The Geography of Development - Readme for the code

Klaus Desmet, Dávid Krisztián Nagy and Esteban Rossi-Hansberg

September 23, 2016

Contents

1 Description of the files

1.1 Original data files

- triangulated_sphere.off

  Contains the coordinate of the triangles that are used to approximate the Earth's surface for the Fast Marching algorithm (see below).

- natural_features.csv

  Aggregated data up to the $1° \times 1°$ grid cell level giving the fraction of smaller cells within cell r corresponding to rail, major road, other road or water as described in part 4.3.

- H0.mat

  180x360 matrix yielding the fraction of land per cell. (Recall each cell is $1° \times 1°$).

- C.csv

  180x360 table of the country indices per cell (per location).

- l.csv

  180x360 table of the population level (number of people per cell) in the data for year 2000 (G-Econ data).

- pop1.csv, pop5.csv, popminus5.csv, popminus10.csv

  180x360 tables giving the distribution of population levels (number of people per cell) in the data for years 2001, 2005, 1995 and 1990 respectively. These have been scaled up or down so that the world population remains the same as in 2000. Data for 2005, 1995 and 1990 come from G-Econ data, and data for 2001 is calculated such that the growth rate between 2000 and 2001 is 1/5 of the growth rate between 2000 and 2005.

- w.csv

  180x360 table of cell level GDP (in PPP) divided by cell population. The levels are normalized to the wage in Princeton, New Jersey.

- ubar.csv

  180x360 table of subjective utility measure for each country, based on the Cantril ladder (Deaton & Stone 2013).

1.2 List of Matlab running codes

- instantaneous_trade_costs.m
- fast_marching.m
- reduce_trmult.m
- ownmult_calc.m
- gd_a_tau_vect.m
- gd_m2_vect.m
- diffmult_calc.m
- main.m
- distelasticity.m

1.3 List of functions called

- initialize.m
- model.m
- results.m
- maps.m
- plots.m
- backward.m

2 Shaping the data

1. instantaneous_trade_cost.m

   Generates the $\varsigma(r)$ matrix of instantaneous trade costs for any location as discussed in Section 4.3

   (a) Loads `natural_features.csv`

   (b) Assigns parameter values for ς_{rail}, $\varsigma_{\text{no\_rail}}$, $\varsigma_{\text{major\_road}}$, $\varsigma_{\text{other\_road}}$, $\varsigma_{\text{no\_road}}$, ς_{water} and $\varsigma_{\text{no\_water}}$ from Allen and Arkolakis (2014)

   (c) Derives the resulting instantaneous cost of passing through location r as described in part 4.3

(d) Result : Matrix of instantaneous trade costs for any location on earth - `Sig.csv`

2. fast_marching.m

   Generates the matrix of bilateral trade costs (choosing the shortest path between two cells) as described in Section 4.3

   (a) Loads `H0.mat` and `Sig.csv`, as well as `triangulated_sphere.off`

   (b) Applies Gabriel Peyre's Fast Marching algorithm as mentioned in Section 4.3

   (c) Result : Four-dimensional matrix of bilateral trade costs between any two cells on the planet - `trmult_theta65_upsilon1.mat`, where `theta65` and `upsilon1` come from setting the required parameters θ and υ to 6.5 and 1 respectively.

   NB: In order to run this file, one needs to download the fast marching algorithm toolbox, Copyright (c) 2009, Gabriel Peyre. All rights reserved. It is available with a legal Matlab license on the Matlab toolbox website:
   `http://www.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching`
   This toolbox needs to be saved in the same folder as `main.m`. One needs to set a path in Matlab to both the whole folder "toolbox_fast_marching" and the sub-folder "toolbox", that is inside the former.

