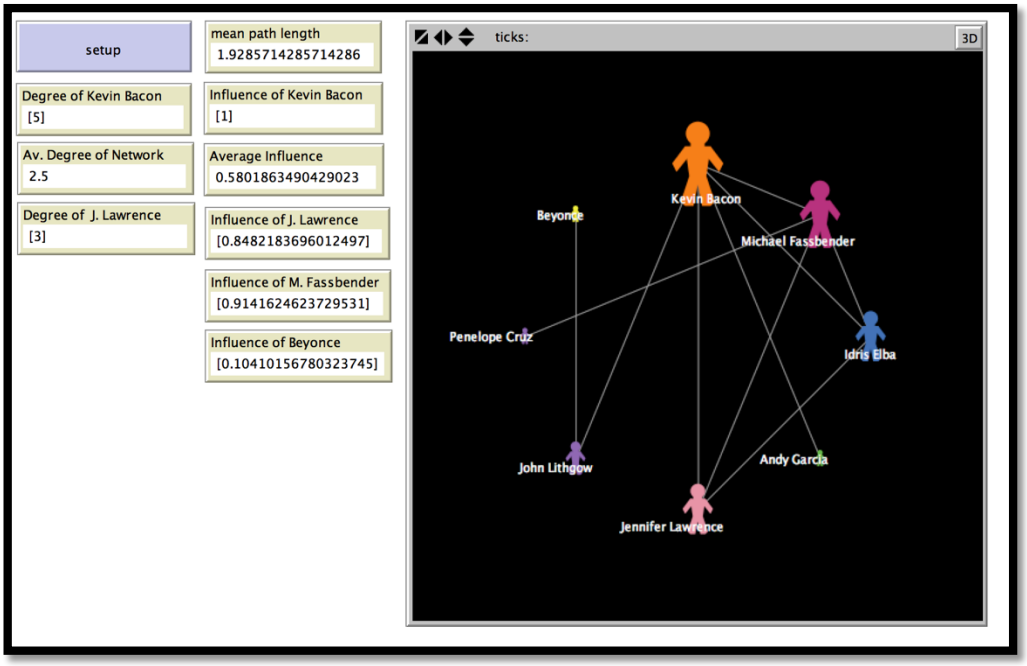
The Kevin Bacon Game



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)

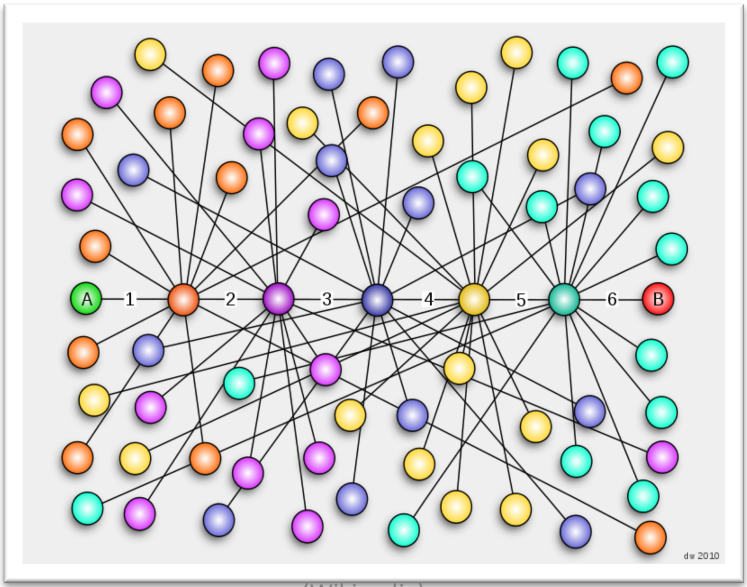
Well, there's a network for that!

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.)



(Wikipedia)

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.

