Adequacy and Degeneracy in ERG Models

SOC 280: Analysis of Social Network Data

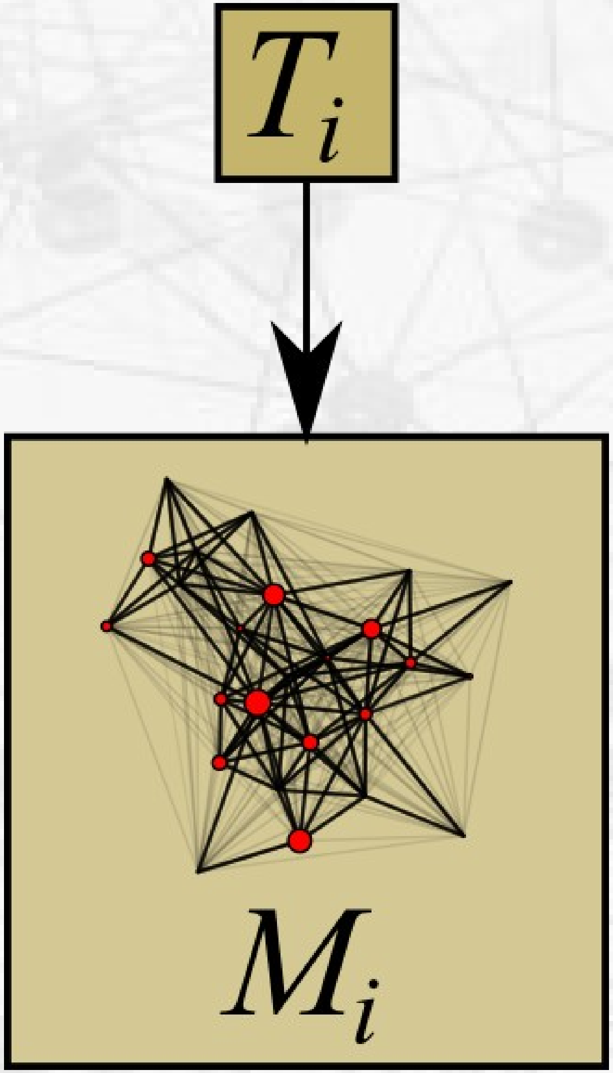
Carter T. Butts
Department of Sociology and
Institute for Mathematical Behavioral Sciences
University of California, Irvine

From Model Fitting to Model Assessment

- **Last time, we saw how to construct (and fit) nontrivial ERGs**
  - Started with dyad independent terms
  - Added basic dependence terms
  - Fit the whole thing via MLE
- **Today's focus: degeneracy checking and model assessment**
  - Looking under the hood to make sure that the engine is still running (and, occasionally, getting out to turn the crank)
  - Checking the results to ensure that the model makes sense

The Role of Simulation in ERG Research

- **Simulation is central to ERG modeling**
  - Even simple models too complex to get analytical solutions – need to use simulation to study model behavior, make predictions
  - ERG computations too difficult to perform directly; simulation used for purely computational purposes (e.g., dealing with normalizing factors)
- **Implication: we need to know something about ERG simulation to use the tools effectively**



Simulating ERGs via MCMC

- **Simulation of simple models like N,p, $U|MAN$ was plug-n-play....**
  - Yours truly had to worry about how it was done, but otherwise everything worked as a "black box"
  - Even behind the curtain, simulation is exact (and fairly easy)
- **....not so for more general ERGs**
  - No (effective) way to draw directly from ERG distribution; have to use approximation algorithms (not counting recent work by yours truly)
  - Primary tool: Markov chain Monte Carlo (MCMC)
    - Iterative method for simulating draws from a given distribution
    - Algorithm is approximate (although often very, very good)
- **So what?**
  - MCMC requires more "care and feeding" than simple methods
  - Algorithm can fail, requiring user intervention (i.e., get out and push)

