ESSC Tutorial

James D. Wilson

This file demonstrates how to implement the ESSC algorithm to extract statistically significant communities in undirected networks. If you have any questions, please contact James D. Wilson at jdwilson4@usfca.edu. For more information, please see the following publication:

Wilson, James D., Wang, Simi, Mucha, Peter J., Bhamidi, Shankar, and Nobel, Andrew B.

"A testing based extraction algorithm for identifying signifcant communities in networks."

The Annals of Applied Statistics 8(3) 1853 - 1891

Sourcing the code in R

The ESSC algorithm is currently available in R and is stored in one of my Github repositories. To source the code, type the following.

```
download.file("https://github.com/jdwilson4/ESSC/archive/master.zip",
              destfile = "master.zip")

# unzip the file
unzip("master.zip")

#Source the files
source(file = "ESSC-master/ESSC.R")
```

Arguments for ESSC

The ESSC function has a minimal number of arguments, described below

- *Adj.Matrix*: the (symmetric and binary) adjacency matrix that represents the undirected network from which you wish to extract communities. This matrix can be coded as a Sparse Matrix for memory.
- *alpha*: the false discovery rate associated with extracted communities. Default is 0.05.
- *Null*: the null hypothesis against which the connection strength of each node to a candidate community is compared. This can either be *"Binomial"* or *"Poisson"*.
- *Num.Samples*: integer value indicating the number of seed sets you would like to search from. Each seed set used will be the neighborhood from a randomly chosen (but unique) node. Default is set to the number of nodes in the graph so that all node neighborhoods are explored.
- *seed*: seed selected for reproducibility. Default is 1.

Output for ESSC

A list with two components - *Communities*: a list of significant communities where each component of the list gives the set of nodes contained in a community.

- *Background*: a numeric whose values indicate which nodes were not considered to be contained in a significant community decided by the FDR cutoff.

Example: Facebook Data

Here, we use ESSC to identify significant communities in the Personal Facebook Data described in the original paper.

Loading needed packages

```r
install.packages("httr", repos='http://cran.us.r-project.org')
```

```
## Installing package into '/Users/jdwilson4/Library/R/3.2/library'
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/hm/8gnvskgx0rb1c11fzmz7sgf82j1yqg/T//RtmpNiFYb8/downloaded_packages
```

```r
install.packages("igraph", repos='http://cran.us.r-project.org')
```

```
## Installing package into '/Users/jdwilson4/Library/R/3.2/library'
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/hm/8gnvskgx0rb1c11fzmz7sgf82j1yqg/T//RtmpNiFYb8/downloaded_packages
```

```r
install.packages("Matrix", repos='http://cran.us.r-project.org')
```

```
## Installing package into '/Users/jdwilson4/Library/R/3.2/library'
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/hm/8gnvskgx0rb1c11fzmz7sgf82j1yqg/T//RtmpNiFYb8/downloaded_packages
```

```r
require(Matrix, quietly = TRUE)
require(igraph, quietly = TRUE)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```r
require(httr, quietly = TRUE)
```

