NITP 2013 GRAPH THEORY LAB
Steffie Tomson

DOWNLOADING TOOLBOXES

Browse to:
https://bitbucket.org/gastats/monet/downloads
Click the file → gastats-monet-3bd316c81487.tar.gz to download, and save it to your Downloads folder.
Click the file → TestABIDE.mat to download, and save it to your Downloads folder.
Browse to: https://sites.google.com/site/bctnet/
Click the file → BCT.zip to download, and save it to your Downloads folder.
Open Terminal (press Command+Spacebar to search Terminal)
Change directory (cd) to your Downloads folder and type:

```
>>tar –xvf gastats-monet-3bd(press Tab here for it to auto-complete)
>>unzip BCT.zip
```

Two new extracted folders (called gastats-monet-3bd316c81487 and 2012-12-04) should appear.
Move the TestABIDE.mat file into the gastats folder by typing:

```
>> mv TestABIDE.mat gastats-monet-3bd316c81487/
```

Open Matlab.
Go to File, Set Path (if you're in 2013, Set Path is a button on the HOME ribbon)
In that window, click → Add with Subfolders
Browse to your Downloads folder and select the folder → gastats-monet-3bd316c81487
Browse to your Downloads folder and select the folder → 2012-12-04

Now you should have everything in your Matlab path, and you can start running functions!
From now on we'll be working entirely in Matlab.

CREATING YOUR MATRIX

We have given you some preprocessed data from the publicly-available ABIDE dataset.

```
>>load TestABIDE.mat
>>size(TestABIDE)

ans =

   120    50    21
```

You'll see that the dimensions are 120 timepoints by 50 regions of interest by 21 subjects. This is the type of data that you can input to the toolbox.

Let's see what the raw data looks like. Feel free to copy/paste the code, or type it yourself. Whichever works with your timeline/level of interest.

```
>> plot(TestABIDE(:,1,1)); hold on; plot(TestABIDE(:,10,1), 'g'); hold on;
plot(TestABIDE(:,20,1),'r')
>> title('Regions 1, 10, and 20 from subject 1')
```

What you see is a figure showing your one timeseries from three regions of interest from one subject. You can plot all (or none) of the data, but I always make it a habit to visualize at least some of the timeseries just to check that everything looks OK. This looks OK to me.

GROUP ANALYSIS

We will be using the data file TestABIDE.mat. Let's say that there are only 6 subjects in the TestABIDE dataset, and we want to group them into the first 3 and the next 3 subjects.

Run…

```
>>[finalgrphsG lambdasG pathgraphsG
lambdarangeG]=monet_group(TestABIDE(:,:,1:6), [3,3], 0,0,1);
```

This will take about 5 minutes to run.

While you're waiting:
This function is creating a Markov network (inverse covariance matrix) from your timeseries data. What comes out will be a thresholded graph of *direct* connections between all of your nodes. The method that we use to threshold involves an L1 penalty, which shrinks elements closest to zero more than those farthest from zero. The edges that are left technically still have weights, but because each edge has been non-linearly shrunken, it can't be used to interpret the strength of connectivity. For this reason, all graphs that come out of MONET should get binarized (1=connected, 0=not connected) for display and graph theory calculations. For more information on the thresholding method we have chosen (called graphical lasso), see:
http://www-stat.stanford.edu/~tibs/ftp/graph.pdf

Read through the following descriptions of inputs and ouputs that the function will deliver to your Matlab workspace.

Output #1 – **finalgrphsG** will be a cell containing final graphs for each of your subjects (or groups, depending on which function you run). In this example, we are estimating graphs for the two groups. FinalgrphsG{1} will be our final 50x50 connectivity matrix for group 1, finalgrphsG{2} is the final matrix for group 2.

Output #2 – **lambadsG** is a cell that contains the final threshold value that was selected through the stability selection method we use in Monet. In our example, lambdasG{1} is the threshold value selected for group 1, lambdasG{2} for group 2, etc… If you're doing a subject analysis, lambdas{1} is the threshold selected for subject 1, lambdas{2} for subject 2, etc…

Output #3 – (optional) **pathgraphsG** is a cell that contains an estimated graph for each of the 30 lambda values that were tested. In our example, pathgraphs{1} is a 50x50x30 matrix. This contains one 50x50 matrix for each of the 30 lambdas tested.