Mc, MC, and MCMC In One Slide

- **Markov chain**
  - Stochastic process $X_1,X_2,...$ on state space S, such that $p(X_i|X_{i-1},X_{i-2},...)=p(X_i|X_{i-1})$ (i.e., only the previous state matters – this is the *Markov condition*)
- **Monte Carlo procedure**
  - Any procedure which uses randomization to perform a computation, having a fixed execution time and uncertain output (compare w/Las Vegas procedures)
- **Markov chain Monte Carlo (MCMC)**
  - Family of procedures using Markov chains to perform computations and/or simulate target distributions; often, these cannot be done any other way
- **Important Example: Metropolis Algorithm**
  - Given X_i, draw X' from $q(X_i)$; w/probability $\min(1,p(X')/p(X_i))$, let $X_{i+1}=X'$, else let $X_{i+1}=X_i$. Repeat for $i+1$, $i+2$, etc.
  - Started w/arbitrary X_0, $X_0,X_1,...X_n$ converges to $p(X)$ in distribution as $n \rightarrow \infty$
  - Requires some constraints on q, but is very general – used when we can't sample from target distribution p directly (as when p is an ERG distribution)

ERG MCMC In One More Slide

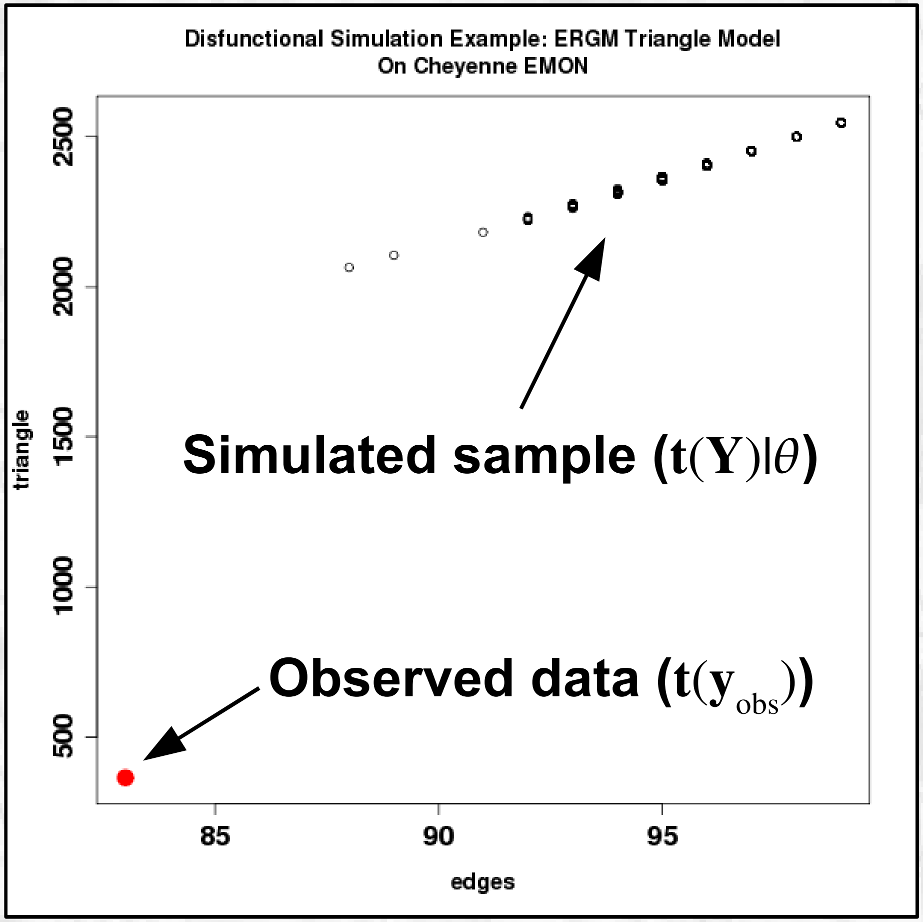
- **When we need to simulate general ERGs, we turn to MCMC**
  - Fairly easy to implement as a general approach
  - Only requires ratios of the form $\Pr(\mathbf{Y}'|\theta)/\Pr(\mathbf{Y}|\theta)$, which (as we saw last time) reduce to $\exp(\theta^T(\mathbf{t}(\mathbf{Y}')-\mathbf{t}(\mathbf{Y})))$ (no normalizing constant!)
  - General procedure: starting with seed graph (random graph or data), run Markov chain
    - Early “burn-in” draws contaminated by initial state – we discard
    - Need to ensure that sample is large enough to have good properties
    - Both aspects sloppily called “convergence” (though true convergence is asymptotic); the chain has “converged” when approximation is adequate
- **In `statnet`, handled by `ergm` and `simulate` commands**
  - Lots under the hood that can be ignored (for now)
  - Mostly automated, but important to use diagnostics to verify behavior