3. reduce_trmult.m

   Vectorizes the trade costs matrix

   (a) Loads `trmult_theta65_upsilon1.mat` and `H0.mat`

   (b) Reduces a four-dimensional object to a two-dimensional matrix

   (c) Keeps only the cells with positive mass of land

   (d) Result : Trade costs between emerged cells - `trmult_reduced.mat`

4. ownmult_calc.m

   Calculates a cell's cost of trading with itself

   (a) Calculates approximate average distance of crossing a cell along its diagonal

(b) Calculates the own trade costs from the distances computed in (a)

(c) Replaces the diagonal of the trade cost matrix with the own costs

5. gd_a_tau_vect.m

   Computes cell-level amenities (ratio $\frac{\bar{a}(r)}{u_0(r)}$) and initial productivity levels ($\tau_0(r)$) as described in section 4.4

   (a) Loads `H0.mat`, `l.csv`, `w.csv`, `trmult_reduced.mat`

   (b) Applies the algorithm described in Appendix B.7:

   - Solves equation (32) (corresponding to equation (60) in the appendix) to compute the ratio $\frac{\bar{a}(r)}{u_0(r)}$ (inner loop 1)
   - Uses equation (31) to compute the corresponding $\tau_0(r)$
   - Plugs the results in equation (24) to solve for corresponding $\bar{L}_0(r)$ (inner loop 2)
   - Compares with total population density in the data (outer loop)

   (c) Result : Distribution of the amenities/utility ratio and initial productivity per location - `a_H0.mat` and `tau_H0.mat`

6. gd_m2_vect.m

   Computes migration costs as discussed in section 4.5

   (a) Loads `H0.mat`, `a_H0.mat`, `tau_H0.mat`, `l.csv`, `pop1.csv`, `ubar.csv`, `trmult_reduced.mat`

   (b) Derives the initial country-specific utility level $u_0(c(r))$ from `ubar.csv` (using equation (36))

   (c) Derives pure amenities (constant per location) : $\bar{a}(r)$

   (d) Updates first period productivity $\tau_1(r)$ taking $\bar{L}_0(r)$ as given (using equations (8), (12) and (13))

   (e) Uses equations (7) and (24):

   - Taking $\bar{L}_1(r)$ as given computes $u_1(r)$ as a function of $m_2(.)$ (equation (7))
   - Uses equation (24) to solve for $m_2(.)$

   (f) Result : Distribution of the migration costs when entering location r - `m2.mat`

7. diffmult_calc.m

   Calculates the matrix of diffusion across cells with positive mass of land, as mentioned in Appendix A.4

(a) Loads `H0.mat` and sets parameters value for $\aleph$ and the earth radius

(b) Computes the diffusion cost $\exp^{\aleph\delta(r,s)}$ for two locations r and s in a four-dimensional matrix (separating longitude and latitude for each location)

(c) Reduces this object into a two-dimensional matrix saved as `diffmult_reduced.mat`

(d) This matrix will be used when generating counterfactual scenarios with spatial diffusion of technology

3 Simulation of the model

1. main.m

   Once the data are generated by the above running codes, `main.m` is the main file to run, as it will simulate the model forward and backwards, plot the wanted graphs and figures as well as store the output

   (a) Uses functions `initialize.m`, `results.m`, `backward.m`, and `plots.m`

   (b) Uses `initialize.m` to generate the global variables of the model

   (c) Uses `results.m` to run the model 600 periods forward and generate summary statistics

   (d) Uses `plots.m` to plot time series and maps

   (e) Uses `backward.m` to backcast the model for 180 periods and computes the correlation between data and model predictions about the population levels, log-levels, changes and log-changes at cell and country level for years 1995 and 1990

   (f) Generates the whole Output file which consists of: `realgdp_w`, `u_w`, `u2_w`, `prod_w`, `phi_w`, `PDV_u_w`, `PDV_u2_w`, `PDV_realgdp_w`, `migr_cell`, `migr_ctry`, `l`, `u`, `realgdp`, `tau`, `l_b`.

   NB: In order for the code to save the results properly, one needs to have two folders labelled "Output" and "Maps" in the same folder as `main.m`. Note that these folders are already included in the Replication file, where they are empty.

2. initialize.m

   `initialize.m` creates the necessary global variables of the model, and loads the needed data files

(a) Function : input = dummy for loading `trmult_reduced`, which in this model means the input into initialize should always be set to 1.