Output #4 – (optional) **lambdarangeG** is a cell that contains all 30 lambda values tested for each subject or group. In our example, lambdarangeG{1} is a 1x30 vector list of the 30 lambda values tested for subject 1.

Input #1 – **DataMatrix** is the only required input to this function. It must be at least a 2D matrix where the first dimension is timeseries, the second dimension is your set of ROIs, and the 3rd dimension is your list of subjects. In our example, we have a timeseries consisting of 120 timepoints taken from each of 50 regions of interest for 21 distinct subjects.

Input #2 – (optional) **Whiten** allows you to choose whether you want to whiten the signal data before estimating the Markov network. In a nutshell, whitening reduces autocorrelation within a set of signals while preserving important aspects of the signal by equalizing the power spectrum. Many neuroimaging software programs already have a pre-whitening option, so you might not need to use this. 0=no whitening, 1=whiten please

Input #3 – (optional) **FullPath** tells the program to plot the range of all graphs at different lambda values. The lambda value is **not** a hard threshold, but rather a soft threshold. This means that each value in the matrix is shrunken by value lambda, and each value could be shrunken by lambda a different number of times. Each algorithm estimates this differently. Graphical lasso is one such algorithm, and it is the one we implement here. 0=I don't want to see all 30 graphs at different lambda values, just show me the graph that fits the data best. 1=I'd like to see all 30 graphs please.

Input #4 – (option) **DispFig** tells the program to plot the final output graphs for you. Meaning, it will plot the final matrices that have been estimated and thresholded at the appropriate lambda value.

Note that the input structure for monet_group requires that when you make *your* matrix, all subjects in a given group must be grouped together. Also note that we've added a new input between DataMatrix and Whiten for monet_group. The input [3,3] tells the program that you want to divide your first 6 subjects into two groups: first three subjects and the last three subjects. If you just wanted to treat subjects 1:6 as their own group and estimate that graph only, your second input would simply be [6].

Once the function has finished running, you should see 4 new variables in the workspace. FinalgrphsG contains two graphs: one for each group you specified. These two graphs are plotted next to one another in a figure for you.
If you wanted to plot these yourselves, you would:
```
>>imagesc(finalgrphsG{1}~=0); axis square;
```
This command above plots all values that are not zero, which represents all edges in your matrix that survived the thresholding procedure.

To see the lambda values that were chosen for each graph, you can simply type
```
>>lambdasG
```
to see the variable lambas. LambdasG{1} is the threshold selected for FinalgrphsG{1}; LambdasG{2} for FinalgrphsG{2}.

LambdarangeG{1} contains the range of lambda values tested for FinalgrphsG{1}.

PathgraphsG{1} contains a 50x50x30 matrix, which represents one 50x50 matrix for each of the 30 lambda thresholds tested. You don't necessarily need to use this data, but it's there in case you want to look at how the final graph changes with each threshold.

Let's look at some of the data.

Enter the following code:

```
>>subplot(2,2,1); imagesc(pathgraphsG{1}(:,:,5)~=0); axis square;
subplot(2,2,2); imagesc(pathgraphsG{1}(:,:,10)~=0); axis square;
subplot(2,2,3); imagesc(pathgraphsG{1}(:,:,22)~=0); axis square;
subplot(2,2,4); imagesc(pathgraphsG{1}(:,:,28)~=0); axis square;
```

Now we are looking at a single figure that shows how the graphs change with threshold. But how does our final graph look in comparison?

```
>> lambdasG

lambdasG =

    [0.0556]    [0.0478]
```

We want to find the graph thresholded at .0556. If you look at lambdarangeG{1}, you'll see that .0556 is the 19th element. That means that if we plot the 19th graph in pathgraphsG, we'll see our final matrix again.

```
>> figure; imagesc(pathgraphsG{1}(:,:,19)~=0); axis square;
```

There you have it! We've plotted a selection of the thresholded graphs, plus our final graph that we can use to calculate graph theory in the future.

For subject-level analyses skip to the end. For now, let's use this data and calculate some graph theory metrics.

CALCULATING GRAPH THEORY METRICS

Now that we have our group graphs, we want to calculate graph theory metrics. To do this, we'll use the Brain Connectivity Toolbox (the BCT.zip file that you downloaded earlier).

First,

```
>>Group1=finalgraphsG{1};
>> Group1(Group1~=0)=1;
```

We just binarized the matrix such that any metric that survived (i.e. greater than zero) is now equal to 1. We do this because our thresholding procedure shrinks values in the matrix to the point where the weight is no longer informing the relationship between regions. All we care about is that they are connected.