Simulation and Inference

- **One reason ERG simulation matters: we need it for likelihood-based inference**
  - Recall that $\Pr(\mathbf{Y}=\mathbf{y}|\theta)$ includes an effectively incomputable normalizing factor that depends on θ
  - Likelihood calculation uses a version of *importance sampling* to estimate the normalizing factor (or, more exactly, ratios thereof)
- **What happens when you run `ergm`**
  - Little gnomes make an initial guess at θ using the MPLE
  - More gnomes simulate $\mathbf{y}_1,\ldots,\mathbf{y}_n$ based on the initial guess
  - The simulated sample is used to find θ using MLE
  - Possibly, the previous two steps are iterated a few times for good measure (since initial estimate may be off)

When Simulation Fails

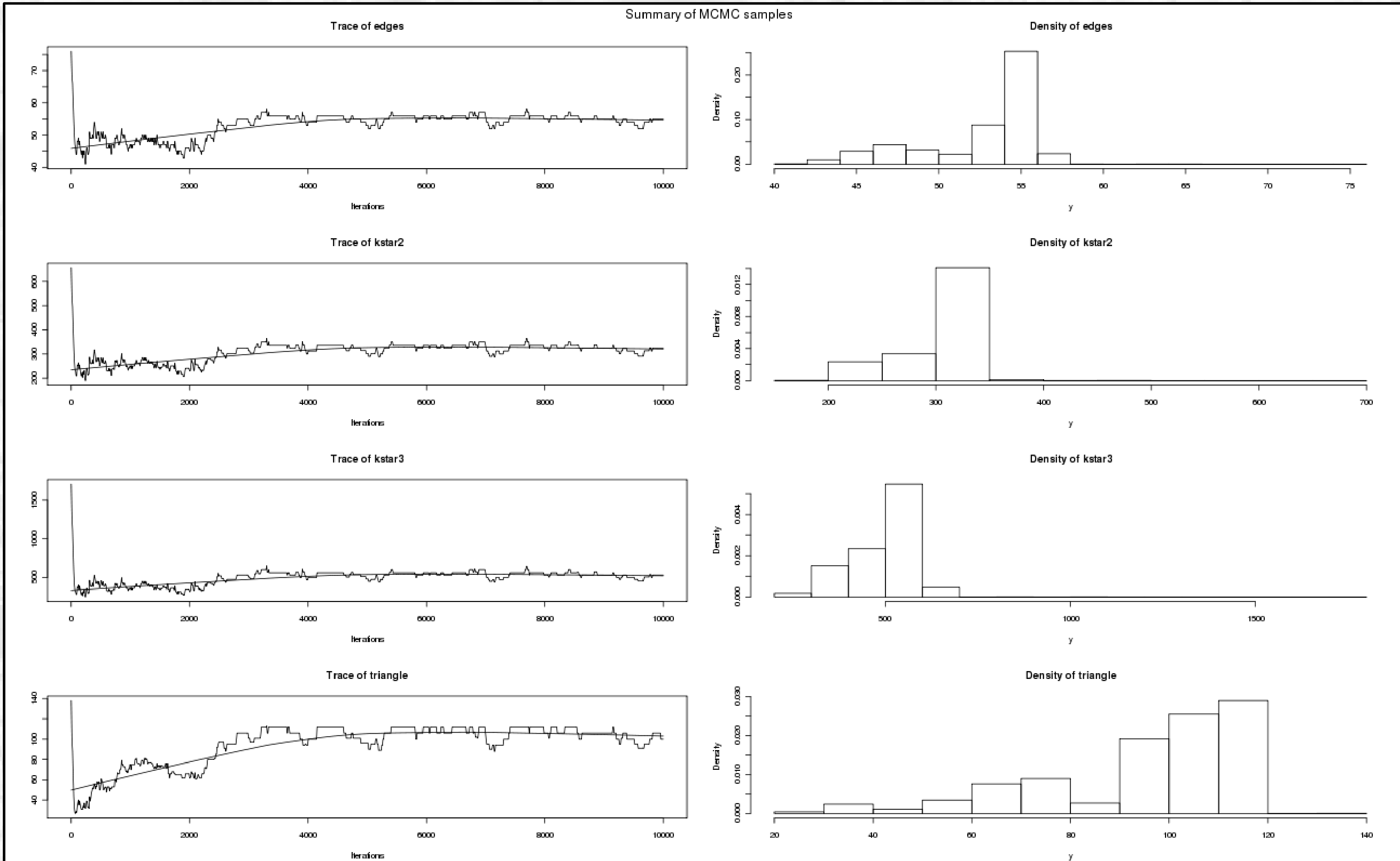
- **Simulation can fail in several (essentially four) ways**
  - Insufficient burn-in: starting point still affects results
  - Insufficient post-burn samples: sample hasn't converged
  - May be degenerate: almost all graphs are same (usually complete/empty)
  - Sample does not cover observed graph (problematic for inference)
- **Want to examine output to check for these problems...**



