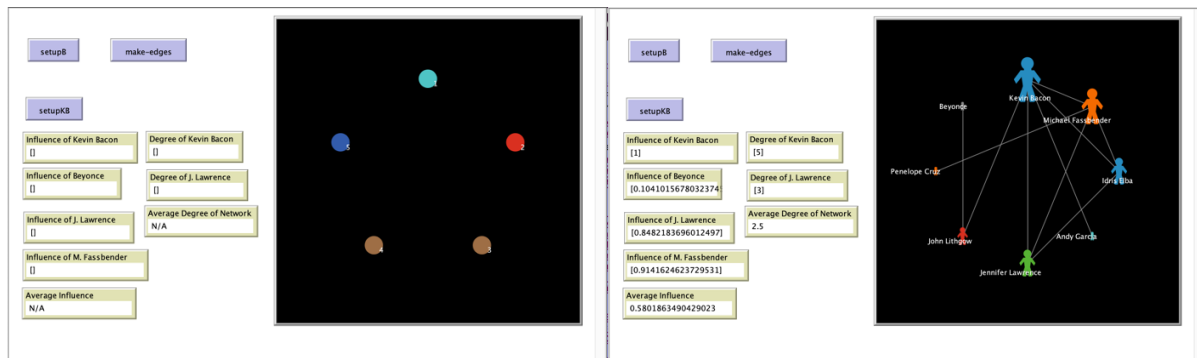


CS108L Computer Science for All

Module 8: Guide Networks Milestone 1

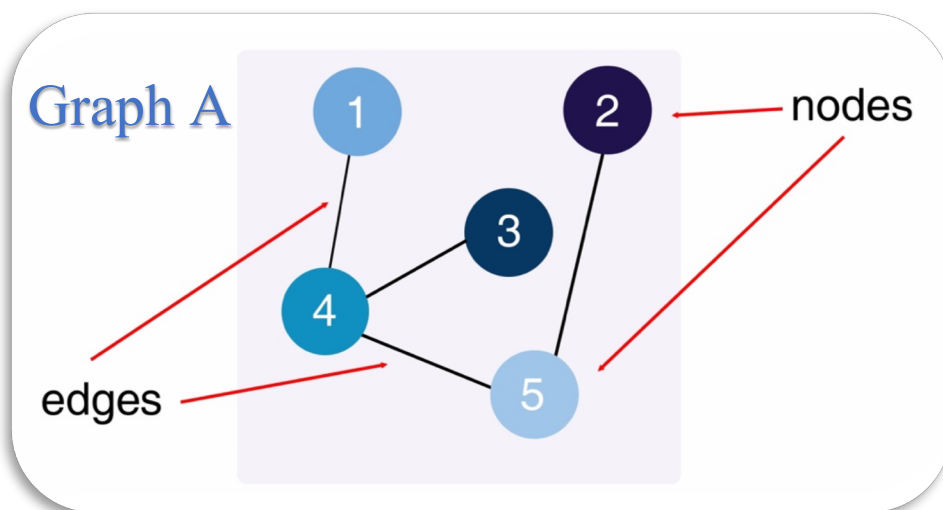
The Basics and Kevin Bacon Game



In this module you will create 2 independent networks, a basic one and one used for the Kevin Bacon game. In doing so you will learn about networks in general as well as NetLogo labels, turtle layouts, a new type of agent called a link and the foreach loop.

So, what are networks?

Networks are a collection of points joined together by lines (Newman 2010). We can think of a network as a **graph**. The points on the graph are called **nodes** or **vertices**; the lines are called **edges**. A network can take many shapes, but the general form is something like this:



Graph A is an **undirected network**. That means that if there is an **edge** between two **nodes**, you can travel between them. For example, we can travel directly between node 1 and node 4, and back from node 4 to node 1. But to get from node 1 to node 3, we can't go straight there. We must go through node 4 first. This is called a **path**: in this case, the path is 1 – 4 – 3.

EXAMPLE: Consider this picture of flights between different cities in Eastern USA.

Q: Why does this matter?