First, let's start with a simple metric, Degree. How many nodes is my node of interest connected to? The BCT has two available functions: degrees_und.m and degrees_dir.m. The 'dir' and 'und' suffixes determine whether you're inputting a directed or undirected data matrix. We are using undirected data (i.e. no directional information), so:

```
>>degreeGroup1=degrees_und(Group1);
```

You will see that degreeGroup1 is a 1x50 vector containing the degree of each node.

Let's try calculating clustering coefficient. If you cd to the BCT folder (Downloads/2012-12-04), you'll see that there are actually 4 functions for calculating clustering coefficient. You only need to use one, clustering_coef_bu. The suffixes BD, BU, WD, and WU stand for binary directed, binary undirected, weighted directed, and weighted undirected, respectively. Be sure to use the appropriate functions on your own data.

We are using undirected data, so:

```
>>ccGroup1=clustering_coef_bu(Group1);
```

Again, the output is a 1x50 vector containing clustering coefficient values for each of the 50 nodes.

Those are the basics! Explore some of the other functions like betweenness centrality or global efficiency. Use the help files to guide you (>>help clustering_coef_bu).

Run the same metrics for Group2 and see how the results compare.
This is the time when I would recommend implementing permutation testing to determine whether the degree of node 10 in group 1 is significantly different from the degree of node 10 in group 2. MONET does not have this functionality, but there are several toolboxes out there for Matlab and R that can do testing for you.
http://www.mathworks.com/help/bioinfo/ref/mattest.html
http://courses.washington.edu/matlab1/Bootstrap_examples.html

Play around with some of the other functions in the BCT. To visualize some pre-analyzed datasets, move onto the next section.

Before you go, start this subject analysis running in the background:

```
>>[finalgrphs lambdas pathgraphs
lambdarange]=monet_subjects(TestABIDE(:,:,1:2), 0, 0, 1);
```

VISUALIZATION

Open a browser and go to:
http://umcd.humanconnectomeproject.org/

Click 'Analyze Network', Analyze Normal Network

This site allows you to view network statistics that will be calculated on data that you can upload to the site. Alternatively, you can view graph theory has already been calculated from previously-uploaded data. You can also play with all the other datasets available on the database and see how the graph theory metrics compare across different populations.

Go to Study Name and select OCD_and_control
Under Network Name, select CON_17_mean
Weighting scheme: Binary (though choose weighted next to see what happens)
% edges: Choose 30 here.

When the results page loads, you will see information about the study group on the left, under Network Information. Global Network Metrics on the right gives you your average clustering coefficient, number of components (modules), global efficiency, etc… Have a look at these values and the resulting graph below.

In another window, Select OCD_and_control, but this time select OCD_19_Mean. Compare the results from the two groups.

What do you think these differences mean?

Check out some of the other very interesting datasets already uploaded onto the site. Perhaps open two windows so you can compare controls with patients side-by-side. Notice that you can download all of your metrics into a .txt file, our output everything to a .pdf.

This toolbox utilizes the Brain Connectivity Toolbox to calculate graph theory metrics. You could also download the toolbox yourself and play with the values live on your own computer (as we did earlier).

Do if you have time

SUBJECT-LEVEL ANALYSIS

Let's estimate some of your individual subject graphs to see what they look like using the MONET toolbox.

In this case, we are doing a subject analysis, so we'll use monet_subjects.

Run:

```
>>[finalgrphs lambdas pathgraphs
lambdarange]=monet_subjects(TestABIDE(:,:,1:2), 0, 1, 1);
```

This will take a few minutes to run. Monet_subjects will estimate individual graphs for the first 2 subjects, plot them, and output the lambda values. You can easily run all individual subjects, but this algorithm takes a while without the parallel processing toolbox, so for this demo we're just going to look at the first 2 subjects.

When your data has finished running, you should get a figure that pops up and shows you the graphs estimated for each of your subjects. Finalgrphs{1} and finalgrphs{2} are the variables that contain your two final graphs. You could plot them yourself by executing:

```
>>imagesc(finalgrphs{1}~=0); axis square
```

If you want, you could calculate graph theory metrics on these individual subject graphs and compare them to the groups we ran earlier.

Feel free to email me with any questions:
steffietomson@gmail.com