Statistical Inference for Networks

Second Practical: Exponential random graph models

1 Which packages we need and which packages we do not need

For this practical you **must not** load the *igraph* package. We shall need the *ergm* package. You will also need to install the *sna* package.

Remember that this practical is also available at

http://www.stats.ox.ac.uk/~reinert/dtc/networks.html

and you can copy and paste from it.

Recall how it works: First we load the packages which we shall need. From the drop-down menu packages, select “load package”, then select *ergm*. Also from the drop-down menu packages, select “install package”. You will then be asked to connect to a UK mirror site for the R-project, and you can download the package from there. Remember that once you have installed a package, you will still need to load it.

2 Logistic models

First let’s look at our Florentine marriage data,

```
data(flo)
```

We convert this into an undirected network,

```
nflo<-network(flo, directed= FALSE)
```

Have a look at what you did:

```
nflo
```

Now we fit an exponential random graph model to the data, based on the number of 1-stars (i.e. edges), the number of 2-stars, and the number of triangles. For the ergm command, the "formula" format in R is required; which is the form

" network object ~ <model term 1> + < model term 2> + ...

So let’s try

```
gflo<- ergm(nflo ~ kstar(1:2) + triangle)
```

Type

summary(gflo)

to see the result.

To assess the goodness of fit,

gof(ergm(nflo ~ kstar(1:2) + triangle))

calculates p-values for geodesic distance (shortest path length), degree, and reachability summaries.

To diagnose the goodness-of-fit of ergms let's have a look at the plots

```
gofflo<-gof(gflo)
par(mfrow=c(1,3))
plot(gofflo)
plot(gofflo, plotlogodds=TRUE)
plot(gof(nflo ~  kstar(1:2) + triangle, theta0=c(-0.7891, -0.0288, 0.2461)))
```

It seems that the model fits reasonably well.

For a more interesting data set, the Faux Mesa High data represents a simulation of an in-school friendship network. The network is named faux.mesa.high because the school commnunity on which it is based is in the rural western US, with a student body that is largely Hispanic and Native American.

```
data(faux.mesa.high)
fmh<-faux.mesa.high
fmh
summary(fmh)
table(component.dist(fmh)$csize)
```

Try to fit a model based on nodematch for Grade, and on gender only.

```
modelfmh<- ergm(faux.mesa.high ~ nodematch("Grade", diff=T) +
nodefactor("Sex"))
```

summary(modelfmh)

Now try yourself to incorporate the number of edges in the model (just denoted by "edges" in ermg).

3 Adding and removing nodes, Monte Carlo tests

The command

```
sim1<-simulate(modelfmh)
```

would simulate a random network according to the model derived for fmh.

To check whether or not edge is present in the network, use “is.adjacent”, for example

```
is.adjacent(fmh, 1, 2)
is.adjacent(fmh, 1, 25)
```

To remove an edge, we just set that edge indicator to zero,

```
fmhminus<-fmh
fmhminus[1,25]<-0
```

Check:

```
is.adjacent(fmhminus, 1, 25)
```

To add an edge

```
fmhplus <- add.edge(fmh, 1, 2)
```

Check:

```
is.adjacent(fmhplus, 1, 2)
```

For a Monte Carlo test whether the edge (1,25) is likely to be in the network, fit the network model with this edge removed.

```
modelfmhminus<- ergm(fmhminus ~  nodematch("Grade", diff=T) +
nodefactor("Sex"))
```

Now you can count how often edge (1,25) is included in simulated graphs from this network model:

```
fsim<-0
for(i in 1:100){
sim1<-simulate(modelfmhminus)
 fsim<- fsim + sim1[1,25]}
fsim
```

My simulations give 3 out of 100, not a large probability.

You can repeat the procedure with the model which includes the number of edges as covariate.

When trying to repeat this analysis for the yeast data, you may have to wait a very long time for the results!