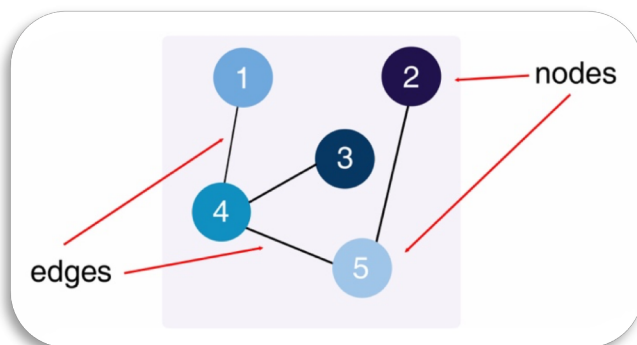
A: Networks are everywhere in real life. They help us to study objects and the relationship between them.



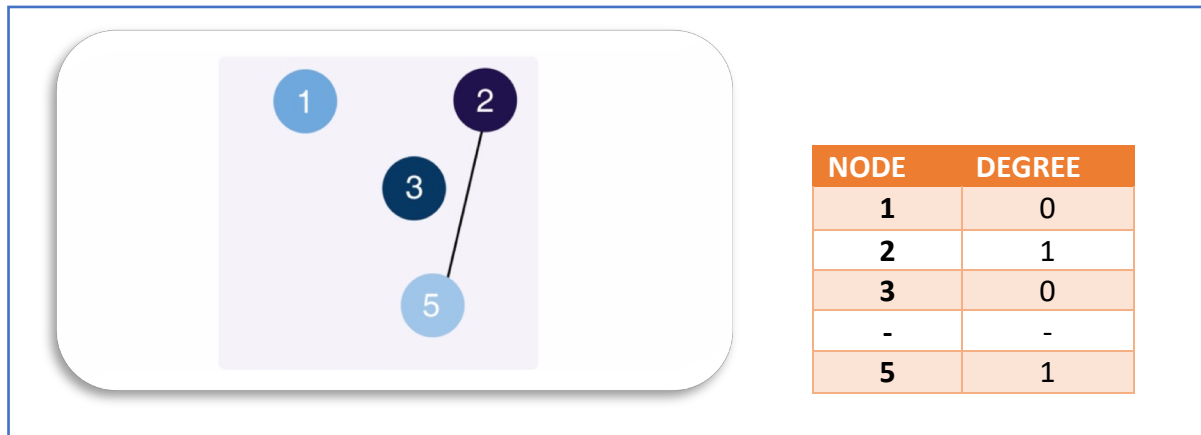
If you wanted to go from Indianapolis to Baltimore, and you could only take the flights pictured here, you would have to fly through Atlanta. Observe how much longer of a trip that is than flying directly from Indianapolis to Baltimore.

What would happen if Atlanta airport was closed for bad weather?

Let's look at Graph A again. We can reach any node on Graph A from any other node by following the edges. We can say that Graph A is **connected**. This is like the example above: we can reach any city from any other city, but we might have to travel through another city to get there.



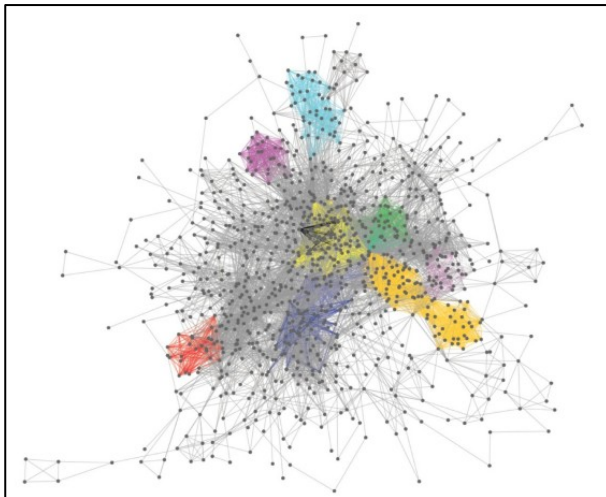
The **degree of a node** is the number of edges connected to that node. **Node 4 has the highest degree.** Node 4 in our graph is like Atlanta on the flight map, with a lot of connections coming in and going out. What happens if we remove node 4 from the network?



