Network functions and visualization

by Alexander Pico

2018-03-17

This vignette will show how to perform basic network operations on an iGraph networks and use this information to customize its appearance in Cytoscape directly from R using the RCy3 package

*From Vessy's "Fun with R blog": http://www.vesnam.com/Rblog/viznets5/

Installation

Required Software

The whole point of RCy3 is to connect with Cytoscape. You will need to install and launch Cytoscape:

- Download the latest Cytoscape from http://www.cytoscape.org/download.php
- Complete installation wizard
- Launch Cytoscape

Read a data set.

Data format: dataframe with 3 variables; variables 1 & 2 correspond to interactions; variable 3 is weight of interaction

```
lesmis <- system.file("extdata","lesmis.txt", package="RCy3")
dataSet <- read.table(lesmis, header = FALSE, sep = "\t")
```

Create a graph. Use simplyfy to ensure that there are no duplicated edges or self loops

```
gD <- igraph::simplify(igraph::graph.data.frame(dataSet, directed=FALSE))
```

Verify the number of nodes (77) and edges (254):

```
igraph::vcount(gD)
igraph::ecount(gD)
```

Common iGraph functions

Calculate some node properties and node similarities that will be used to illustrate different plotting abilities

Calculate degree for all nodes

```
degAll <- igraph::degree(gD, v = igraph::V(gD), mode = "all")
```

Calculate betweenness for all nodes

```
betAll <- igraph::betweenness(gD, v = igraph::V(gD), directed = FALSE) /
(((igraph::vcount(gD) - 1) * (igraph::vcount(gD)-2)) / 2)
betAll.norm <- (betAll - min(betAll))/(max(betAll) - min(betAll))
rm(betAll)
```

Calculate Dice similarities between all pairs of nodes

```
dsAll <- igraph::similarity.dice(gD, vids = igraph::V(gD), mode = "all")
```

Add attributes to network

Add new node attributes based on the calculated node properties/similarities

```
gD <- igraph::set.vertex.attribute(gD, "degree", index = igraph::V(gD), value =
degAll)
gD <- igraph::set.vertex.attribute(gD, "betweenness", index = igraph::V(gD), value =
betAll.norm)
```

Check the attributes. You should see “degree” and “betweeness” now, in addition to “name”.

```
summary(gD)
```

And now for the edge attributes…

```
F1 <- function(x) {data.frame(V4 = dsAll[which(igraph::V(gD)$name ==
as.character(x$V1)), which(igraph::V(gD)$name == as.character(x$V2))])}
dataSet.ext <- plyr::ddply(dataSet, .variables=c("V1", "V2", "V3"), function(x)
data.frame(F1(x)))

gD <- igraph::set.edge.attribute(gD, "weight", index = igraph::E(gD), value = 0)
gD <- igraph::set.edge.attribute(gD, "similarity", index = igraph::E(gD), value = 0)
```

Note: The order of interactions in dataSet.ext is not the same as it is in dataSet or as it is in the edge list and for that reason these values cannot be assigned directly

```
for (i in 1:nrow(dataSet.ext))
{
    igraph::E(gD)[as.character(dataSet.ext$V1) %--%
as.character(dataSet.ext$V2)]$weight <- as.numeric(dataSet.ext$V3)
```

```
    igraph::E(gD)[as.character(dataSet.ext$V1) %--%
as.character(dataSet.ext$V2)]$similarity <- as.numeric(dataSet.ext$V4)
}
rm(dataSet,dsAll, i, F1)
```

Check the edge attributes. You should see “weight” and “similarity” added.

```
summary(gD)
```

Let check it out in Cytoscape

Update: You can go straight from igraph to Cytoscape, sending all attributes and displaying graph!

```
createNetworkFromIgraph(gD,new.title='Les Miserables')
```

Let’s decide on a layout

A list of available layouts can be accessed from R as follows:

```
getLayoutNames()
```

We’ll select the “fruchterman-rheingold” layout. To see properties for the given layout, use:

```
getLayoutPropertyNames("fruchterman-rheingold")
```

We can choose any property we want and provide them as a space-delimited string:

```
layoutNetwork('fruchterman-rheingold gravity_multiplier=1 nIterations=10')
```

But that is a crazy layout, so let’s try “force-directed” instead:

```
layoutNetwork('force-directed defaultSpringLength=70
defaultSpringCoefficient=0.000003')
```

Next, we can visualize our data

On nodes…

```
setNodeColorMapping('degree', c(min(degAll), mean(degAll), max(degAll)), c('#F5EDDD',
'#F59777', '#F55333'))
lockNodeDimensions(TRUE)
setNodeSizeMapping('betweenness', c(min(betAll.norm), mean(betAll.norm),
max(betAll.norm)), c(30, 60, 100))
```

…and edges

```
setEdgeLineWidthMapping('weight', c(min(as.numeric(dataSet.ext$V3)),
mean(as.numeric(dataSet.ext$V3)), max(as.numeric(dataSet.ext$V3))), c(1,3,5))
setEdgeColorMapping('weight', c(min(as.numeric(dataSet.ext$V3)),
mean(as.numeric(dataSet.ext$V3)), max(as.numeric(dataSet.ext$V3))), c('#BBEE00',
'#77AA00', '#558800'))
```

We will define our own default color/size schema after we defined node and edge rules, due to possible issues when using rules

```
setBackgroundColorDefault('#D3D3D3')
setNodeBorderColorDefault('#000000')
setNodeBorderWidthDefault(3)
setNodeShapeDefault('ellipse')
setNodeFontSizeDefault(20)
setNodeLabelColorDefault('#000000')
```

Viola! All done.

Track versions for your records

```
cytoscapeVersionInfo()
```

```
sessionInfo()
```