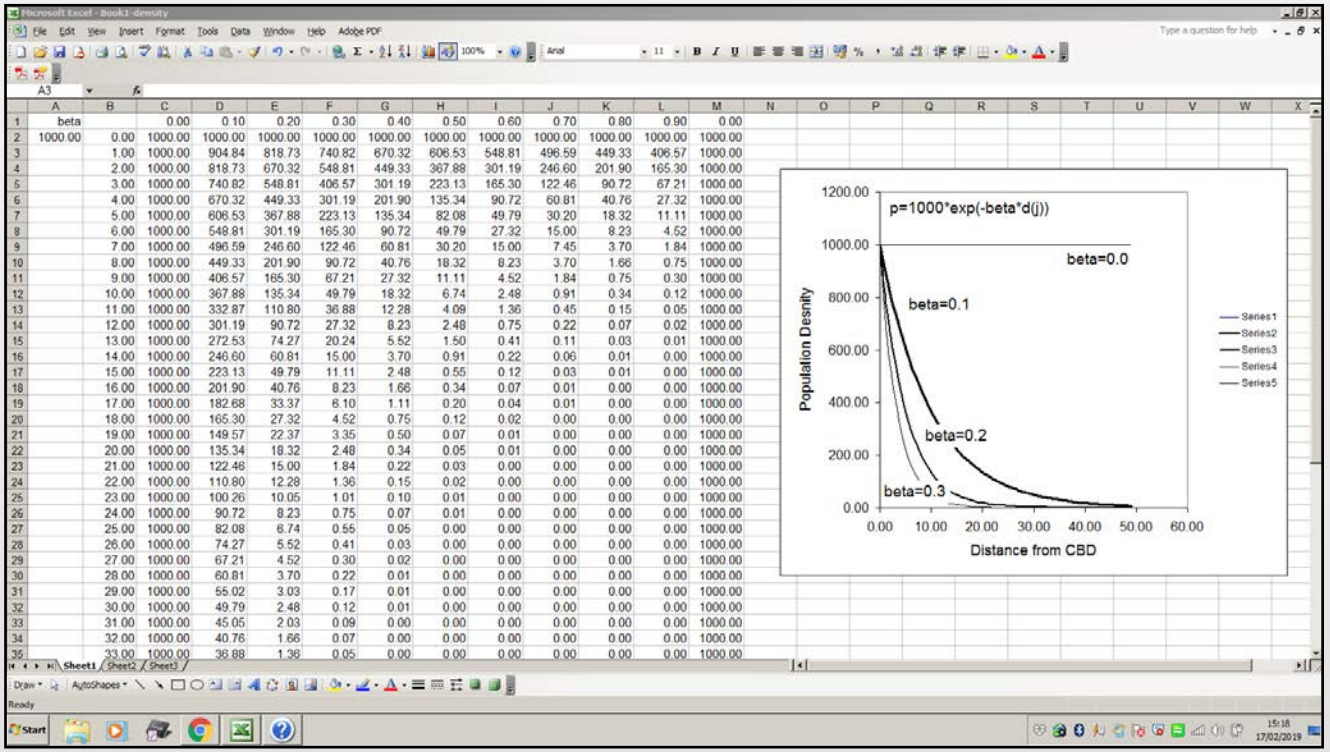
We can now get rid of the origin notation which is always the city centre (CBD) and the scaling constant of proportionality and the origin at zone 1 are always the same. We thus collapse them into a constant Z and the travel cost is thus $c_{1j}= c_j$. The origin at zone 1 are always the same. We thus collapse them all into a constant Z and the travel cost is thus $c_{1j}= c_j$. If we call $P_j= T_{1j}$, the population, then the model is

$$P_j = ZD_j^{\gamma} \exp\left(-\beta c_j\right)$$

Now we might think of the flow as population density and if we assume the destination variable is the land area, then this gives us a density, that is

$$\rho_j = \frac{P_j}{D_j^{\gamma}} = Z \exp\left(-\beta c_j\right)$$

Here is a demo of what these curves look like – to get experience of them you can either proposes some data in a spreadsheet or wite a little python program