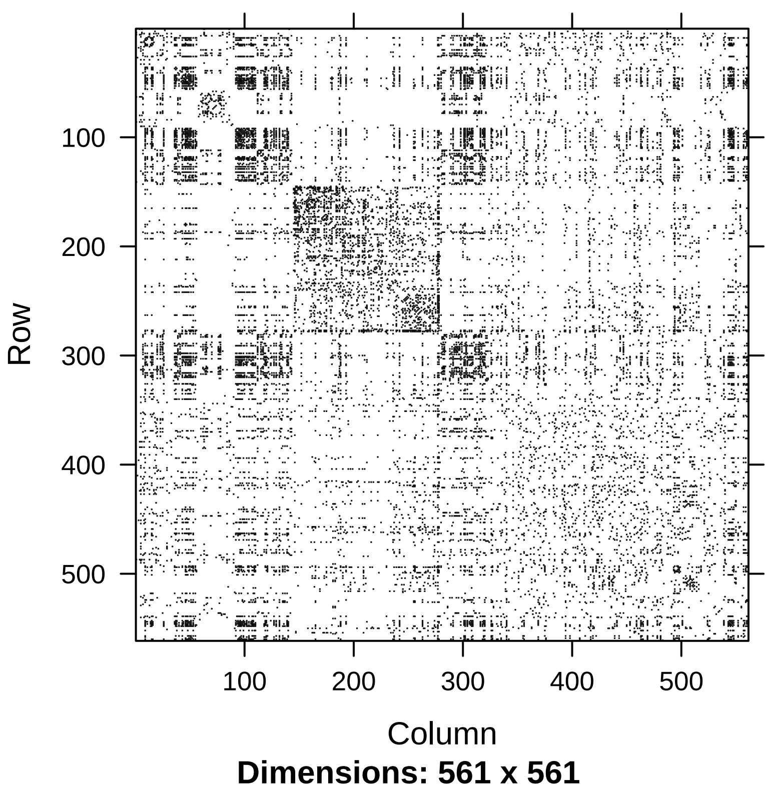
Loading the data

The Facebook data is contained in the unzipped folder, which we load from Github below.

```r
#Work around function to download data from Github
source_GitHubData <-function(url, sep = ",", header = TRUE)
{

  request <- GET(url)
  stop_for_status(request)
  handle <- textConnection(content(request, as = 'text'))
  on.exit(close(handle))
  read.table(handle, sep = sep, header = header)
}

#Load the Personal Facebook Data
Facebook.edgelist <- source_GitHubData(url =
"https://raw.githubusercontent.com/jdwilson4/ESSC/master/Personal_FB_Edgelist.csv")

#Convert this into an adjacency matrix
graph.1 <- graph.edgelist(as.matrix(Facebook.edgelist[,1:2]),directed = FALSE)
Facebook.adjacency <- get.adjacency(graph.1)
```

Heatmap of the Facebook Data

```r
image(Matrix(Facebook.adjacency))
```



Extract Communities with $\alpha = 0.01$

```r
Results.Facebook <- ESSC(Facebook.adjacency, 0.01, Null = "Poisson")

#Look at the number of communities, the average size, and the number of background

num.communities <- length(Results.Facebook$Communities)
num.background <- length(Results.Facebook$Background)
size.communities <- rep(0, num.communities)
for(i in 1:num.communities){
  size.communities[i] <- length(Results.Facebook$Communities[[i]])
}
```

Summary of the Identified Communities

```r
cat("Number of Communities =", num.communities)
```

```
## Number of Communities = 20
```

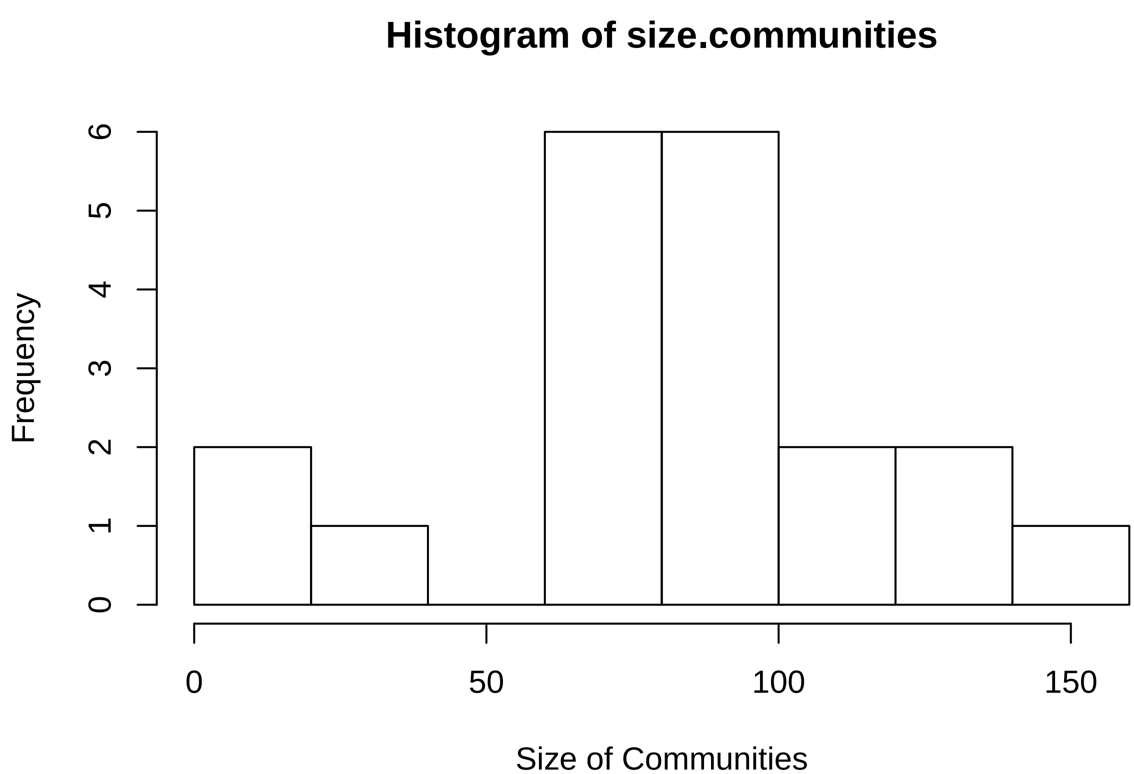
```r
cat("Number of Background Vertices =", num.background)
```

```
## Number of Background Vertices = 191
```

```r
cat("Community Sizes")
```

```
## Community Sizes
```

```r
hist(size.communities, n = 10, xlab = "Size of Communities")
```