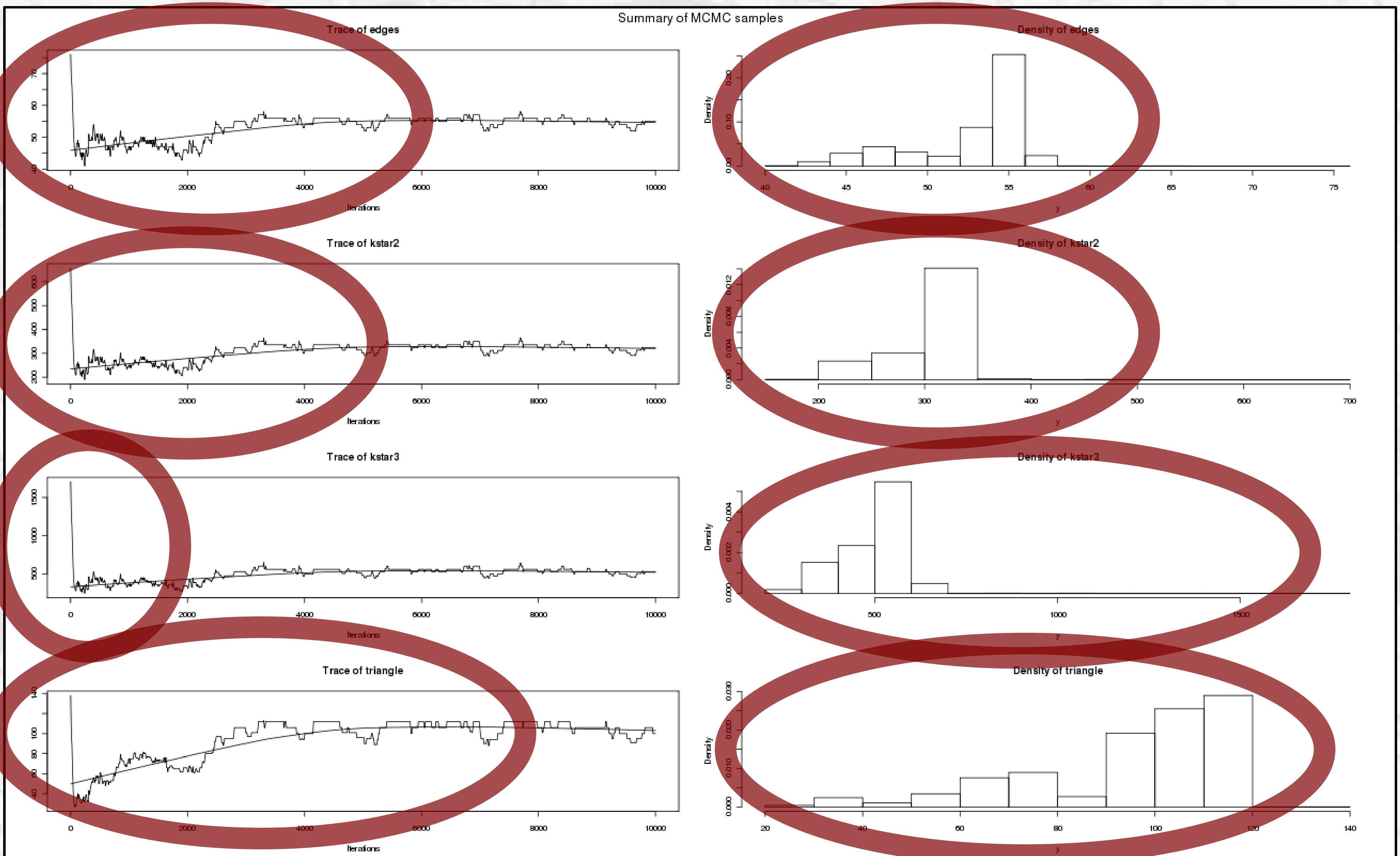
Assessing Simulation Quality

- **No foolproof method, but several heuristics**
- **In `ergm`, primary tool is `mcmc.diagnostics`**
  - Requires `coda` library
  - Calculates various diagnostics on MCMC output
    - Correlations and lagged correlations of model statistics
    - Raftery-Lewis convergence diagnostics
  - Basic syntax: `mcmc.diagnostics(fit)`
    - `fit`: `ergm` object (i.e., fitted model) <u>or</u> MCMC output
- **Can also directly plot statistics vs. observed**
  - `fit$sample` provides normalized simulated statistics from an `ergm` object (0=observed value)

Example: flobusiness w/edges, 2-3 stars, triangles (no thinning)



Example: flobusiness w/edges, 2-3 stars, triangles (no thinning)



```
> mcmc.diagnostics(fit, center=F)

Correlations of sample statistics:
, , cor

             edges    kstar2    kstar3  triangle
edges    1.0000000 0.9897132 0.9558623 0.8990734
kstar2   0.9897132 1.0000000 0.9863819 0.8777416
kstar3   0.9558623 0.9863819 1.0000000 0.8103664
triangle 0.8990734 0.8777416 0.8103664 1.0000000

, , lag1

             edges    kstar2    kstar3  triangle
edges    0.9958246 0.9842669 0.9490012 0.8960903
kstar2   0.9844877 0.9929200 0.9771626 0.8738368
kstar3   0.9493799 0.9773312 0.9878666 0.8053863
triangle 0.8988158 0.8774429 0.8101006 0.9990596


r=0.0125 and 0.9875:

Quantile (q) = 0.025
Accuracy (r) = +/- 0.0125
Probability (s) = 0.95

