Homework 6

Jiaying Liu

November 23, 2018

Rerun the code

chapter2:Model specification and fitting

2.1 A simple motivating example

```
data('sampson')
network.vertex.names(samplike)

##  [1] "John Bosco"  "Gregory"     "Basil"       "Peter"       "Bonave
nture"
##  [6] "Berthold"    "Mark"        "Victor"      "Ambrose"     "Romaul
d"
## [11] "Louis"       "Winfrid"     "Amand"       "Hugh"        "Bonifa
ce"
## [16] "Albert"      "Elias"       "Simplicius"
```

From this result we can see the vertex name of the sampson dataset.

```
samplike %v% "group"

##  [1] "Turks"    "Turks"    "Outcasts" "Loyal"    "Loyal"    "Loyal"

##  [7] "Turks"    "Loyal"    "Loyal"    "Loyal"    "Loyal"    "Turks"

## [13] "Outcasts" "Turks"    "Turks"    "Turks"    "Outcasts" "Outcast
s"
```

And notices that in this dataset, the observations are sperated into different group, so in the next step, we are going to fit the model with the group.

We use the maximum likelihood method to get the estimator for the latent position model. Use the defaults value of the algorithm we get the result as following.

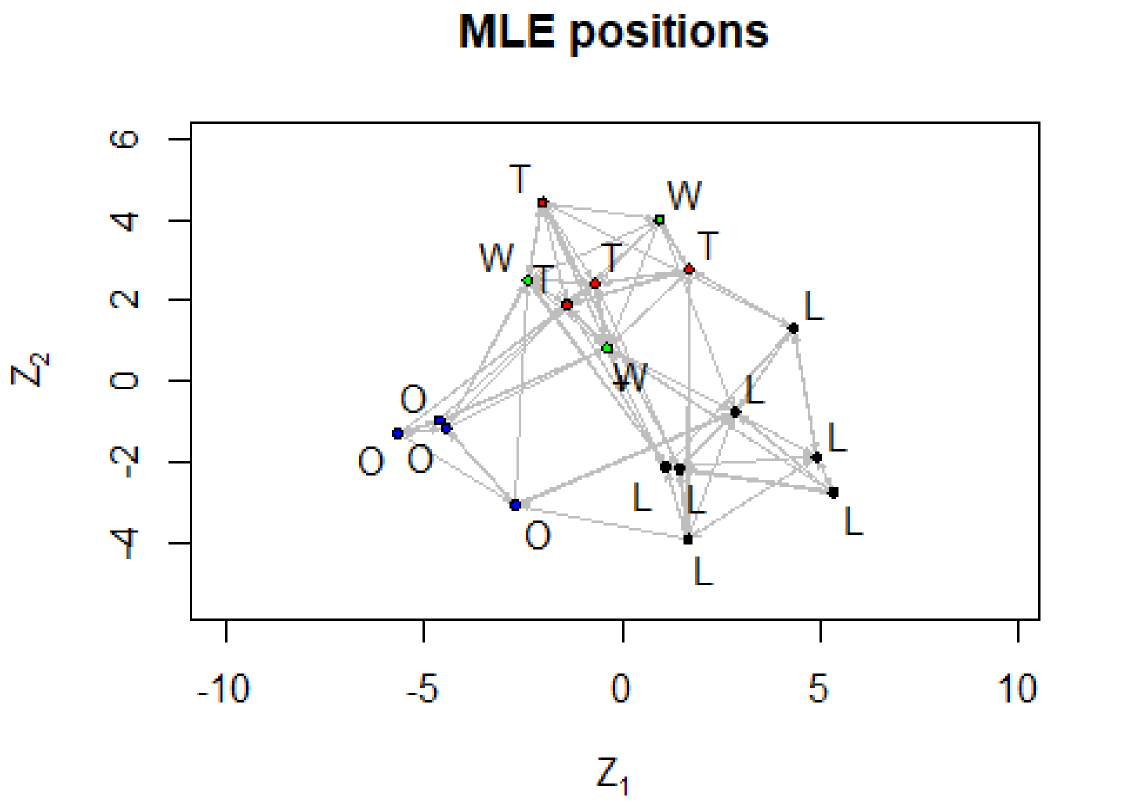
```
samplike.fit <- ergmm(samplike ~ euclidean(d = 2), tofit = c("mle"))
samplike.fit$mle$Z
```