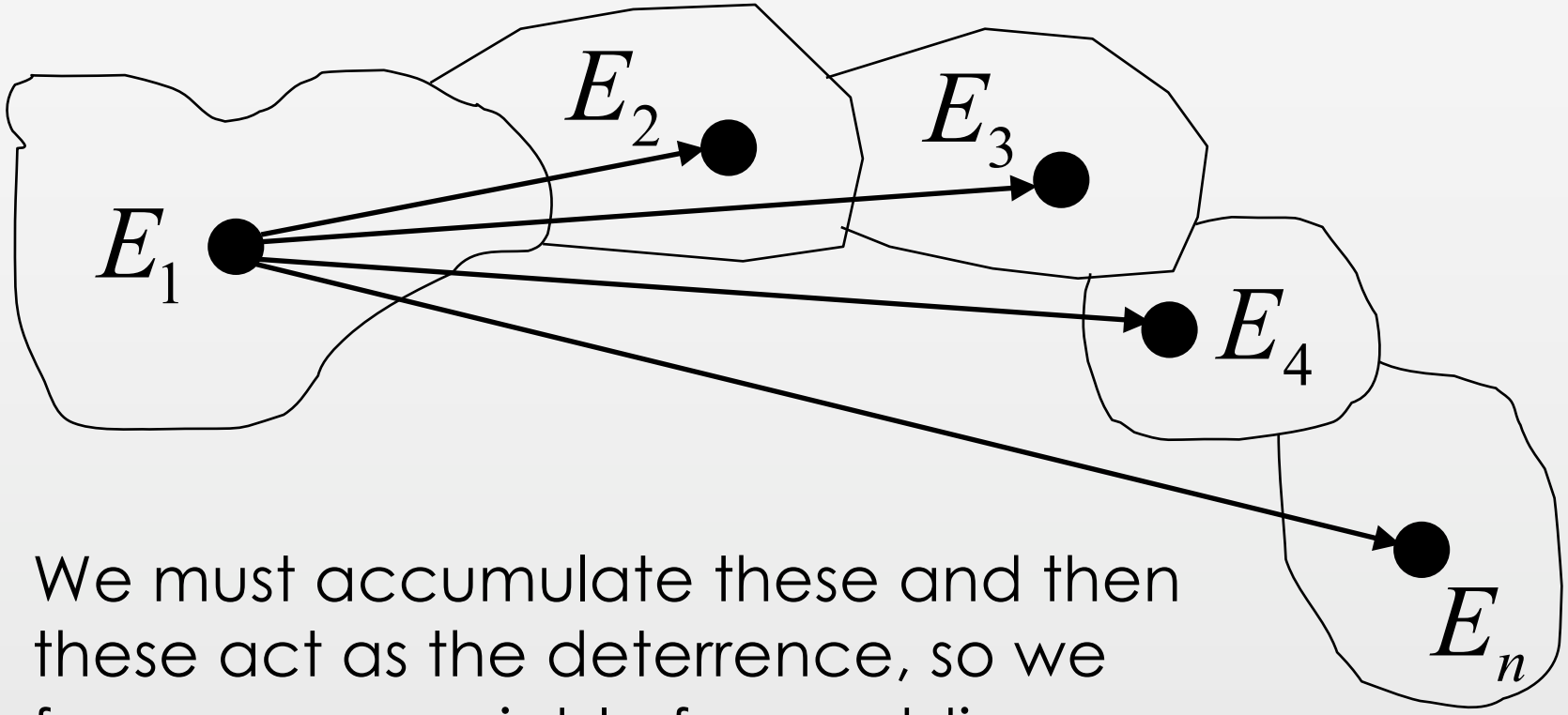
Intervening Opportunities Models

Now we have seen that we can define the deterrence function as either distance or travel time or travel cost or a more generalised measure of travel cost $f(c_{ij})$ using a specific function such as the inverse power or negative exponential

$$d_{ij}^{-\beta} \quad or \quad c_{ij}^{-\beta} \; or \; \exp(-\beta c_{ij}) \; or \; f(c_{ij})$$

We can in fact replace this deterrence with all the opportunities called E_z that a traveller or trip maker passes at z as they head out further from the origin in question. In short the more opportunities – jobs say – that they pass, the less likely they are to travel further

Now these are the opportunities that are passed as the trip makers heads out to zone n



We must accumulate these and then these act as the deterrence, so we form a new variable from adding these as

$$C_j = E_1 + E_2 + E_3 + \ldots + E_n = \sum_{z=1}^{n} E_z$$

All we do is substitute the cumulative opportunities for the deterrence in any version of the gravity model that we wish to use. But we note that these intervening opportunities are subscripted as $C_{i(z)j}$ where the path from i to j is over zones z. We can put these into our unconstrained model which is

$$T_{ij} = KO_i^{\alpha} D_j^{\gamma} \exp\left(-\beta C_{i(z)j}\right)$$

Basically there are different ways of counting intervening opportunities and there are even ways of counting opportunities within zones. Thus there many variants of these kinds of model.