          Burn-in  Total   Lower bound  Dependence enough   enough
          (M)      (N)     (Nmin)       factor (I) burn-in? samples?
 edges    1150     1952976 600          3250         no       no
 kstar2   160      54954   600          170          no       no
 kstar3   97       17215   600          100          no       no
 triangle 1440     2547600 600          4250         no       no
```

```
> mcmc.diagnostics(fit, center=F)

Correlations of sample statistics:
, , cor

             edges    kstar2    kstar3  triangle
edges    1.0000000 0.9897132 0.9558623 0.8990734
kstar2   0.9897132 1.0000000 0.9863819 0.8777416
kstar3   0.9558623 0.9863819 1.0000000 0.8103664
triangle 0.8990734 0.8777416 0.8103664 1.0000000

, , lag1

             edges    kstar2    kstar3  triangle
edges    0.9958246 0.9842669 0.9490012 0.8960903
kstar2   0.9844877 0.9929200 0.9771626 0.8738368
kstar3   0.9493799 0.9773312 0.9878666 0.8053863
triangle 0.8988158 0.8774429 0.8101006 0.9990596


r=0.0125 and 0.9875:

Quantile (q) = 0.025
Accuracy (r) = +/- 0.0125
Probability (s) = 0.95

          Burn-in  Total   Lower bound  Dependence enough   enough
          (M)      (N)     (Nmin)       factor (I) burn-in? samples?
 edges    1150     1952976 600          3250         no        no
 kstar2   160      54954   600          170          no        no
 kstar3   97       17215   600          100          no        no
 triangle 1440     2547600 600          4250         no        no
```