maximum likelihood position		
obs	[,1]	[,2]
[1,]	-0.376	0.818642
[2,]	-1.3932	1.895042
[3,]	-4.60591	-0.96649
[4,]	4.90604	-1.8821
[5,]	2.852213	-0.76819
[6,]	5.321734	-2.76941
[7,]	-2.38686	2.502459
[8,]	1.099442	-2.13453
[9,]	1.453505	-2.16958
[10,]	1.658881	-3.92531
[11,]	4.315043	1.310109
[12,]	-0.68063	2.417095
[13,]	-2.68898	-3.07092
[14,]	1.691511	2.770308
[15,]	0.940941	4.003357
[16,]	-2.00835	4.428827
[17,]	-5.66076	-1.29648
[18,]	-4.43864	-1.16283

From above table, we can get the result of maximum likelihood positions.

To make it more directly to detect the position, we are going to plot the result. And before drawing the picture, we need to change the group into a single sign which make it more easy to plot.

```
oneL <- samplike %v% "group"
oneL[oneL == "Turks"] <- "T"
oneL[oneL == "Outcasts"] <- "O"
oneL[oneL == "Loyal"] <- "L"
oneL[c(1, 7, 15)] <- "W"
oneL

##  [1] "W" "T" "O" "L" "L" "L" "W" "L" "L" "L" "L" "T" "O" "T" "W" "T"
 "O"
## [18] "O"

oneLcolors <- c("red", "blue", "black", "green")[match(oneL,c("T", "O",
 "L", "W"))]
plot(samplike.fit, label = oneL, vertex.col = oneLcolors,what = "mle",
main = "MLE positions", print.formula = FALSE,labels = TRUE)
title(sub ="Color represents the estimated groups; Labels the Sampson's
 groups")
```



In the above figure, we can see the position of the vertex and the group distribution with different color.

2.3 An example of a latent position fit with clustering

In this section, we fit the latent position model with two dimensions and three group to the same dataset.

```
set.seed(3141)
samplike.fit <- ergmm(samplike ~ euclidean(d = 2, G = 3),verbose = TRUE
)

## Generating initial values for MCMC:
## Computing geodesic distances... Finished.
## Computing MDS locations... Finished.
## Computing other initial values... Finished.
## Finding the conditional posterior mode... Finished.
## Burning in... Finished.
## Starting sampling run... Finished.
## Post-processing the MCMC output:
## Performing label-switching... Finished.
## Fitting the MKL locations... Finished.
## Fitting MBC conditional on MKL locations... Finished.
## Performing Procrustes transformation... Finished.

summary(samplike.fit)

## NOTE: It is not certain whether it is appropriate to use latentnet's
 BIC to select latent space dimension, whether or not to include actor-
specific random effects, and to compare clustered models with the unclu
stered model.
```

Summary of model fit
Formula: samplike ~ euclidean(d = 2, G = 3)
Attribute: edges
Model: Bernoulli
MCMC sample of size 4000, draws are 10 iterations apart, after burnin of 10000 iterations.

Covariate coefficients posterior means				
	Estimate	2.50%	97.50%	pvlue
Intercept	1.9982	1.3297	2.7504	< 2.2e-16

Overall BIC	336.8183
Likelihood BIC	254.0368
#Latent space/clustering BIC	82.78152

Covariate coefficients MKL:	
Intercept	1.16573

When we set the dimention to be 2 and have 3 group, we have the result as above table. We can see after 10 iteration, we have the estimator intercept=1.9982. And its p-value smaller than 0.05, so we can say this estimator is significant. The covariate coefficients of MKL is 1.16573.

