# An Example Analysis Using LOLOG

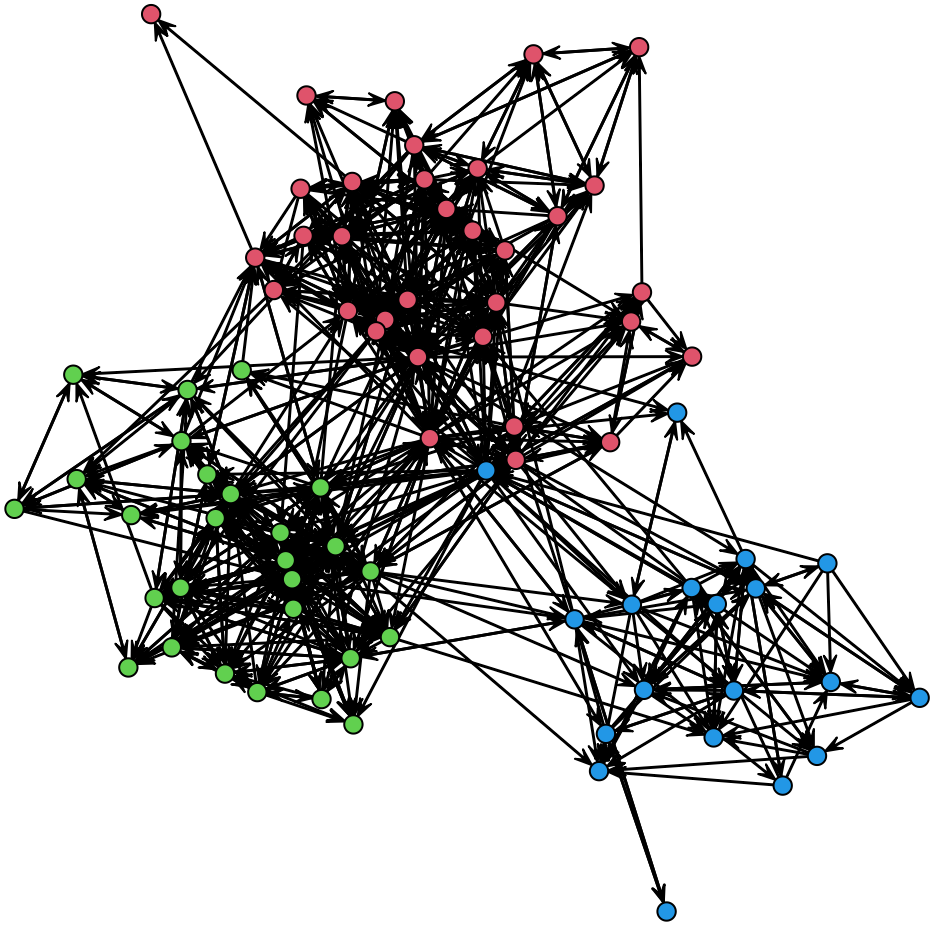
The Statnet Development Team

2024-01-03

## The `ukFaculty` dataset

The personal friendship network of a faculty of a UK university, consisting of 81 vertices (individuals) and 817 directed and weighted connections. The school affiliation of each individual is stored as a vertex attribute. Two individuals are missing the school attribute. We will remove these for analysis.

```
suppressPackageStartupMessages(library(network))
library(lolog)
#> Loading required package: Rcpp
data(ukFaculty)
#?ukFaculty
ukFaculty %v% "Group"  # The school affiliation of the faculty
#>  [1]  3  1  3  3  2  2  2  1  3  2  1  2  2  1  1  2  3  1  1  1  1  2  2  1  1
#> [26]  1  2  2  1  2  1  1  2  1  1  3  1  3  1  2  1  2  1  3  3  1  2  1  2 NA
#> [51]  1  1  3  1  1  1  1  1  3  3  3  3  2  1  2  2  2  2  2 NA  2  2  3  3  3
#> [76]  2  2  3  1  1  3
ukFaculty %v% "GroupC" # affiliation coded as categorical
#>  [1] "3" "1" "3" "3" "2" "2" "2" "1" "3" "2" "1" "2" "2" "1" "1" "2" "3" "1" "1"
#> [20] "1" "1" "2" "2" "1" "1" "1" "2" "2" "1" "2" "1" "1" "2" "1" "1" "3" "1" "3"
#> [39] "1" "2" "1" "2" "1" "3" "3" "1" "2" "1" "2" NA  "1" "1" "3" "1" "1" "1" "1"
#> [58] "1" "3" "3" "3" "3" "2" "1" "2" "2" "2" "2" "2" NA  "2" "2" "3" "3" "3" "2"
#> [77] "2" "3" "1" "1" "3"
delete.vertices(ukFaculty, which(is.na(ukFaculty %v% "Group")))
plot(ukFaculty, vertex.col = (ukFaculty %v% "Group" ) + 1)
```



```
ukFaculty
#>  Network attributes:
#>   vertices = 79
#>   directed = TRUE
#>   hyper = FALSE
#>   loops = FALSE
#>   multiple = FALSE
#>   bipartite = FALSE
#>   Type = TSPE
#>   Date = Mon Mar 19 21:56:02 2007
#>   Citation = Nepusz T., Petroczi A., Negyessy L., Bazso F.: Fuzzy communities and the concept of bri
#>   Author = Nepusz T., Petroczi A., Negyessy L., Bazso F.
#>   total edges= 781
#>     missing edges= 0
#>     non-missing edges= 781
#>
#>  Vertex attribute names:
#>     Group GroupC vertex.names
#>
#>  Edge attribute names:
#>     weight
```

We see a great number of like-to-like ties based on school affiliation, so this will probably be an important thing to model.