I have run out of time and will present the family of spatial interaction models and the relevant computer program next week as well as ideas about coupled models and applications

m.batty@ucl.ac.uk
@jmichaelbatty

Monday, 15 January 2024

Urban Simulation 2 (continued)

Spatial Interaction: Basic Concepts

Michael Batty

m.batty@ucl.ac.uk
X@jmichaelbatty

Monday, 22nd January 2024

http://www.spatialcomplexity.info/

Modifying the Generic Model: The Family of Spatial Interaction Models from Lecture 2

If you compare last year's lectures with this, at Elsa's instigation, I have notated the mass variables in the gravity model – origins and destinations – by scaling coefficients i.e. used $T_{ij} = KO_i^{\alpha} D_j^{\gamma} \exp(-\beta c_{ij})$. This is because the extended model is used in the practical.

But I will now use $T_{ij} = KO_i D_j \, exp\left(-\beta c_{ij}\right)$ when it comes to the family of gravity models. With different constraints, it is tricky to notate the origin and destination variables parameters, so I have dropped these

The Family of Spatial Interaction Models

We have seen how we can constrain the model to produce specific types that meet various constraints on the amount of trips attracted to different locations. Our first and most basic model is unconstrained in that we ensure that the only constraints the model meets are on the total trips which are predetermined, already known, that is $T = \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij}$.

Now if we know the trips attracted to the origin, or to the destination, or to both, then these gives rise to different models. Including the basic model there are four in all that meet the following. This is the family.

1. The Unconstrained Model

$T_{ij} = K O_i D_j \exp(-\beta c_{ij})$ subject to $\sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij} = T$

2. The Singly-Constrained Models

2a: The Origin- Constrained Model

$T_{ij} = A_i O_i D_j \exp(-\beta c_{ij})$ subject to $\sum_{j=1}^{m} T_{ij} = O_i$

2b: The Destination-Constrained Model

$T_{ij} = O_i B_j D_j \exp(-\beta c_{ij})$ subject to $\sum_{i=1}^{n} T_{ij} = D_j$

3. The Doubly Constrained Model

$T_{ij} = A_i O_i B_j D_j \exp(-\beta c_{ij})$ st $\sum_{j=1}^{m} T_{ij} = O_i$ & $\sum_{i=1}^{n} T_{ij} = D_j$

We can work out the full models by substituting the model on the left into the constraint on the right

Now these models not only distribute trips but if we add their flows over origins or destinations or both, the different models give us predictions of what happens at origins and destinations. These are then **location models** and their predictions are in red

The Models	**The Predictions**
$T'_{ij} = KO_iD_j \exp(-\beta c_{ij})$;	$\sum_{j=1}^{m} T'_{ij} = O'_i$; $\sum_{i=1}^{n} T'_{ij} = D'_j$
$T'_{ij} = A_iO_iD_j \exp(-\beta c_{ij})$;	$\sum_{j=1}^{m} T'_{ij} = O_i$; $\sum_{i=1}^{n} T'_{ij} = D'_j$
$T'_{ij} = O_iB_jD_j \exp(-\beta c_{ij})$;	$\sum_{j=1}^{m} T'_{ij} = O'_i$; $\sum_{i=1}^{n} T'_{ij} = D_j$
$T'_{ij} = A_iO_iB_jD_j \exp(-\beta c_{ij})$;	$\sum_{j=1}^{m} T'_{ij} = O_i$; $\sum_{i=1}^{n} T'_{ij} = D_j$

We can pick one of these models and show how we can unpack it and work out the scaling constants. We will take the ***origin-constrained model*** and adapt it to a retailing example. Here the is the model

$$T'_{ij} = A_i O_i D_j \exp\left(-\beta c_{ij}\right) \; subject\ to \; \sum_{j=1}^{m} T'_{ij} = O_i$$

and this model predicts what we want to know about where people will locate at the destination

$$\sum_{i=1}^{n} T'_{ij} = D'_j$$

Now we need to figure out the scaling constant A_i and we substitute the model into $\sum_{j=1}^{m} T'_{ij} = O_i$ and then re-arrange as

$$\sum_{j=1}^{m} T'_{ij} = A_i O_i \sum_j D_j \exp(-\beta c_{ij}) = O_i$$

and then by cancelling O_i from both sides we get

$$A_i = 1/\sum_j D_j \exp(-\beta c_{ij})$$

This then lets us predict the amount of activity attracted to destination j

$$D'_j = \sum_{i=1}^{n} T'_{ij}$$

If the origin activity is expenditure on retailing where the population lives and the destination is some measure of the size of activity such as retail floorspace. Then the predicted activity is sales at the destination shopping centres