2.6 Nonbinary edge weights

In this section, we are going to add one more item in the ergmm procedure name family, fam.par. With those parameters, we are able to specify the model for the response variable.With the dataset “tribes”, we have 16 tribes in the Eastern Central Highlands of New Guinea, with each pair of tribes report to be in one of three states of relation: alliance, antagonism, or neutrality.we code alliance as 2, antagonism as 0, and neutrality as 1, and fit a binomial model with 2 trials.

```
data("tribes")
tribes.fit <- ergmm(tribes ~ euclidean(d = 2, G = 3),
 family = "binomial", fam.par = list(trials = 2),
 response = "sign.012", verbose = 1)

## Generating initial values for MCMC:
## Computing geodesic distances... Finished.
## Computing MDS locations... Finished.
## Computing other initial values... Finished.
## Finding the conditional posterior mode... Finished.
## Burning in... Finished.
## Starting sampling run... Finished.
## Post-processing the MCMC output:
## Performing label-switching... Finished.
## Fitting the MKL locations... Finished.
## Fitting MBC conditional on MKL locations... Finished.
## Performing Procrustes transformation... Finished.

summary(tribes.fit)
```

Summary of model fit	
Formula:	tribes ~ euclidean(d = 2, G = 3)
Attribute:	sign.012
Model:	binomial
MCMC sample of size 4000, draws are 10 iterations apart, after burnin of 10000 iterations.	

Covariate coefficients posterior means				
	Estimate	2.50%	97.50%	pvlue
Intercept	2.7137	1.8876	3.5452	< 2.2e-16

Overall BIC	224.398
Likelihood BIC	172.4572
#Latent space/clustering BIC	51.94078

Covariate coefficients MKL:	
Intercept	1.979373

From the above table, we get the result about the parameter. The intercept estimate is 2.701 and its p value is smaller that 0.05. So we can say this is significant. And the covariate coefficients of MKL is 1.95.

Output format and visualization

3.2 summarizing model fit

```
attr(samplike.fit$sample, "Q")
```

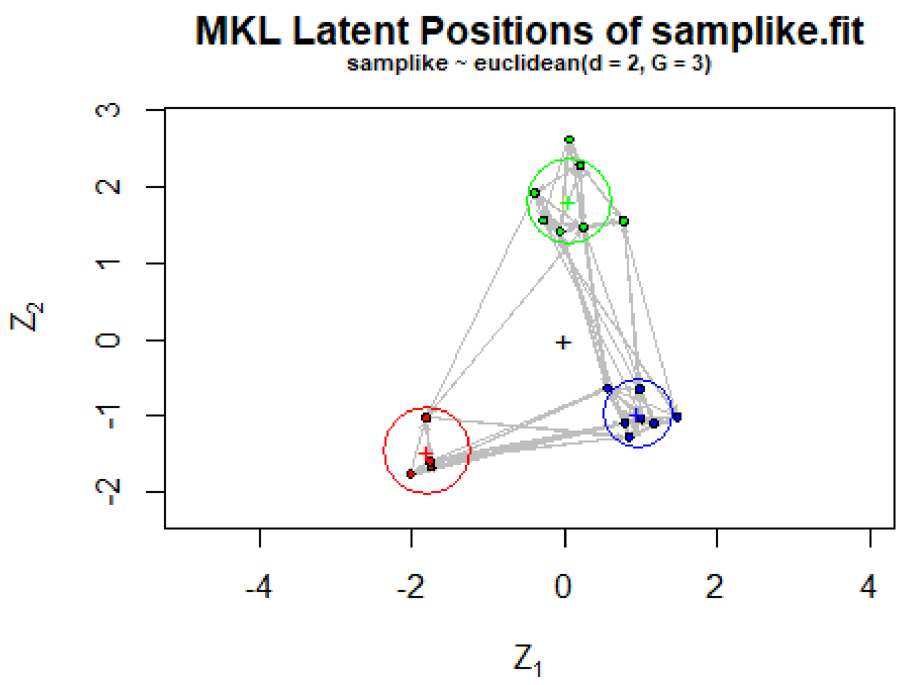
Posterior probabilites			
	[,1]	[,2]	[,3]
[1,]	0.053835	0.092938	0.853227
[2,]	0.035254	0.024076	0.94067
[3,]	0.953183	0.012768	0.034049
[4,]	0.011233	0.981057	0.00771
[5,]	0.022984	0.950187	0.026829
[6,]	0.013166	0.977209	0.009625
[7,]	0.044978	0.024828	0.930195
[8,]	0.016493	0.945623	0.037884
[9,]	0.016343	0.954571	0.029085
[10,]	0.01908	0.964536	0.016385
[11,]	0.021472	0.918249	0.060279
[12,]	0.020321	0.025991	0.953688
[13,]	0.90628	0.055289	0.038431
[14,]	0.023461	0.067618	0.908922
[15,]	0.020884	0.028204	0.950913
[16,]	0.028882	0.057204	0.913914
[17,]	0.962082	0.011334	0.026584
[18,]	0.955627	0.012492	0.031881

This is the posterior probabilities of group membership for each monk in samplike.fit dataset. We can see that when the probabilities above 90%, the monks are more likely to been assigned to a group.