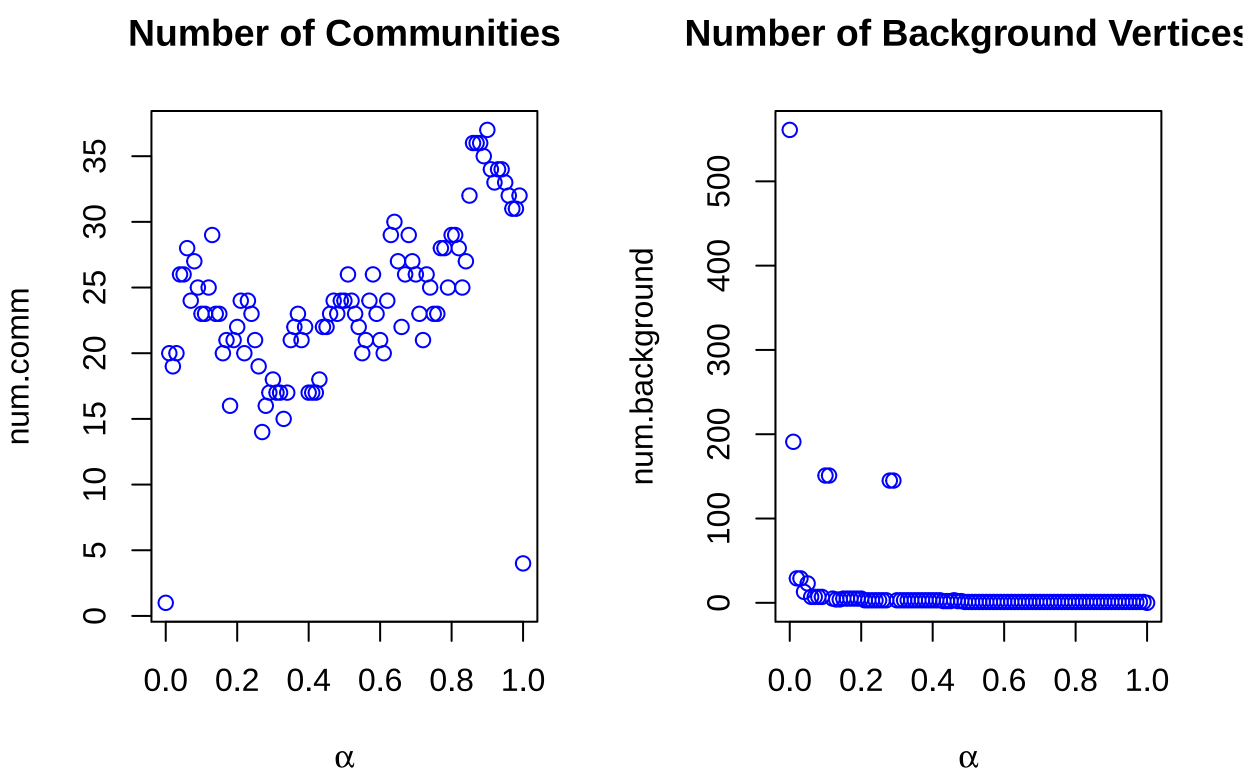


We see that there are 191 background vertices in the Facebook network when we consider an FDR of 0.01! That's nearly 34% of the vertices.

Effects of the FDR α

We evaluate the number of communities and the number of background vertices identified across a grid of FDR values α between 0 and 1 in increments of 0.01. We then plot the number of communities and the number of vertices in the background for each run.

```
alpha.grid <- seq(0, 1, 0.01)
num.comm <- rep(0, length(alpha.grid))
num.background <- rep(0, length(alpha.grid))

for(i in 1 : length(alpha.grid)){
  res <- ESSC(Facebook.adjacency, alpha.grid[i], Null = "Poisson")
  num.comm[i] <- length(res$Communities)
  num.background[i] <- length(res$Background)
}

#Plot the results for a comparison
par(mfrow = c(1, 2))
plot(y = num.comm, x = alpha.grid, type = "p", col = "blue", xlab =
       expression(alpha), main = "Number of Communities")

plot(y = num.background, x = alpha.grid, type = "p", col = "blue", xlab =
       expression(alpha), main = "Number of Background Vertices")
```



We see from above that as we increase α two trends occur:

1) the number of communities tends to increase

2) the number of background vertices tends to decrease

This occurs because small values of α act as an upper bound to the false discovery rate for each community. Thus, we expect that smaller values of α will lead to fewer vertices in each community. As a consequence, more vertices will be treated as a loosely connected background vertex.