## A first attempt

Recall from the introductory vignette, a LOLOG represents the probability of a tie, given the network grown up to a time-point as

$$\text{logit}\big(p(y_{s_t} = 1|\eta, y^{t-1}, s_{\leq t})\big) = \theta \cdot c(y_{s_t} = 1|y^{t-1}, s_{\leq t})$$

where $s_{\leq t}$ is the growth order of the network up to time $t$, $y^{t-1}$ is the state of the graph at time $t-1$. $c(y_{s_t}|y^{t-1}, s_{\leq t})$ is a vector representing the change in graph statistics from time $t-1$ to $t$ if an edge is present, and $\theta$ is a vector of parameters.

If the graph statistics are dyad independent (i.e. the change in graph statistics caused by the addition or deletion of an edge depends only on the vertex covariates of the two vertices connected by the edge), then LOLOG reduces to a simple logistic regression of the presence of an edge on the change statistics.

It is usually a good idea to start off model building with a dyad independent model.

```
fitukInd <- lolog(ukFaculty ~ edges() + nodeMatch("GroupC") + nodeFactor("GroupC"))
#> Initializing Using Variational Fit
#>
#>  Model is dyad independent. Replications are redundant. Setting nReplicates <- 1L.
#> Model is dyad independent. Returning maximum likelihood estimate.
summary(fitukInd)
#>                      observed_statistics       theta         se pvalue
#> edges                                781 -3.72284710 0.13119098 0.0000
#> nodematch.GroupC                     663  2.71643436 0.10573322 0.0000
#> nodeFactor.GroupC.1                  733  0.09183592 0.06607506 0.1646
#> nodeFactor.GroupC.2                  584  0.20454753 0.06875534 0.0029
```

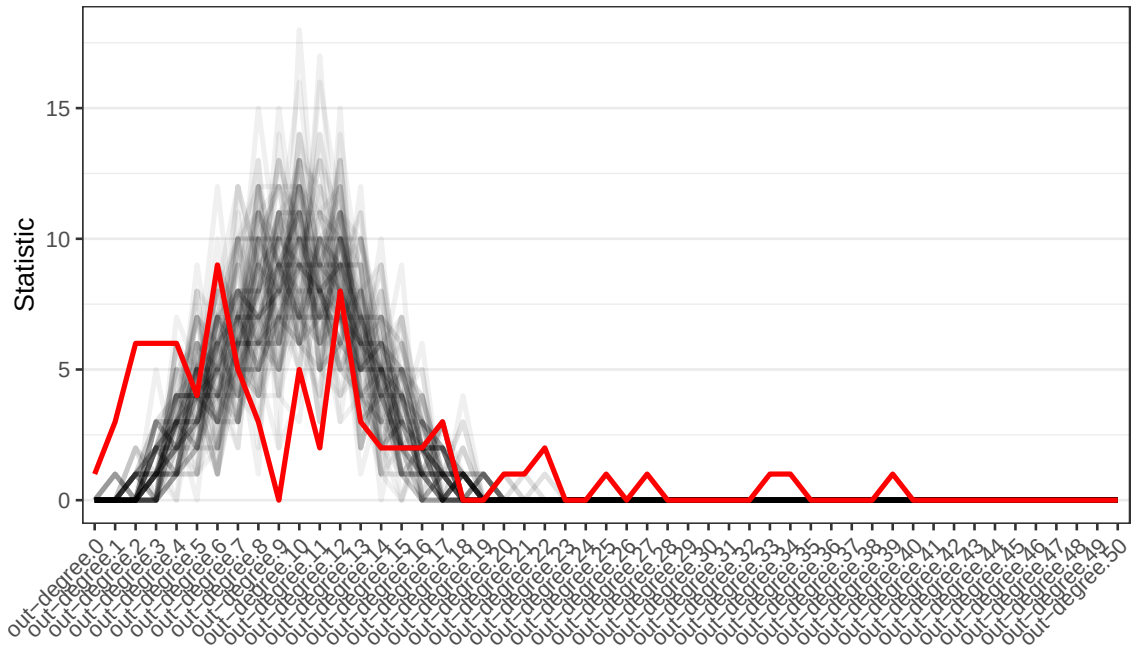
We see a highly significant term for ties within group `nodematch.GroupC`, and a difference in overall activity comparing group 2 to the baseline group 3. Group 1 is not significantly more active than 3.

