CODERS' CORNER

CSAE
CENTRE FOR THE STUDY OF AFRICAN ECONOMIES

HOW TO ANIMATE (NETWORK) GRAPHS IN MATLAB

In this post, we'll use MATLAB's imagesc command to create a colour map and then discuss how to create an animation of the resulting graphs. There are two aims of this post - (1) to discuss how networks can be represented using colour maps and; (2) to discuss how to animate the evolution of the network over time. I use the example of a production network but the code can be used to animate the evolution of any variable over time.

I will be using publicly available data that can be found here. This is from the 2016 Revision of the World Input Output Database covering 15 years, from 2000 to 2014, compiled by the University of Groningen. The data contain trade flows by country-sector.

Part 1: Importing and Storing the Network Data

To start, the code sets the directory and additional paths where the original data and command files are stored. Each data file contains the raw world input-output data for an individual year. I've named the data for each year as file15.xlsx starting from 2014 and counting down for previous year, i.e. for 2013, the file is named file14.xlsx and so on.

The raw imported data files for each year will be stored in the row vector **M** where each element corresponds to the input output matrix of a particular year.

```
numfiles = 15;
M = cell(1, numfiles);
```

MATLAB's default method of importing numbered data files is by character codes, rather than by following a "natural order sort" or an "alphanumeric sort"[1]. The **natsortfiles** command sorts the data by interpreting consecutive numbers as part of one integer, i.e. when ordering the files it treats '10' as the number ten, rather than as a one followed by a zero. This is so that we can work through the files in chronological order.

The first loop in the code imports and cleans each data file, looping over a scalar 'k' that specifies the file number corresponding to a particular year. In this case, k=1 corresponds to the year 2000. I extract intermediate input trade flows and the gross output for each country-sector, storing it in the **wiot_int** matrix and the **wiot_y** vector, respectively.

```
wiot_int = M{1,k}.data(1:2464,1:2464);
wiot_y = M{1,k}.data(end,1:2464);
```

[1] If we have files named file1, file2,…,file10, file11; MATLAB's default method would be to order the files as file1,file10,file11,file2. The natsortfiles command ensures the sorting looks as the former list.

Then, we can calculate each country-sector's technical coefficient – the dollar's worth of inputs from sector x per dollar's worth of sector y's output – to form the weighted adjacency matrix showing interactions between various sectors of the world economy. To do this, I divide the intermediate input demands by gross output, storing it in matrix **A**.

```
A = wiot_int./wiot_y;
```

For illustration purposes, I will limit the size of the adjacency matrix China's intermediate input trade in 35 sectors (data is available for a total of 56 sectors). I also set trade flows of less than two cents, between any two sectors, to zero. The final adjacency matrix that will be pictorialized is contained in the three dimensional matrix **all_adj** with year being the third dimension. This matrix stores China's input-output flows for all 15 years.

```
A_illu = A(393:428, 393:428);
A_less2cent = A_illu < 0.02;
A_illu(A_less2cent == 1) = 0;
all_adj(:,:,k)= A_illu;
```

Now that we have cleaned and constructed the data of interest, we turn to the second part of the code which will produce the animated colour maps of the **all_adj** matrix.

Part 2: Generating and Animating Colour Maps

The **VideoWriter** command creates the .mp4 file which will store the animation, titled as **'alladj.mp4'** in the accompanying code. **v** is the name we specify to call the videowriter later in the code. We can also choose the frame rate – i.e. the number of frames (graphs) we want to display per second. Then, we open the videowriter with the command line **open(v).** Notice that this is specified outside of the second loop which loops over time t.

```
v = VideoWriter('alladj.mp4', 'MPEG-4');
v.FrameRate = 2;
open(v);
```

The second loop plots the adjacency matrix for each year separately (called from all**_adj(:,:,t)**) and plots it as a colour map. MATLAB's imagesc command displays the data contained in a matrix array as a colour map, with each element of the matrix appearing as a pixel in the image. The **colormap** command can be used to specify the colour scheme used, and **colorba**r, which maps data values into the colourmap. The **caxis** command ensures that the colour scaling in the colour bar is consistent across years, ranging from 0 to 0.45.

```
imagesc(all_adj(:,:,t));
colormap(jet);
caxis([0 0.45]);
colorbar;
```

The following lines in the code add the title, x-axis, and y-axis labels. The **text** command allows you to add additional strings to the plot, here I add the year to which each plot refers. The first two inputs are the x and y coordinates where the text will be placed. It's a matter of trial and error to get the text positioned to your liking. Additional text options are available to specify the font size, font colour etc.

```
str = sprintf('China Input-Output Network');
title(str)
ylabel('Input sectors');
xlabel('Output sectors');
text(26,2, [' Year= '  (num2str(1999+t))], 'FontSize', 18, 'Color','white');
```

Each frame is then written to **Q(t)**, where t is indexing the year to which each plot corresponds. The last line of the loop ends with **writeVideo** where the two inputs are **v** – where we stored the videowriter, and the second input is where we stored each frame corresponding to t.

```
Q(t) = getframe(gcf);
writeVideo (v, Q(t));
```

Once we end the loop, all that is left is to close the videowriter with the command line **close(v)**.

```
close(v)
```

You should get an .mp4 file as your output stored in the current folder set. This mp4 file can be converted to a gif using free mp4 to gif converters on the internet. You can play and pause the resulting animation "animation.mp4", attached with this document, to see how the Chinese input-output network of sector-to-sector trade changes over time.

Diana Beltekian, PhD Candidate in Economics, University of Nottingham
12 November 2020

Data citation:

Timmer, M. P., Dietzenbacher, E., Los, B., Stehrer, R. and de Vries, G. J. (2015), "An Illustrated User Guide to the World Input–Output Database: the Case of Global Automotive Production", Review of International Economics., 23: 575–605