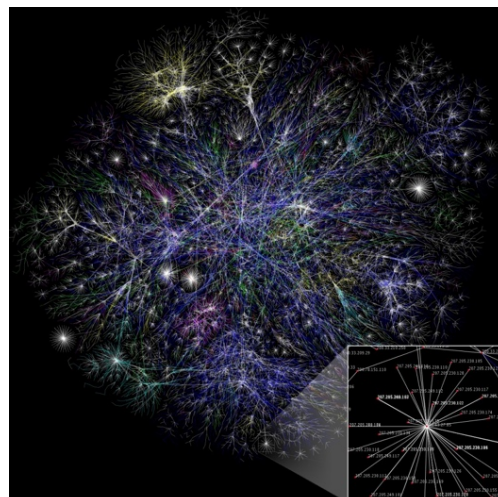
When we remove node 4, we must also remove any edges that go to 4. Now, you can only travel between node 2 and 5. **The other nodes are unreachable.**

That would mean a lot of people stranded at the airport!

Networks aren't just used to describe flight schedules though! Here are some more examples, both man-made and natural:



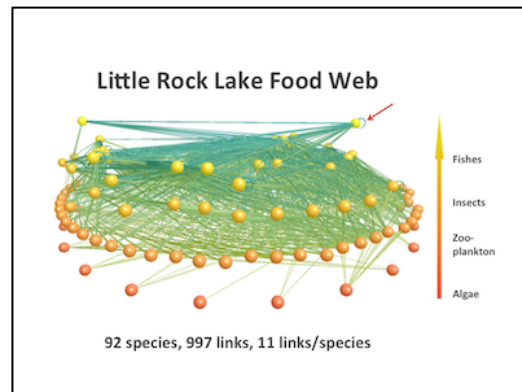
The metabolic network of a genus of green algae.
(NYUAD)



The structure of the Internet.



A social network.
(Aquila Style)



A food web showing who eats who in Little Rock Lake, Wisconsin. (Complexity Explorer)

The Kevin Bacon Game

Have you ever run into a friend somewhere you didn't expect to? Have you ever met someone new and found out you had friends or family in common? When this happens, we often say

“It's a small world!”



(ericwrobbel.com)

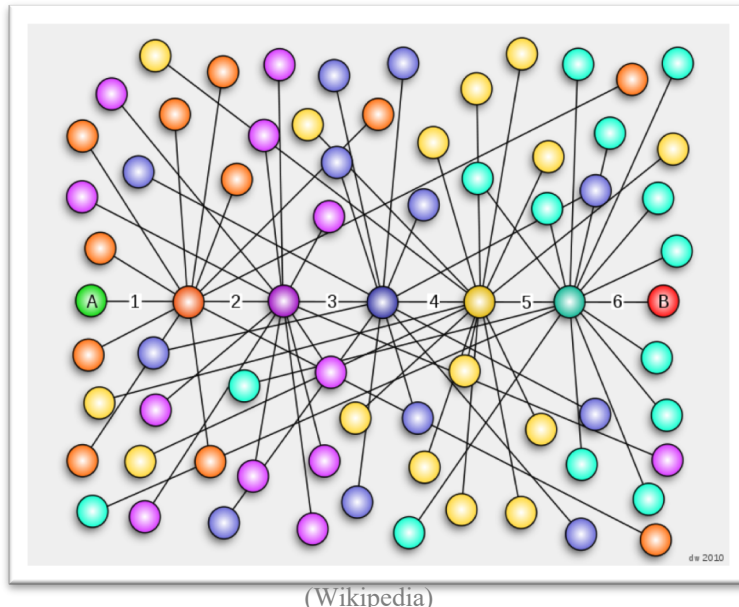
Well, there's a network for that!

Small world networks are a type of network used to model a social group or social connections.

Q: Why are they called small world networks?

A: In a small world network, the path from any given node to any other node will be relatively short. In other words, the mean path length is short.

The popular idea of the **six degrees of separation**, first explored by psychologist Stanley Milgram in the 1950s, describes human society as a small world network where the mean path length is six. (If you're not familiar with the idea of the **six degrees of separation**, click [here](#) to learn more.)



TRY IT!

Look at the graph to the left. Is there a path longer than 6 steps?

- ☐ Pick any starting node in the graph/network on the left.
- ☐ Now pick an ending node.
- ☐ Count the number of edges between them. This is the length of your path.

The **Kevin Bacon game** came out of the idea of the **six degrees of separation**, and is sometimes called the **Six Degrees of Kevin Bacon**. Given any actress or actor, the goal is to link that person to Kevin Bacon in the least amount of steps possible. The number of steps is called someone's **Bacon number**. It is almost impossible to find anyone with a Bacon number higher than six!

Google has a hidden Bacon number feature! Search the name plus "Bacon Number."