For those familiar with ERGM modeling, dyad independent ergms are identical to dyad independent lolog models.

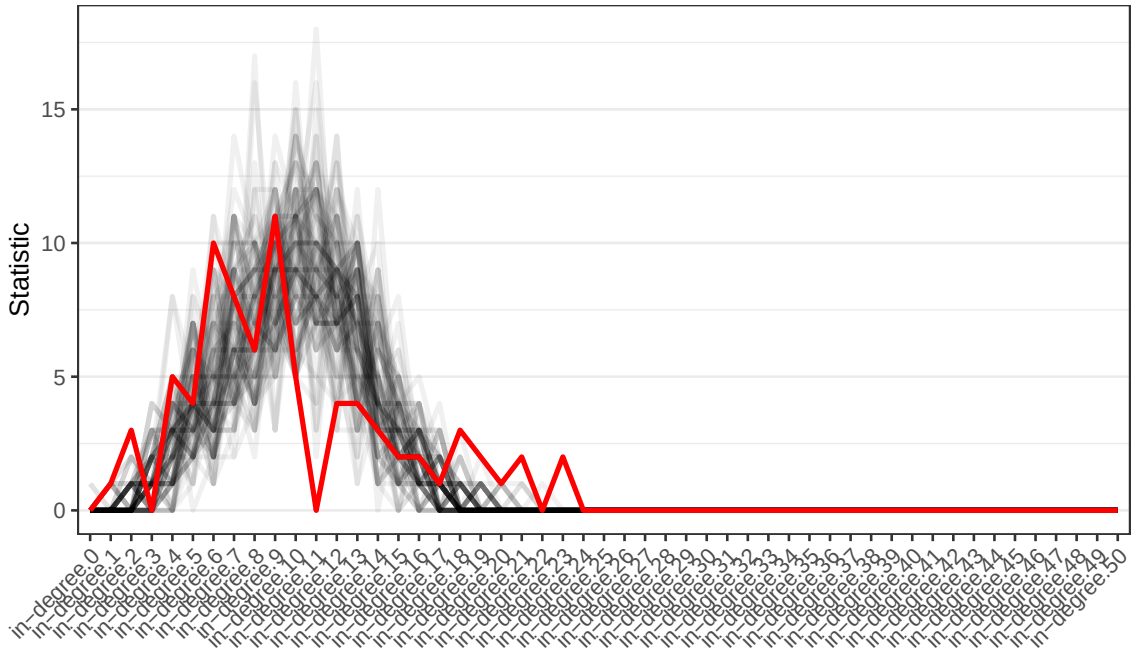
```
suppressPackageStartupMessages(library(ergm))
ergm(ukFaculty ~ edges() + nodematch("GroupC") + nodefactor("GroupC", levels=1:2))
#> Starting maximum pseudolikelihood estimation (MPLE):
#> Obtaining the responsible dyads.
#> Evaluating the predictor and response matrix.
#> Maximizing the pseudolikelihood.
#> Finished MPLE.
#> Evaluating log-likelihood at the estimate.
#>
#> Call:
#> ergm(formula = ukFaculty ~ edges() + nodematch("GroupC") + nodefactor("GroupC",
#>     levels = 1:2))
#>
#> Maximum Likelihood Coefficients:
#>                edges     nodematch.GroupC  nodefactor.GroupC.1
#>             -3.72285              2.71643              0.09184
#> nodefactor.GroupC.2
#>              0.20455
```

At this point we will evaluate the fit of our first LOLOG model by comparing the in-degree, out-degree and esp distribution of graphs simulated from the model to our observed graph.

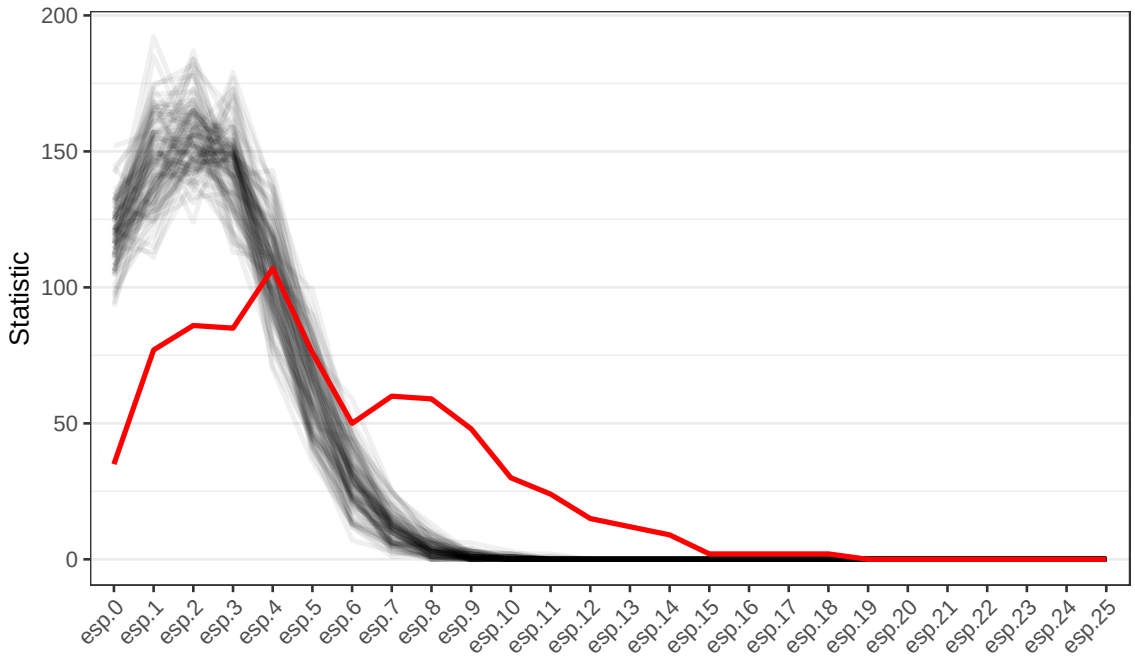
```
g <- gofit(fitukInd, ukFaculty ~ degree(0:50,"out"))
plot(g)
```




```
g <- gofit(fitukInd, ukFaculty ~ degree(0:50,"in"))
plot(g)
```