3.3 Plotting model fits

Next we are going to plot the model and set how is fit. The latent position model can provided us a principled way to visualize the network of interest in 2 or 3 dimensions.

```
plot(samplike.fit)
```

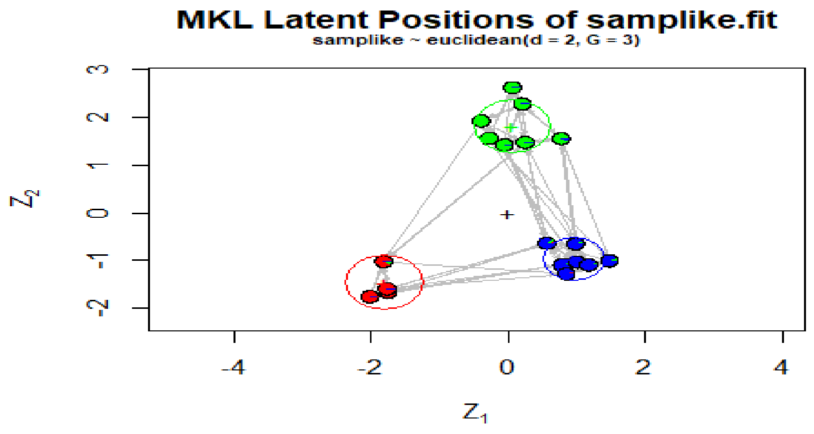


From the above plot we can see the MKL latent positions of samplike fit. Different group and different color

Also we can use the pie=TRUE statement to make the each node in the clustering as a small pie chart. With this pie chart, we are able to know that the proportions of MCMC draws in which that node belonged to each cluster.

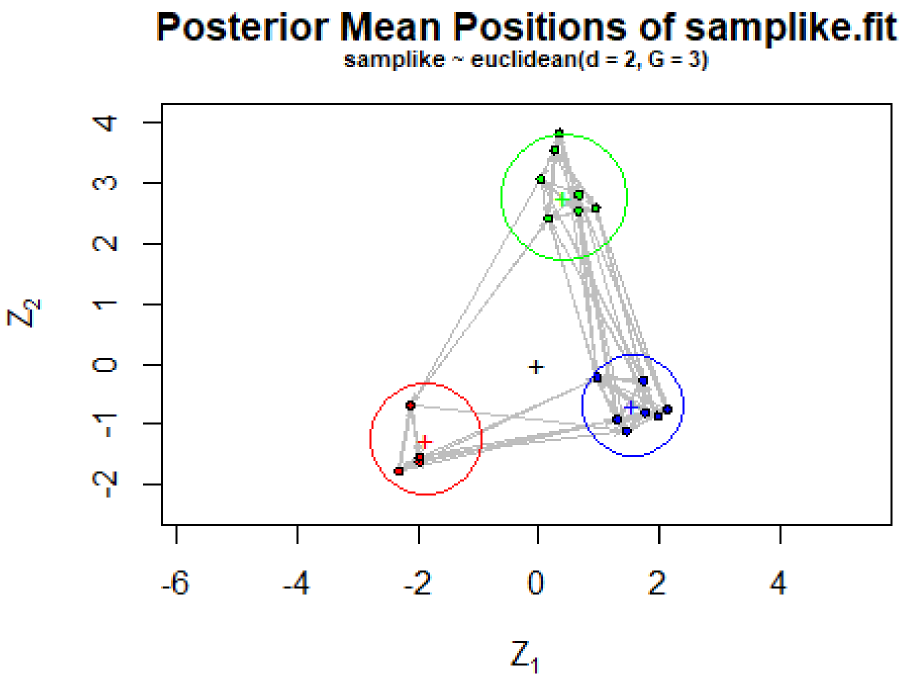
we can get the result of minimum Kullback-Lerbler(MKL) estimates for the sampson's Monks. After we setting the vertex.size, it is able to enlarge the size of vertex so that it is more easy to detect.