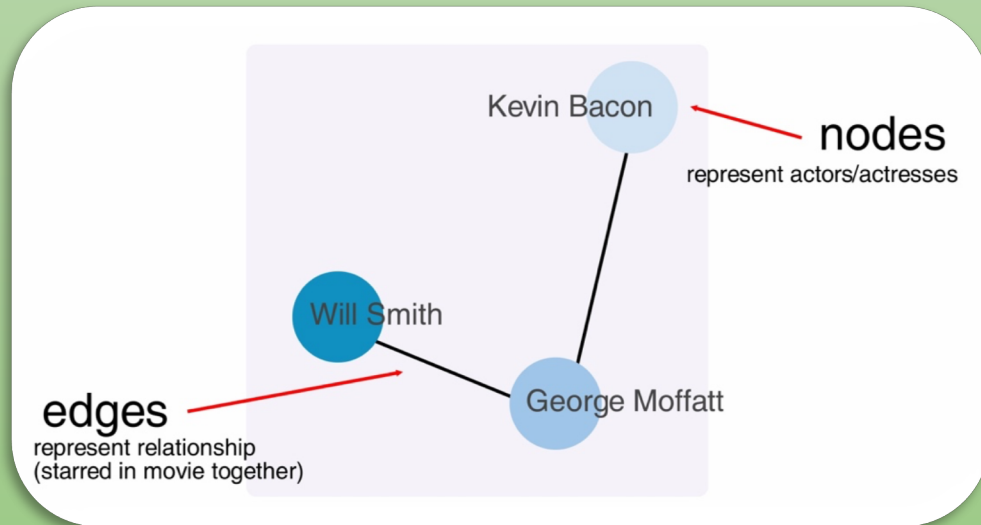
EXAMPLE 1: What is Will Smith's Bacon number?

Will Smith's Bacon number is 2

Will Smith and George Moffatt appeared in *The Pursuit of Happyness*.

George Moffatt and Kevin Bacon appeared in *Quicksilver*.

EXAMPLE 2: Make a graph of **Example 1**.



Influence of a Node (Eigenvector centrality)

Eigenvector centrality is a complicated name for a simple concept: if your friends have a lot of **influence**, then you probably do too. We'll just call it **influence** from here on. Influence is a decimal value between 0 and 1. A **value of 0** indicates that a node has **minimum** influence on a graph, and a **value of 1** indicates that a node has the **maximum** influence on a graph.

Fortunately, NetLogo has an extension, the network extension [nw], that will calculate influence for us automatically, which we will show in the NetLogo section below.

Why do we care about influence?

If we can identify influential nodes in a graph, we can learn more about these types of situations:

- Who starts fashion trends among friends?
- Which country's economy has the most effect on other countries in the region?
- If there's an outbreak of a virus, who's the most at risk of spreading it to other people?
(You will model a virus outbreak in the next lesson!)



How to do it the NetLogo way.

Labeling in NetLogo.

A label is just a way to identify an object with text. In Netlogo, turtles may be labeled using their label variable which takes a string as input. A string is the way programmers refer to text in their code. You worked with strings before when you change the shape variable from the default arrow shape to something else.

So if I want to call turtle 0, “turtle 0” in the world we could do the following;

```
ask turtle 0 [  
  set label "turtle 0"  
]
```



Which produces adds the label to the turtle like above. Just like setting color, shape and heading you may set the label to any text anywhere in your program when you are referencing the turtle agent (using ask turtles, create-turtles, sprout, hatch, etc..).

Layouts in NetLogo.

A layout is just a way of organizing turtles in the world into a predetermined formation. Rather than specifying where each turtle is placed you may use a layout that automatically places all the turtles into a certain formation. There are 3 layouts that NetLogo currently supports;

Circle Layout: <http://ccl.northwestern.edu/netlogo/docs/dict/layout-circle.html>

Radial Layout: <http://ccl.northwestern.edu/netlogo/docs/dictionary.html - layout-radial>

Spring Layout: <http://ccl.northwestern.edu/netlogo/docs/dict/layout-spring.html>

For this assignment we will be working with the circle layout. To use the circle layout, just specify the layout, the agents set affected, if it is sorted and the radius of the circle as shown below.

```
layout-circle sort turtles 5
```

Links in NetLogo.