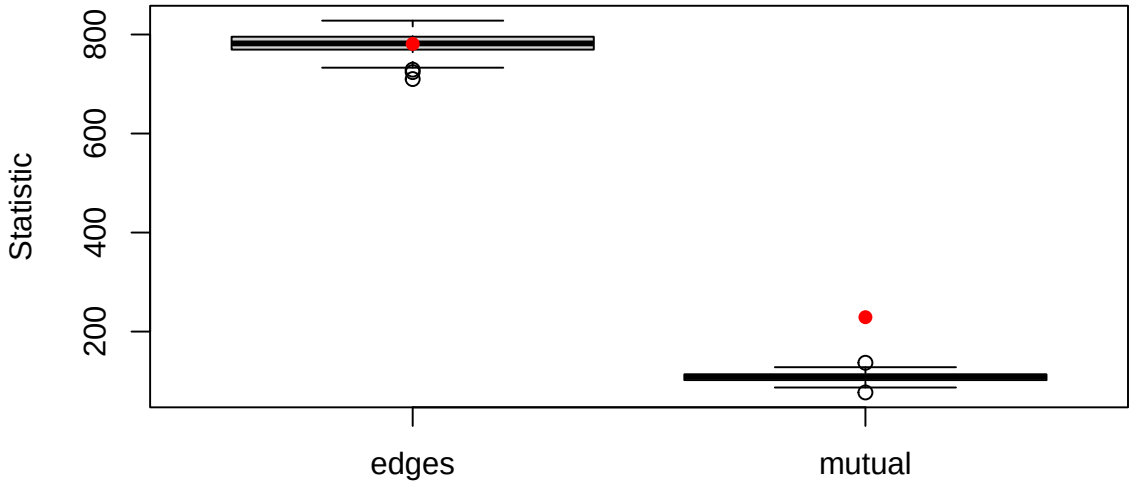



```
g <- gofit(fitukInd, ukFaculty ~ esp(0:25))
plot(g)
```




The statistics of the simulated networks are traced by the black lines, while out observed network is marked in red. The degree distributions are not terribly mismatched (out-degree is not great though), but the ESP distribution indicates simulated networks have far less transitivity than the observed network. Additionally, the number of reciprocated ties (mutual) is far too low in the simulated graphs

```
g <- gofit(fitukInd, ukFaculty ~ edges +mutual)
plot(g,type="box")
```




Okay, so let's try adding a `triangles` term for transitivity. For dyad dependent models the `nsamp` parameter controls how many samples are used for MC-GMM estimation. If you have a lot of terms in your model and are having trouble with convergence, try increasing this parameter.