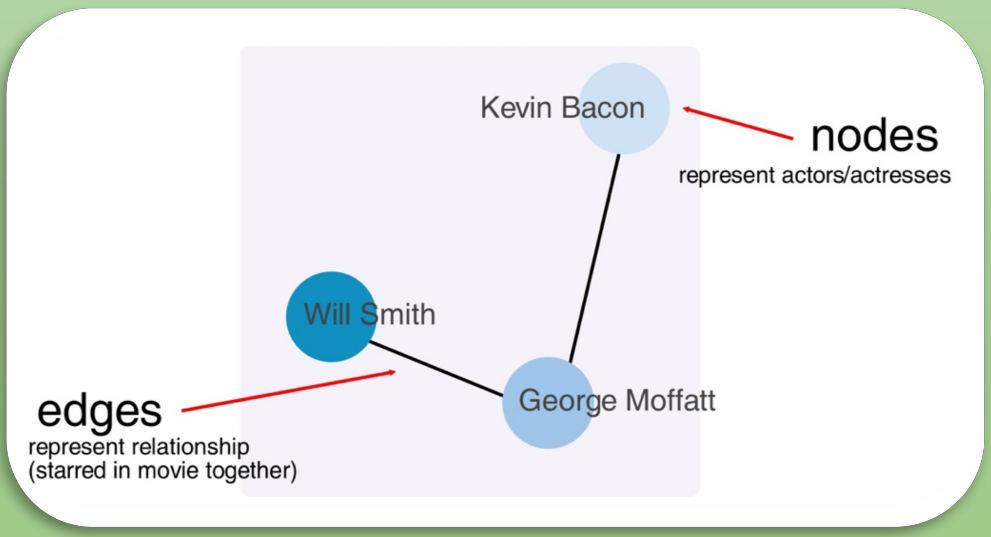
The **Kevin Bacon game** came out of the idea of the **six degrees of separation,** and is sometimes called the **Six Degrees of Kevin Bacon**. Given any actress or actor, the goal is to link that person to Kevin Bacon in the least amount of steps possible. The number of steps is called someone's **Bacon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



Let's get started!

Activities Overview

You will create a small world network from the Kevin Bacon game.

You'll **review and use** these concepts:

1. Breeds
2. Links
3. Labels
4. Degree
5. The network extension/network formatting

You'll learn these **new** concepts:

1. Influence of a node (Eigenvector centrality)
2. **foreach** (loops and nested loops)

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.

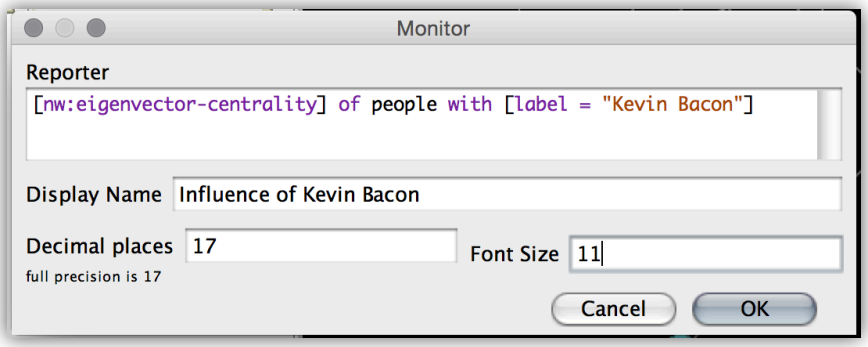
Why do we care about influence?

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!)

In NetLogo, we can calculate the influence of a node by using the **networks extension**. Be sure to include it at the top of each file as you did in the previous activities.

We can make a monitor to observe a node's influence like this:



Notice the `nw:` before `eigenvector-centrality`. The `nw:` tells NetLogo to look in the networks extension to find the `eigenvector-centrality` command.

Remember that we can use **breeds** to name our turtles whatever we want (here we named them **people**). Also, we can use **labels** to tell our people apart.

```
let num [1 2 3 4]
foreach num show
```

foreach

In the last lesson, we had to make each edge by hand. By using **foreach**, we can simply make a list of nodes we want to make or add properties to, and then specify a generic behavior for each of them. (If you're not familiar with lists in NetLogo, go here.)

Let's start with a simple example. Let's make a list of numbers and use **foreach** to show them:

How about something a little more interesting? Let's use the same list, but create turtles instead:

```
let num [1 2 3 4]
foreach num [create-turtles ?]
```

WHAT'S THAT QUESTION MARK ABOUT?

- NetLogo processes a list in order when a **foreach** loop uses it. **The ? is the element in the list that NetLogo is currently using.**
- In the example above, NetLogo will create 1 turtle. Then, it will go to the next element in the loop and create 2 turtles, then 3, then 4.
- If you're working with multiple lists, you can use ?1, ?2, etc.

TROUBLESHOOTING **foreach** blocks

- you may have to add parentheses, like this:

```
(foreach num move [create-turtles ?1] [ask turtles [forward ?2]])
```

(Here, `move` is another list.)

Activity 1: Kevin Bacon

Create a new NetLogo file. Name it *yourlastname_yourfirstname*_BaconGame.nlogo.

Refer to the box below for an example of people with Bacon number one and Bacon number two, or find your own using the Google tool mentioned earlier.