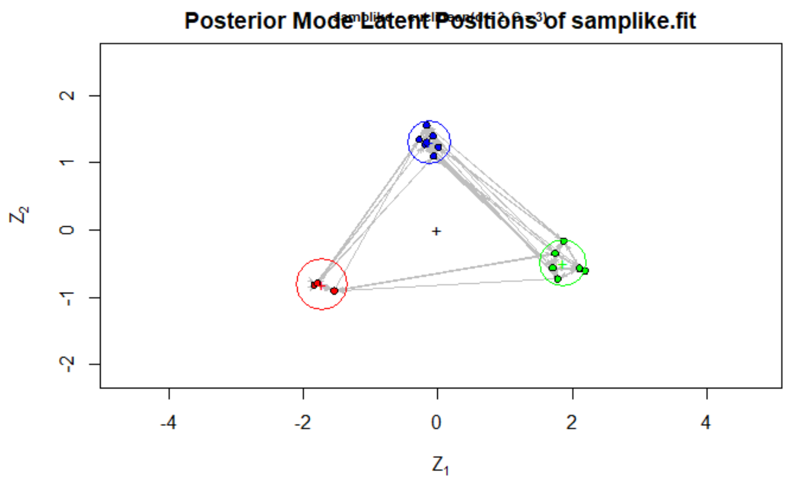
```
plot(samplike.fit, pie = TRUE, vertex.cex = 2.5)
```



We can also use other point estimates using the argument what='pmean' or what='pmode'. For example, in this dataset, we use the posterior means to estimate the model and get the result as above showing.

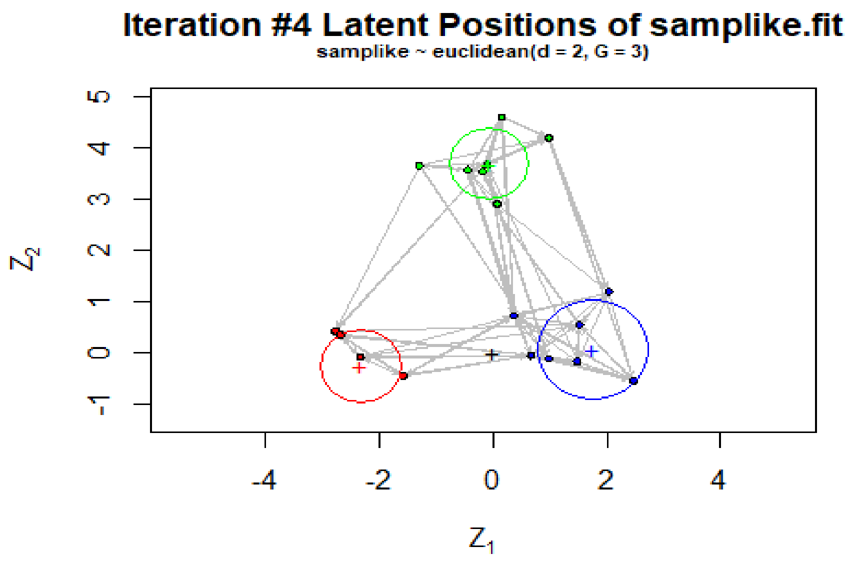
```r
plot(samplike.fit, what = "pmean")

samplike.fit <- ergmm(samplike ~ euclidean(d = 2, G = 3),verbose = TRUE
,tofit = c("pmode")

plot(samplike.fit, what = "pmode")
```





And there is another way to show the MCMC iteration. For example if we are interested in the position of 4th iteration, we can use the what statement to specify the iteration we want.