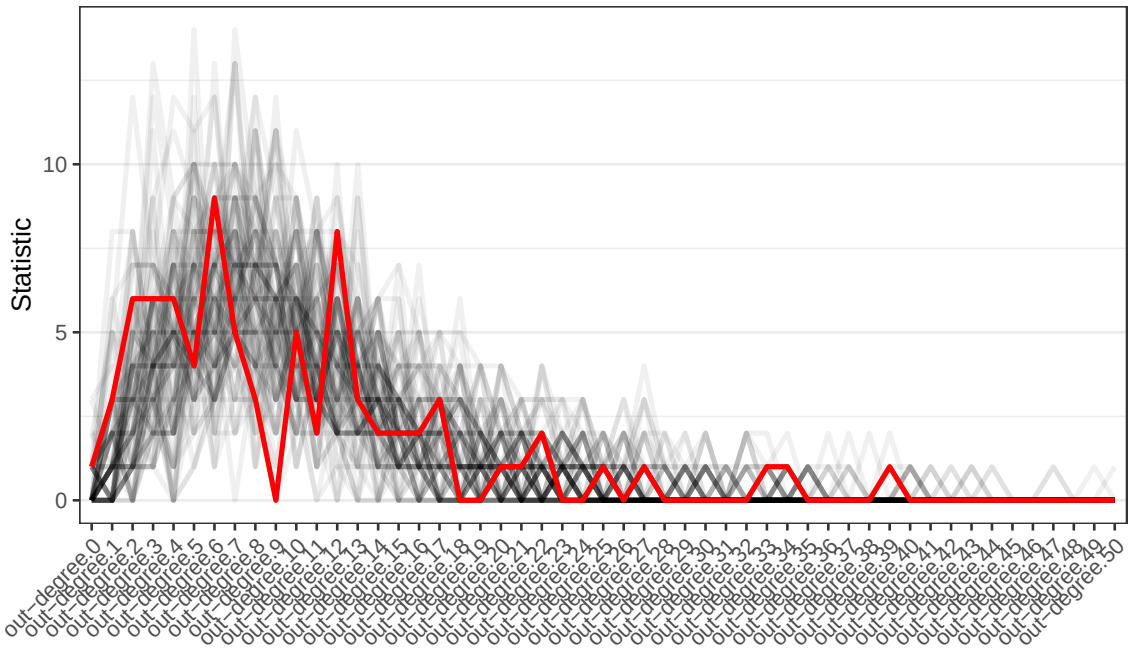
```
fituk <- lolog(ukFaculty ~ edges() + nodeMatch("GroupC") + nodeFactor("GroupC") +
            triangles, nsamp=2000, verbose=FALSE)
summary(fituk)
#>                     observed_statistics       theta         se pvalue
#> edges                               781 -4.17904957 0.40989894 0.0000
#> nodematch.GroupC                    663  2.34986841 0.23018392 0.0000
```

```
#> nodeFactor.GroupC.1          733 -0.15024955 0.09506938 0.1140
#> nodeFactor.GroupC.2          584 -0.04812598 0.10723060 0.6536
#> triangles                   5139  0.67418661 0.29956269 0.0244
```

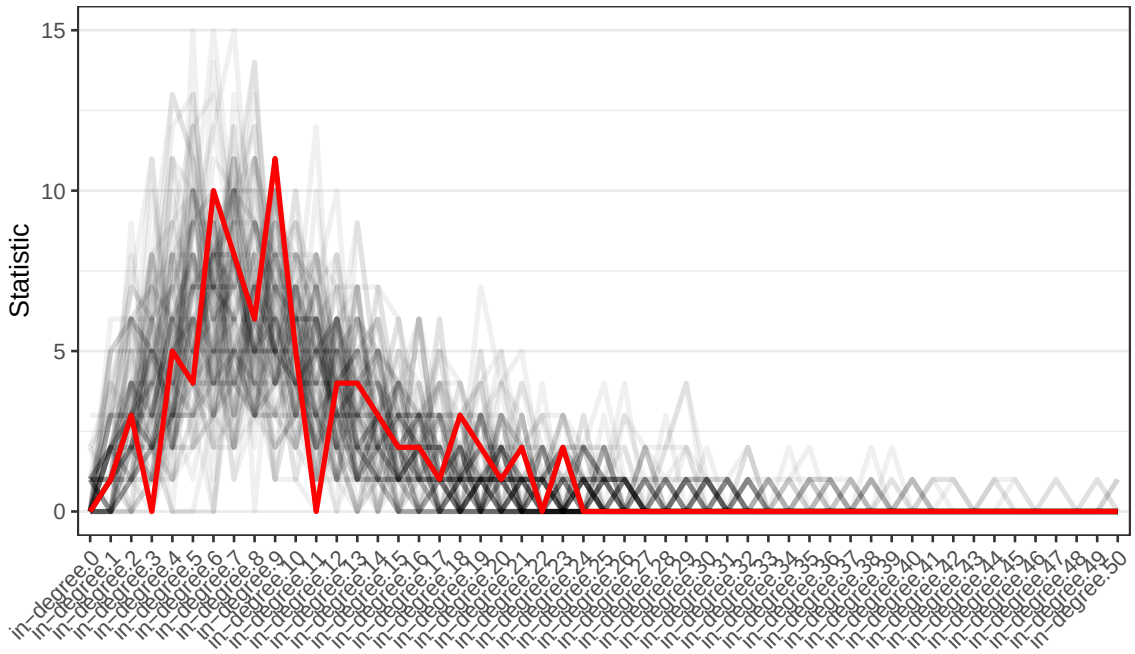
The `triangles` term is significant, but now the `nodeFactor` terms are not. This can be intuitively explained by the fact that groups 1 and 2 are larger than group 3. Because the matching term is so highly significant, there are more opportunities for within group clustering than in group 3, so when triangles are added, the larger activity within these groups is explained by the clustering activity.

Now let's look at those goodness of fit plots again...