BACON NUMBER of 1
Michael Fassbender --X-Men First Class w/Kevin Bacon
Idris Elba -- X-Men First Class w/Kevin Bacon
Jennifer Lawrence -- X-Men First Class w/Kevin Bacon
Andy Garcia -- The Air I Breathe w/Kevin Bacon
John Lithgow -- Footloose w/Kevin Bacon

BACON NUMBER of 2
Penelope Cruz --The Counselor w/Michael Fassbender
Beyoncé -- Dreamgirls w/John Lithgow

1. Create a breed of **people**.
2. Create a `bacon-game` procedure and add a button for it on the **interface** tab. Your `bacon-game` procedure should clear-all and set the default shape of your **people** to "person". Then, your `bacon-game` procedure **must call the following procedures**:
   - setup-kevin-bacon
   - setup-bacon-number-one
   - setup-bacon-number-two
   - setup-intermediate-links
   - setup-size-degree
3. Create the procedures from 2). They don't need buttons. Use the table below.

setup-kevin-bacon	Create a person. Set its label to "Kevin Bacon." `to setup-kevin-bacon` `  create-people 1 [` `  set label "Kevin Bacon"` `  ]` `end`

setup-bacon-number-one	Make a list. The list should contain the labels (names) of everyone with a Bacon number of one. **foreach** of the names in your list, create a person, set their label to the list value, and link them to Kevin Bacon. `to setup-bacon-number-one` ` let labels ["Michael Fassbender" "Idris Elba" "Andy Garcia" "Jennifer Lawrence" "John Lithgow"]` ` (foreach labels [` `  create-people 1 [` `  set label ?` `  create-link-with one-of other people with [label = "Kevin Bacon"]` `  ]` `  ])` `end`
setup-bacon-number-two	Make 2 lists. The first list should contain the labels (names) of everyone with a Bacon number of two. The second list contains the corresponding person linked with the person in the first list. To be clear, the first person in the first list is linked with the first person in the second list. The second person in the first list is linked with the second person in the second list…etc. You already created the people in the second list in setup-bacon-number-one. **foreach** of the labels in list one, create a person and make a link with the corresponding person in list two. `to setup-bacon-number-two` `  let node["Penelope Cruz" "Beyonce"]` `  let nodeConnection["Michael Fassbender" "John Lithgow"]` `  (foreach node nodeConnection[` `  create-people 1[` `  set label ?1` `  create-link-with one-of other people with [label = ?2]` `  ]` `  ])` `end`
setup-intermediate-links	All of these people starred in the same movie. Make two identical lists of their labels (names). **foreach** of the people in list one, go through the second identical list (with a nested **foreach**) and make links with everyone (except themselves).

	`` to setup-intermediate-links let node ["Michael Fassbender" "Idris Elba" "Jennifer Lawrence"] let nodeConnection ["Michael Fassbender" "Idris Elba" "Jennifer Lawrence"] (foreach node [ask people with [label = ?] [(foreach nodeConnection[if ? != label [create-link-with one-of other people with [label = ?]]])]]) end ```
setup-size-degree	Set the peoples' size to their **degree**. (We can get the degree of a person by **count**ing how many **link-neighbors** there are.) ``` to setup-size-degree ask people [set size [count link-neighbors] of self] end ```

4. Use a **layout** to organize the nodes. Put this code at the end of your `bacon-game` procedure.
5. Create the following monitors. (Look at page 4 of this document for additional help with the Influence monitors):

Degree of Kevin Bacon [count link-neighbors] of people with [label = "Kevin Bacon"]
Average Degree of Network mean [count link-neighbors] of people
Degree of J. Lawrence
Influence of Kevin Bacon [nw:eigenvector-centrality] of people with [label = "Kevin Bacon"]
Average Influence mean [nw:eigenvector-centrality] of people
Influence of J. Lawrence
Influence of M. Fassbender
Influence of Beyonce

HINTS:

1. *Don't forget to include the **nw** extension.*
2. *Turn off world wrapping, or your edges will look crazy.*

Turning in Your Project

Pre-submission checklist

- ☐ Is your name, date, and project title at the top of your file in comments?
- ☐ Is your code organized and formatted to course standards?
- ☐ Does it work without errors?
- ☐ Is your file named correctly?

If you answered YES to all of these questions, congrats! You are ready to turn in your project. You will be given instructions for how to turn-in in class.

Rubric

	Networks 2: The Kevin Bacon Game (10 points)
Points	Task to be completed
2	**Clarity and professionalism:** Your file is named and commented appropriately. **[1 point]** The code is formatted and is readable. **[1 point]**
8	**Activity 1:** The Kevin Bacon Network Activity is complete and works as expected. **[8 points]**