(b) Loads `a_H0.mat`, `tau_H0.mat`, `m2.mat`, `l.csv`, `pop5.csv` `popminus5.csv`, `popminus10.csv`, `ubar.csv`, `C.csv`, and `trmult_reduced.mat`

(c) Creates all the global variables of the code

(d) Manipulates country indices to get rid of missing indices

3. results.m

   `results.m` runs the model and creates several detailed descriptions of the results

   (a) Function : input = `H`, which is the matrix of land per cell and `T` which indicates the time period up to which the model runs (both are defined in `main.m`). Output = `realgdp_w`, `u_w`, `u2_w`, `prod_w`, `phi_w`, `PDV_u_w`, `PDV_u2_w`, `PDV_realgdp_w`, `migr_cell`, `migr_ctry`, `l`, `u`, `u2`, `realgdp`, `tau`

   (b) Calls function `model.m` to run the simulations

   (c) Derives correlations between data and the model predictions for levels, log-levels, changes (between year 5 and year 0) and log-changes for population at the cell level, country level, and for fertility adjusted data

   (d) Computes individual utilities `u`, `u2`, which is the expected utility from equation (4), and then world aggregates and present discounted values for the same variables

   (e) Computes world aggregate real income: `realgdp`, as well as the present discounted value

   (f) Computes world aggregate average productivity and innovation

   (g) Computes the time series of the share of world population migrating between cells or between countries

4. model.m

   Procedure running the simulation as described in Section 4.6 of the paper

   (a) Function : input = `H`, which is the matrix of land per cell and `T` which indicates the time period up to which the model runs (both are defined in `main.m`). Output = `l`, `w`, `u`, `realgdp`, `tau`, `phi`

   (b) Uses equation (53) to simplify $u_t(r)$ as $\hat{u}_t(r)$

(c) Applies the algorithm described in the proof of Lemma 3 (Appendix B.4) : uses equation (24) to get equation (51) and solves for $\hat{u}_t(r)$. Uses equation (52) to recover $u_t(r)$

(d) Derives $\bar{L}_t(r)$ using equation (7)

(e) Derives $\phi_t(r)$ using equations (12) and (13)

(f) Uses equation (23) to solve for $w_t(r)$

(g) Derives total real income (`realgdp`) using equation (22)

(h) Derives the domestic trade share (total share of trade happening within a country) using equations (12), (13), (14), (17), (19) and (21)

5. backward.m

   Simulates the model back in time in order to do comparisons with observed data as discussed in Section 4.7

   (a) Function : input = `H`, which is the matrix of land per cell and `T` which indicates the time period up to which the model runs (both are defined in `main.m`). Output = `l`, `w`, `u`, `realgdp`, `tau`, `phi`

   (b) Runs the model backwards using Section 4.7 : solves equation (40) for $\bar{L}_t(r)$, uses (39) to back out productivity, and derives the remaining equilibrium variables as in `model.m`

6. plots.m

   (a) Function : input = `H`, `l`, `u_w`, `u2_w`, `realgdp_w`, `prod_w`, `u`, `tau`, `realgdp`

   (b) Computes and plots growth rates for population, real GDP, utility and expected utility as well as log-variables. See Figure 5 and Figure 8

   (c) Computes and plots correlations for log-real GDP, log population and log-productivity. See Figure 4

   (d) Calls function `maps.m` to plot cell-level earth maps for population density, average productivity, utility and real GDP at periods 1, 200 and 600. See Figure 2 and Figure 3

7. maps.m

   (a) Function : input = vectors for population, utility, productivity and real GDP at the cell level, a time period t

   (b) The code is only considering log-variables

(c) Plots four maps for $\bar{L}_t$, u_t, $\left(T_t \bar{L}_t^{\alpha}\right)^{\frac{1}{\theta}}$ and y_t in period t

8. distelasticity.m

   Calculates the elasticity of simulated trade flows to distance

   (a) Loads `a_H0.mat`, `H0.mat`, `ubar.csv`, `C.csv`, `trmult_reduced.mat`, as well as `l`, `u` and `tau` from the model's output

   (b) Computes great-circle distances across cells

   (c) Computes trade flows across cells, nominal GDP and population of cells

   (d) Saves all the variables in `gravity.mat`

   (e) Regresses log trade flows on log distance, log origin GDP, log destination GDP, log origin population and log destination population, and stores the estimated coefficients in vector `b`