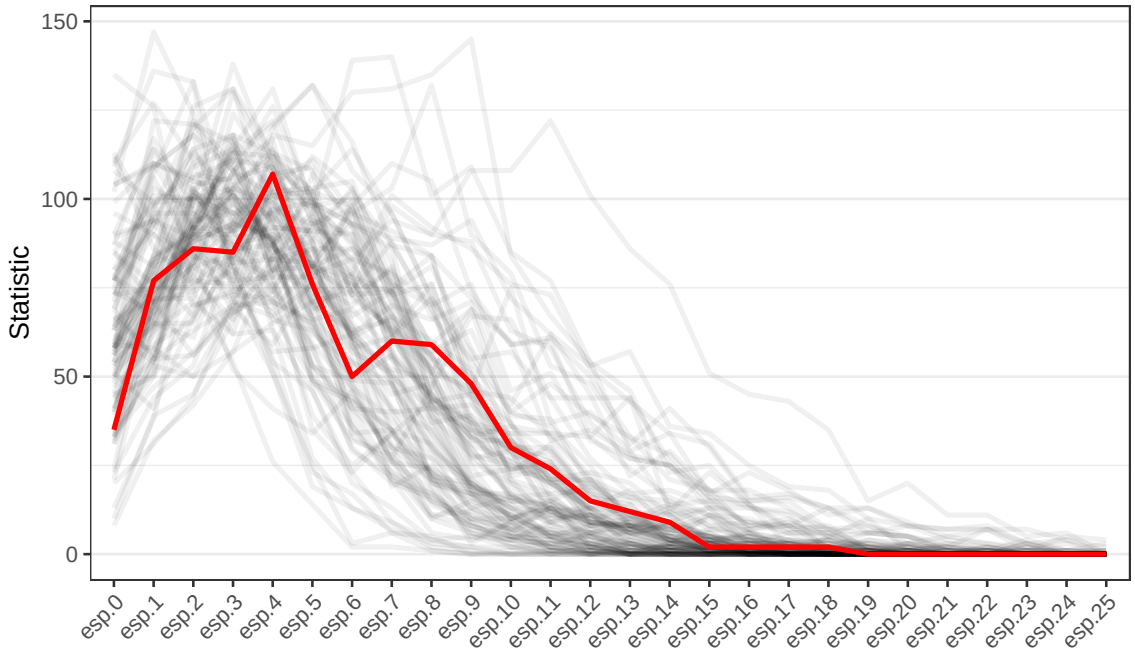
```
g <- gofit(fituk, ukFaculty ~ degree(0:50,"out"))
plot(g)
```



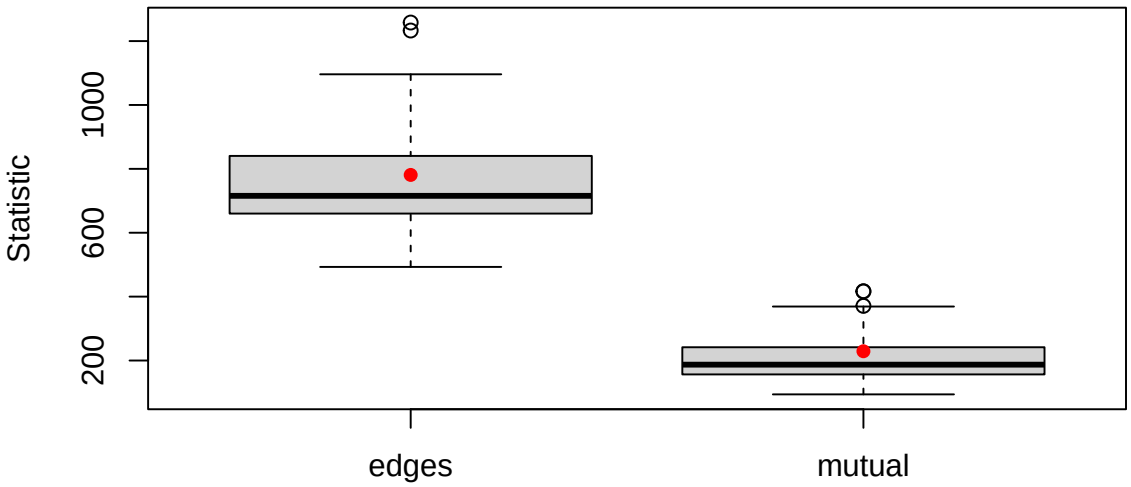

```
g <- gofit(fituk, ukFaculty ~ degree(0:50,"in"))
plot(g)
```




```
g <- gofit(fituk, ukFaculty ~ esp(0:25))
plot(g)
```