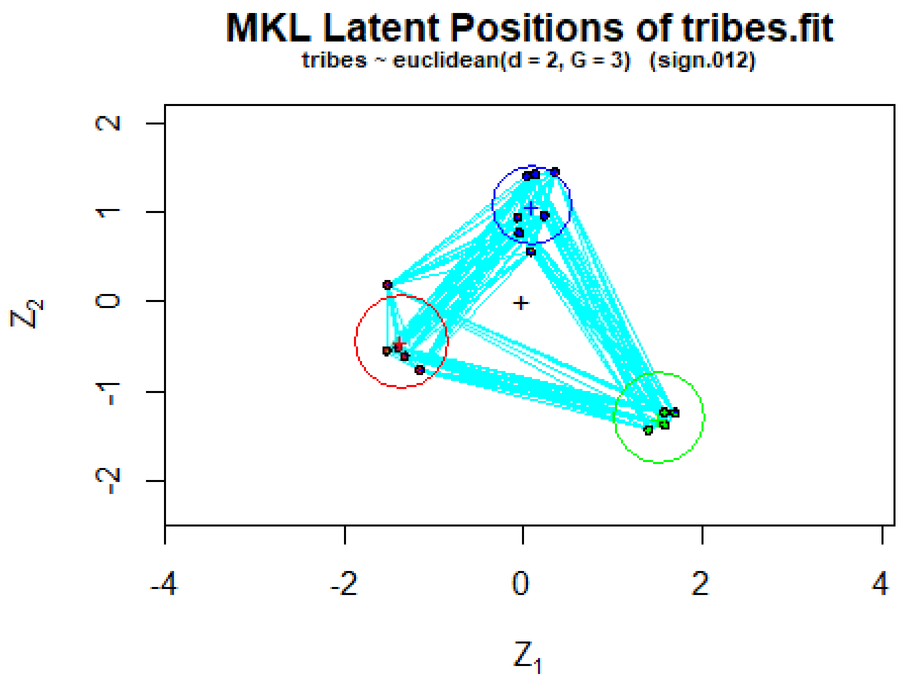
```r
plot(samplike.fit, what = 4)
```



```r
#for (i in 1:samplike.fit$control$sample.size) {
# plot(samplike.fit, what = i)
# Sys.sleep(0.1)
#}
```

This statement will create all the iteration plot and it will a lot. There are 4000 of figure draw from above statement. It is different to show but it will be same as above figure just with different time of iteration.

```r
plot(tribes.fit, edge.col = as.matrix(tribes, "gama",
 m = "a") * 3 + as.matrix(tribes, "rova", m = "a") *
 2, pie = TRUE)
```



Assessment of model fit via posterior predictive checks

In this section, we are going to run the assessment of the model fit. With is assessment, we can consider posterior predictive checks.The goodness of fit method based on how similar the networks simulated from the posterior predictive distribution are to the original for some higher-order statistics of interest, such as the distribution of geodesic distances.

```
samplike.fit.gof <- gof(samplike.fit)
summary(samplike.fit.gof)

plot(samplike.fit.gof)
```

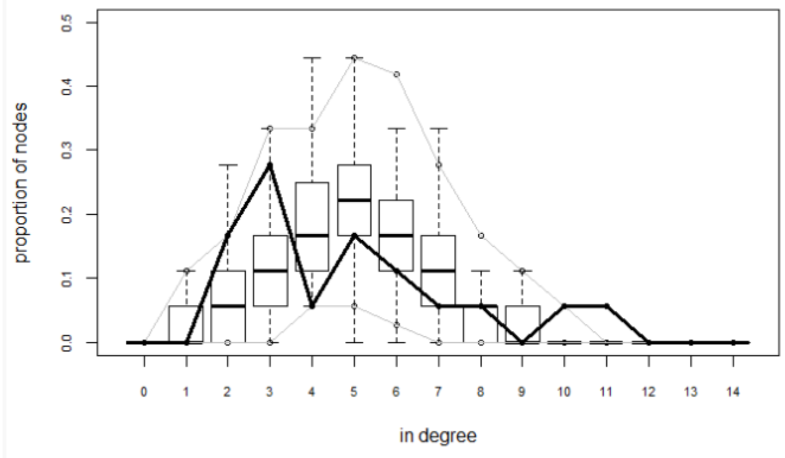
Goodness-of-fit for in-degree					
	obs	min	mean	max	p-value
0	0	0	0.07	1	1
1	0	0	0.54	2	1
2	3	0	1.34	5	0.2
3	5	0	2.62	7	0.24
4	1	0	3.4	8	0.26
5	3	1	3.62	8	0.96
6	2	0	3.03	7	0.78
7	1	0	2	7	0.76
8	1	0	0.94	5	1