A link in NetLogo is an agent, just like turtles and patches. It ties 2 turtles together. If you move the turtles, the link between the two of them will remain. As agents, turtles have several variables associated with them (breed, color, size, etc..) however we will just be using them to show the whether or not a turtle is connected to another turtle in the network. You may see all the commands associated with links by going to the NetLogo Dictionary and scrolling to the Link section.



Link Breeds

Creating a breed of link is slightly different from creating a breed for turtles because you need to specify if the link is a directed link (if traveling down the link you may only go in 1 direction) or an undirected link (may travel in either direction down the link). A directed link breed is created

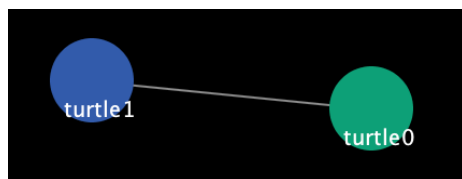
```
directed-link-breed [lines line]
```

as above. An undirected breed would just swap out the word directed for undirected. After the breed is setup it works the same as turtle breeds accept it is a link agent instead of a turtle agent.

Linking Between two Turtles

There are many ways to link between two turtles, which one you use depends how you are referencing the turtles. They take the form of create-link-(to or from or with). You should refer to the NetLogo dictionary for a complete set of examples but if you are working labels to identify the turtle. You might create a link between two turtles labeled “turtle 0” and “turtle 1” this way.

```
ask turtles with [label = "turtle0"] [create-link-with one-of turtles with[label = "turtle1"]]
```



Which produces something similar to the above.

Finally, any turtle linked to another turtle becomes a ‘neighbor’, so you may determine the **degree** of a turtle by simply counting the ‘link-neighbors’.

```
[count link-neighbors] of people with [label = "Wayne"]
```

Or if you are talking directly with the turtle (using ask) you may use the keyword ‘self’.

```
[count link-neighbors] of self
```

NOTE: Links take the shortest path to another turtle. Because of this, if wrapping is on, the links may exit one edge and enter another to get to the linked turtle. This can be confusing so it is recommend to turn wrapping off when working with networks.

The Foreach Loop

A foreach loop is a way to loop through a list or agent set. In a list or set each entry is called an ‘item’ in NetLogo (element is also a common way of referencing them). Thus a foreach loop just



means: Select the first item from the list/set, do something to or with that item, then select the next item and perform the same action on it. Keep doing this until each item in the list/set has been selected. Hence you may say “For each item in the list/set do something to/with it” which leads to the term foreach loop.

Foreach loops in NetLogo are a bit tricky because they use some ‘hidden’ procedures and shorthand references to do the work. However, once we go through this each item, everything should become clear.

Let’s say I have a list of names Aaron, Angela, Jannatul and Melanie. I want to create a turtle for all of them and label them appropriately. I can do that with a foreach loop.

First create a list with the names in it;

```
let names ["Angela" "Aaron" "Jannatul" "Melanie"]
```

Then loop through the list, create a turtle and give the turtle the label according to the list. Since a turtle is created, give it some settings making it size 2 and a circle shape. The code for that would look like this:

```
(foreach names [ [name] ->
  create-turtles 1 [
    set label name
    set size 2
    set shape "circle"
  ]
])
```

Lets address each item of the code individually.

- The parenthesis are there to ensure NetLogo knows that everything between them is part of the foreach loop. Sometimes NetLogo makes a mistake about this so enclosing the whole foreach loop (including the words foreach) make it clear.
- ‘foreach’ signals Netlogo it is a foreach loop just like ‘repeat’ signals to use a repeat loop
- The foreach takes 2 arguments, the list/set (in this case ‘names’) and what you are going to do in the loop enclosed by the first []. This is similar to the repeat loop where you have the first argument as a # and what you do in the [].
- ‘name’ is a variable that contains the value of each item of the list/set as it cycles through. You may pick any variable identifier (you don’t have to use ‘name’).

- \rightarrow is NetLogo's form of anonymous procedure. For the foreach loop all it means is set the variable before the \rightarrow to the item and in the list then do everything that follows \rightarrow .
 - In this case, it would first set name to "Angela" then create a turtle and give it the label "Angela", set its size to 2 and set its shape to "circle". Since it is in a foreach loop this is repeated until you run out of items in the list/set.