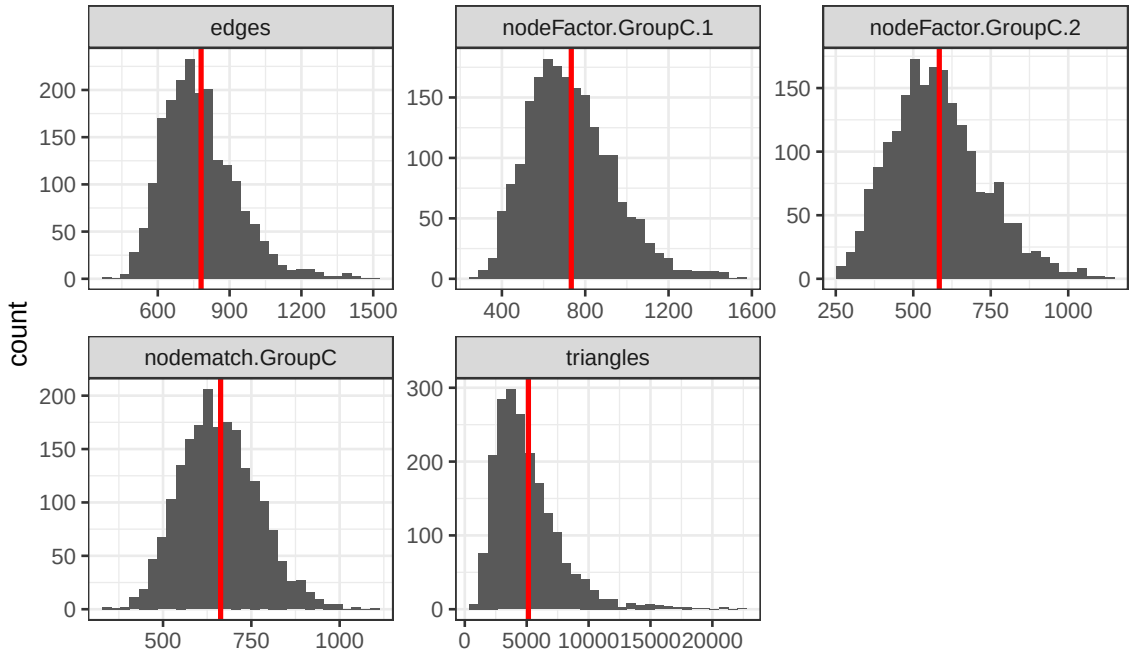



```
g <- gofit(fituk, ukFaculty ~ edges + mutual)
plot(g, type="box")
```




These look pretty good, with the observed values falling within the range of values simulated from the model. Additionally, because `lolog` matches the expected model graph statistics with their observed values, we are assured that statistics included in the model will have good goodness of fit. We can see this by plotting the model diagnostics.

```
plot(fituk)
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




## Attempting to fit with ERGM

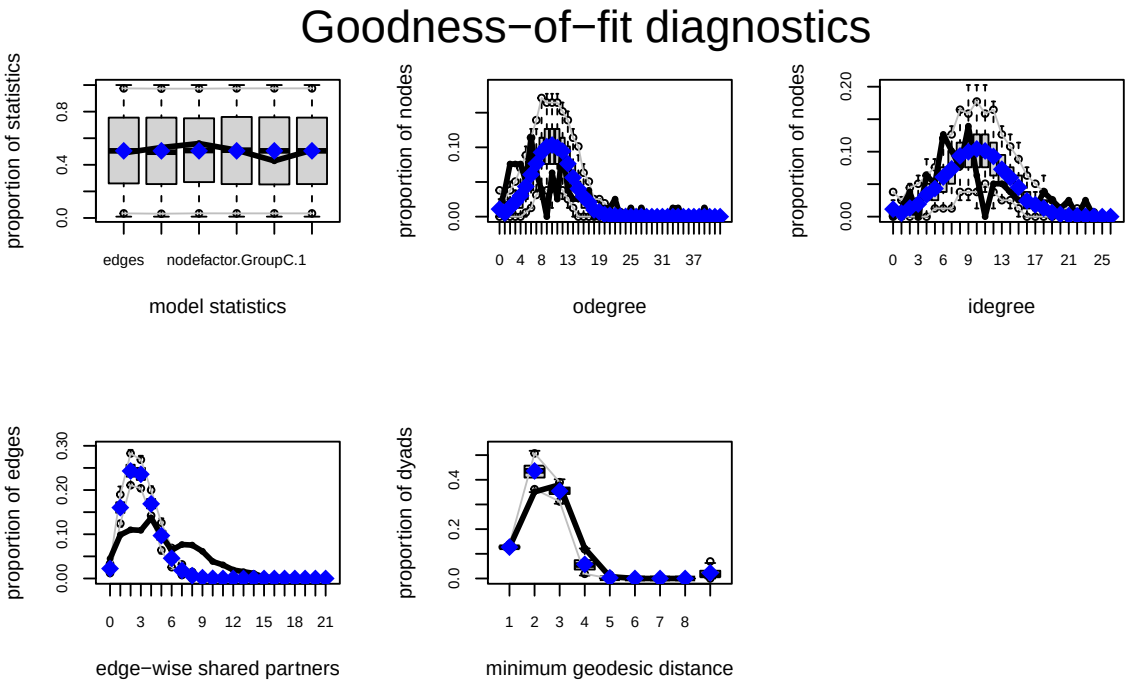
Like LOLOGs, ERGMs are an incredibly flexible model class. However, they are prone to model degeneracy, so it is recommended that great care is taken in choosing appropriate model statistics to use. Even using best practices in choosing these statistics it is not unusual to be unable to fit a network due to degeneracy related issues.

When modeling transitivity, the best practice for ERGMs is to include a gwesp term, which is "robust" to model degeneracy. We attempted many different models using this term (and others), and the best

fitting non-degenerate model that we could find fixed the decay parameter at 0.25. Larger values exhibited degeneracy problems, and allowing the parameter to be fit using curved ERGM estimation failed to converge.