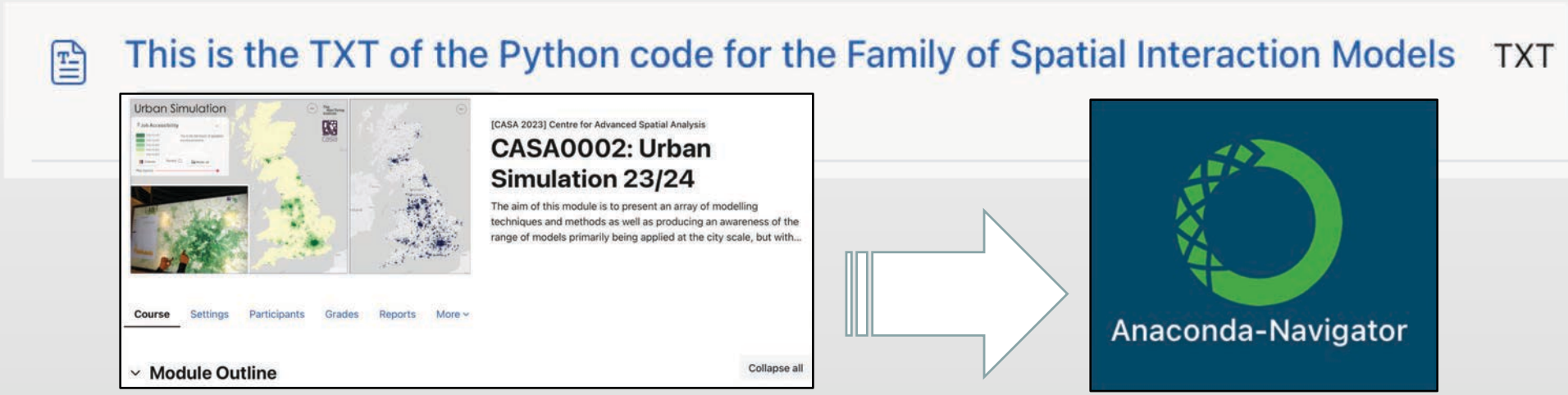
Now the new computer program on the web site

Spatial-Interaction-Models.ipynb

Lets you choose one of these four models and using the same hypothetical data set shown earlier , it runs the model and presents the input and output data and the correlations of predicted v observed trips, origins and destinations.

```
Enter the Grid Size: The Number of Squares Along One Side of a Square Grid, 5 or 10, say ---  5
The Size of the Hypothetical Spatial System is 5 Zones by 5 Zones, Making 25 in All

Enter the Parameter Value on Distance - beta - that You Think Best Fits the Data: It should be greater than 0 and less than 1 ----  0.5

You now have to input one of the Four Model Variants that you want to run

We will number these as

1 The Unconstrained Model as in the Basic Gravity Model Program
2 The Singly-Constrained (Origin-Constrained) Model
3 The Singly-Constrained (Destination-Constrained) Model
4 The Doubly-Constrained (Origin-and-Destination Constrained) Model

Type in the Relevant Number from 1 to 4 3
You have chosen the Destination-Constrained Model
Zone     ObsO    ObsD    PredO   PredD

1        28.93   34.57   22.18   34.57
2        45.28   32.50   41.04   32.50
3        39.63   39.23   37.97   39.23
```

I will now load Anaconda and then go to Moodle and get the text file and copy that to the clipboard and past it into a new file on the Jupyter Notebook. I will then run the model. Note that we can pick any of the four models from this program and run them and look at the predicted origins and destinations and trips as we did last week. Let me show you (and I have a new machine that works better than last week).



Urban Models: Coupled Spatial Interaction

Ok – we have the building blocks of more comprehensive urban models now. Imagine we want to build a model of where people work, where they live and where they shop.

We first define where people work as employment at origins E_i and where they live as population at destinations P_j. This can be modelled as a singly constrained model where we predict P_j from E_i and then we predict where these people will shop as the number of workers or size of the shopping centre S_i at the origins. Note we are no longer using O_i or D_j .

We thus start with journey to work model defined as