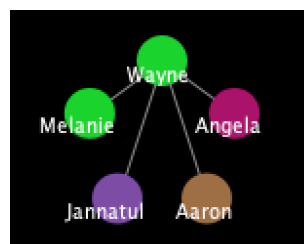
If using a circle layout the output might look like the below:



Now let's say we have another turtle with the label "Wayne" and in addition we want to create a link to "Wayne" from each of the other turtles' labels. We know how to create links from labels from above, since we are creating the above turtles already, let's just link them up as we create them.

```
(foreach names [ [name] ->
  create-turtles 1 [
    set label name
    set size 2
    set shape "circle"
    create-link-with one-of turtles with [label = "Wayne"]
  ]
])
```

We just add the create-link-with and select the turtle we want to link with by the label to the above code and we end up with something that looks like this.

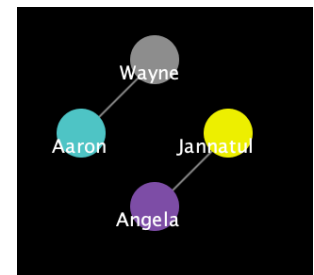




A NetLogo foreach loop is not limited to working with just one list/set. It may cycle through multiple list/set simultaneously. However, each list/set must be equal in length.

Given a list containing Angela and Aaron called graders and another list containing Jannatul and Wayne called instructors we can write a foreach loop to connect Angela to Jannatul and Aaron to Wayne using the multiple list method. For this example, the turtle Jannatul and Wayne will exist prior to using the foreach loop so we don't have to create them there, however we could if we wanted too.

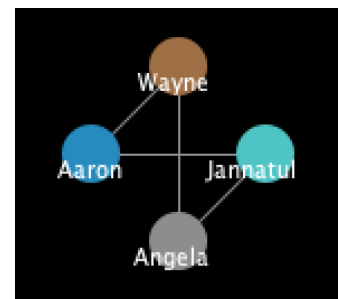
```
(foreach graders instructors [ [grader instructor] ->
  create-turtles 1 [
    set label grader
    set size 2
    set shape "circle"
    create-link-with one-of turtles with [label = instructor]
  ]
])
```



Here the foreach loop takes the first 2 items from the list (Angela and Jannatul) held in the grader and instructor variable and creates the Angela turtle then links it with the already existing Jannatul turtle (note that the turtles must exist before linking or you get a 'found nobody' error. Then it goes to the next two items (Aaron and Wayne) and does the same thing, after that there are no more items in the list so the loop stops.

The final use of a foreach loop is as a nested loop (a foreach loop within a foreach loop). Taking the two list above, you could link Angela and Aaron with both Jannatul and Wayne.

```
(foreach graders [ [grader] ->
  ask turtles with [label = grader] [
    foreach instructors [ [instructor] ->
      if (grader != instructor) [
        create-link-with one-of turtles with [label = instructor]
      ]
    ]
  ]
])
```



This loop assumes the grader turtles have already been created and then selects the first item from graders (Angela), and the next foreach loop selects each element from the instructor list and links it back with Angela if the two items of the list don't match (we don't want to link to ourselves). Then the next item in the grader list is selected and the process repeated. Note that



while we only have two items in each of this list, they do not have to be the same size as in the previous example. This is because each list is cycled through independent of the other one.

The Network Extension

The network extension is used to calculate the eigenvector of centrality or influence of a node in a graph (a turtle in our case). This is a mathematical computation on matrices. Fortunately we don't need to learn the math behind it because the network extension does it for us!

To use any extension, at the top of the code section you need to add the extension (where you put globals and breeds). This is done simply by typing extensions [<name of extension>]. In the networks the name has been simplified to nw.

```
extensions [nw]
```

Then to use an extension you begin by typing the extension name, colon, and the procedure you are using from the extension. Since the network of influence procedure is independent of the agents we work with we need to use 'of' and turtles (or a turtle breed) and 'with' to define exactly what agents we want included in the calculation. This leaves us with the following code, which may be used within a program or in a NetLogo monitor.

```
[nw:eigenvector-centrality] of turtles with [label = "Wayne"]
```

The Mean

The mean is just another more formal way of saying 'average'. This is a built in NetLogo method. To use it just type mean and then a list of numbers. NetLogo will automatically make a list if you calculate a number for several turtles at once.

```
mean [ nw:eigenvector-centrality] of turtles
```

The above calculates the influence for each turtle and puts them in a list, then the mean finds the average value of the list. If you want to calculate the average degree you will need to use the formula for calculating degrees in the link section of this guide.