```
fitukErgm <- ergm(ukFaculty ~ edges() + mutual + nodematch("GroupC") + nodefactor("GroupC", levels=1:2)
            gwesp(decay=.25, fixed=TRUE), verbose=FALSE)
#> Starting maximum pseudolikelihood estimation (MPLE):
#> Obtaining the responsible dyads.
#> Evaluating the predictor and response matrix.
#> Maximizing the pseudolikelihood.
#> Finished MPLE.
#> Starting Monte Carlo maximum likelihood estimation (MCMLE):
#> Iteration 1 of at most 60:
#> Warning: 'glpk' selected as the solver, but package 'Rglpk' is not available;
#> falling back to 'lpSolveAPI'. This should be fine unless the sample size and/or
#> the number of parameters is very big.
#> Optimizing with step length 0.2143.
#> The log-likelihood improved by 2.0292.
#> Estimating equations are not within tolerance region.
#> Iteration 2 of at most 60:
#> Optimizing with step length 0.2434.
#> The log-likelihood improved by 2.4484.
#> Estimating equations are not within tolerance region.
#> Iteration 3 of at most 60:
#> Optimizing with step length 0.4487.
#> The log-likelihood improved by 3.1606.
#> Estimating equations are not within tolerance region.
#> Iteration 4 of at most 60:
#> Optimizing with step length 0.9020.
#> The log-likelihood improved by 3.3962.
#> Estimating equations are not within tolerance region.
#> Iteration 5 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 1.2115.
#> Estimating equations are not within tolerance region.
#> Iteration 6 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.6301.
#> Estimating equations are not within tolerance region.
#> Iteration 7 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.0541.
#> Convergence test p-value: 0.7479. Not converged with 99% confidence; increasing sample size.
#> Iteration 8 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.0191.
#> Convergence test p-value: 0.4448. Not converged with 99% confidence; increasing sample size.
#> Iteration 9 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.0257.
#> Convergence test p-value: 0.9093. Not converged with 99% confidence; increasing sample size.
#> Iteration 10 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.0166.
#> Convergence test p-value: 0.1053. Not converged with 99% confidence; increasing sample size.
```

```
#> Iteration 11 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.0942.
#> Convergence test p-value: 0.8603. Not converged with 99% confidence; increasing sample size.
#> Iteration 12 of at most 60:
#> Optimizing with step length 1.0000.
#> The log-likelihood improved by 0.0422.
#> Convergence test p-value: 0.0001. Converged with 99% confidence.
#> Finished MCMLE.
#> Evaluating log-likelihood at the estimate. Fitting the dyad-independent submodel...
#> Bridging between the dyad-independent submodel and the full model...
#> Setting up bridge sampling...
#> Using 16 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .
#> Bridging finished.
#>
#> This model was fit using MCMC.  To examine model diagnostics and check
#> for degeneracy, use the mcmc.diagnostics() function.
summary(fitukErgm)
#> Call:
#> ergm(formula = ukFaculty ~ edges() + mutual + nodematch("GroupC") +
#>     nodefactor("GroupC", levels = 1:2) + gwesp(decay = 0.25,
#>     fixed = TRUE), verbose = FALSE)
#>
#> Monte Carlo Maximum Likelihood Results:
#>
#>                       Estimate Std. Error MCMC % z value Pr(>|z|)
#> edges                -5.632096   0.204627      0 -27.524   <1e-04 ***
#> mutual                2.011986   0.141418      0  14.227   <1e-04 ***
#> nodematch.GroupC      1.255445   0.081825      0  15.343   <1e-04 ***
#> nodefactor.GroupC.1   0.008381   0.041092      0   0.204   0.8384
#> nodefactor.GroupC.2   0.086997   0.048070      0   1.810   0.0703 .
#> gwesp.OTP.fixed.0.25  1.760734   0.173935      0  10.123   <1e-04 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>      Null Deviance: 8542  on 6162  degrees of freedom
#>  Residual Deviance: 3217  on 6156  degrees of freedom
#>
#> AIC: 3229  BIC: 3269  (Smaller is better. MC Std. Err. = 0.8855)
g <- gof(fitukErgm)
par(mfrow=c(2,3))
plot(g)
```




The added the gwesp term is highly significant, indicating increased levels of transitivity; however, the goodness of fit plot shows that the ERGM is not capturing the full amount of transitivity in the network. Simulated networks have much lower esp values than the one observed. Unfortunately, using the recommended practices for fitting ERGMs, we were unable to find a non-degenerate ERGM that appropriately captures the degree and transitivity patterns of this dataset.