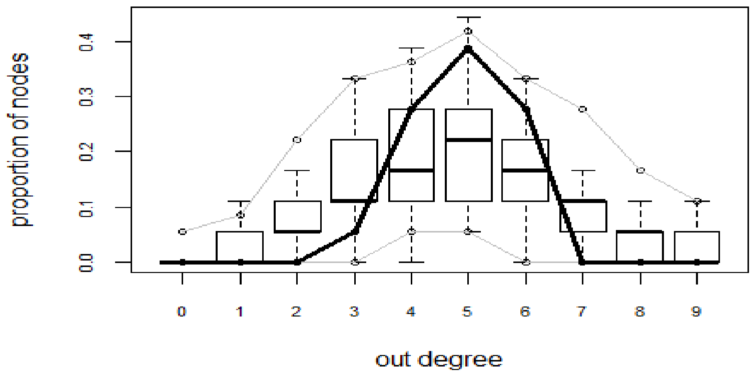
9	0	0	0.33	2	1
10	1	0	0.1	2	0.18
11	1	0	0.01	1	0.02

Goodness-of-fit for out-degree					
	obs	min	mean	max	p-value
0	0	0	0.06	2	1
1	0	0	0.36	2	1
2	0	0	1.37	5	0.48
3	1	0	2.79	7	0.32
4	5	0	3.41	7	0.56
5	7	1	3.96	8	0.12
6	5	0	2.79	6	0.24
7	0	0	1.83	6	0.3
8	0	0	0.93	4	0.76
9	0	0	0.35	3	1
10	0	0	0.14	1	1
11	0	0	0.01	1	1

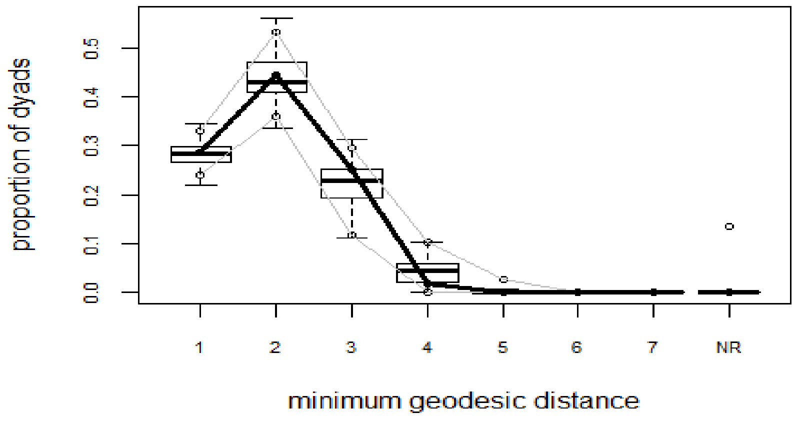
Goodness-of-fit for minimum geodesic distance					
	obs	min	mean	max	p-value
1	88	67	86.56	106	0.92
2	136	103	134.86	172	0.8
3	77	32	67.64	96	0.52
4	5	0	12.71	35	0.44
5	0	0	1.09	10	1
6	0	0	0.07	5	1
7	0	0	0.02	1	1
Inf	0	0	3.05	56	1

We have three kind of default statistics which are idegree, odegree and geodes ic distance.





Goodness-of-fit diagnostics



For the in-degree plot, the line is the acuter monks and the box is the produce monks. we can see that the overall model reproduces the observed statistics well. However, there are a tendency to under produce monks of degree 2 and

3. And there is a tendency to over produce monks of degree 4. In other word, there is a tendency for the actual monks to have degree 2 or 3 more often and 4 less often than the model expects.

For the out-degree plot, the line is the acuter monks and the box is the produce monks. we can see that the overall model reproduces the observed statistics well. However, there are a tendency to under produce monks of degree 4 5 and 6. And there is a tendency to over produce monks of degree 3 and 7. In other word, there is a tendency for the actual monks to have degree 4 5 or 7 more often and 3 or 7 less often than the model expects.

For the minimum geodesic distance, we can see the model is well describe the distance. the actual monk contains the fit monks.