High cross-correlations

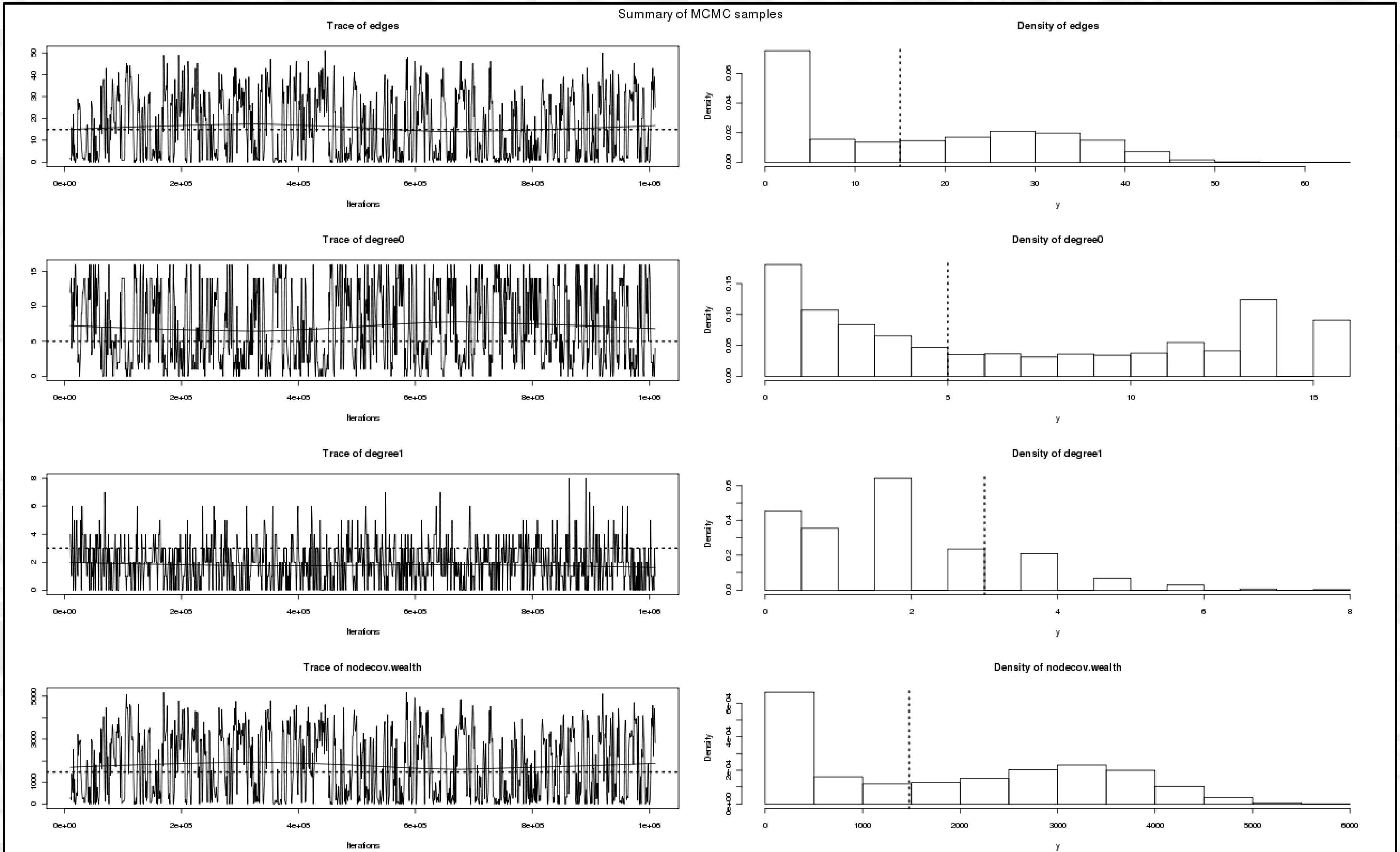
High autocorrelation

Non-convergence

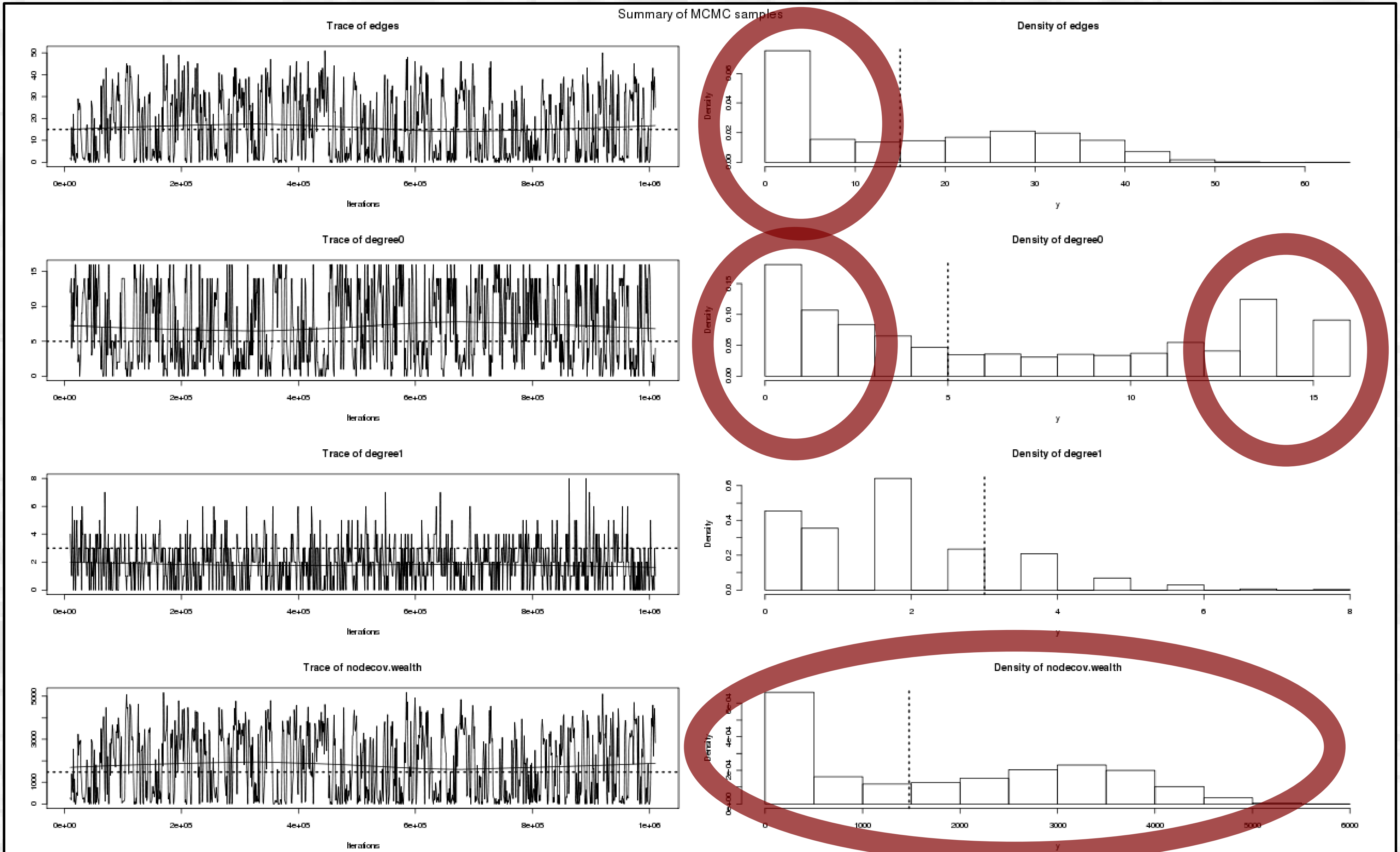
What To Do If Things Go Wrong?

- **If things go wrong, there are several palliatives**
  - For burn-in issues, increase `burnin` parameter
  - For post-burn convergence, increase `MCMCsamplesize` and/or interval parameters
    - `MCMCsamplesize` increases final sample (subsampled from larger set of MCMC draws)
    - `interval` increases subsampling interval; useful if chain mixes slowly (i.e., high autocorrelation and/or slow movement)
- **If none of these work, may need to change the model**
  - Poor convergence is often a sign of a bad model
    - E.g., degeneracy due to runaway clique formation
  - Triangle, star terms especially bad (due to “nesting”)
    - Try curved variants (to be discussed next time)

Example 2: flobusiness w/edges, 0/1 degree, wealth

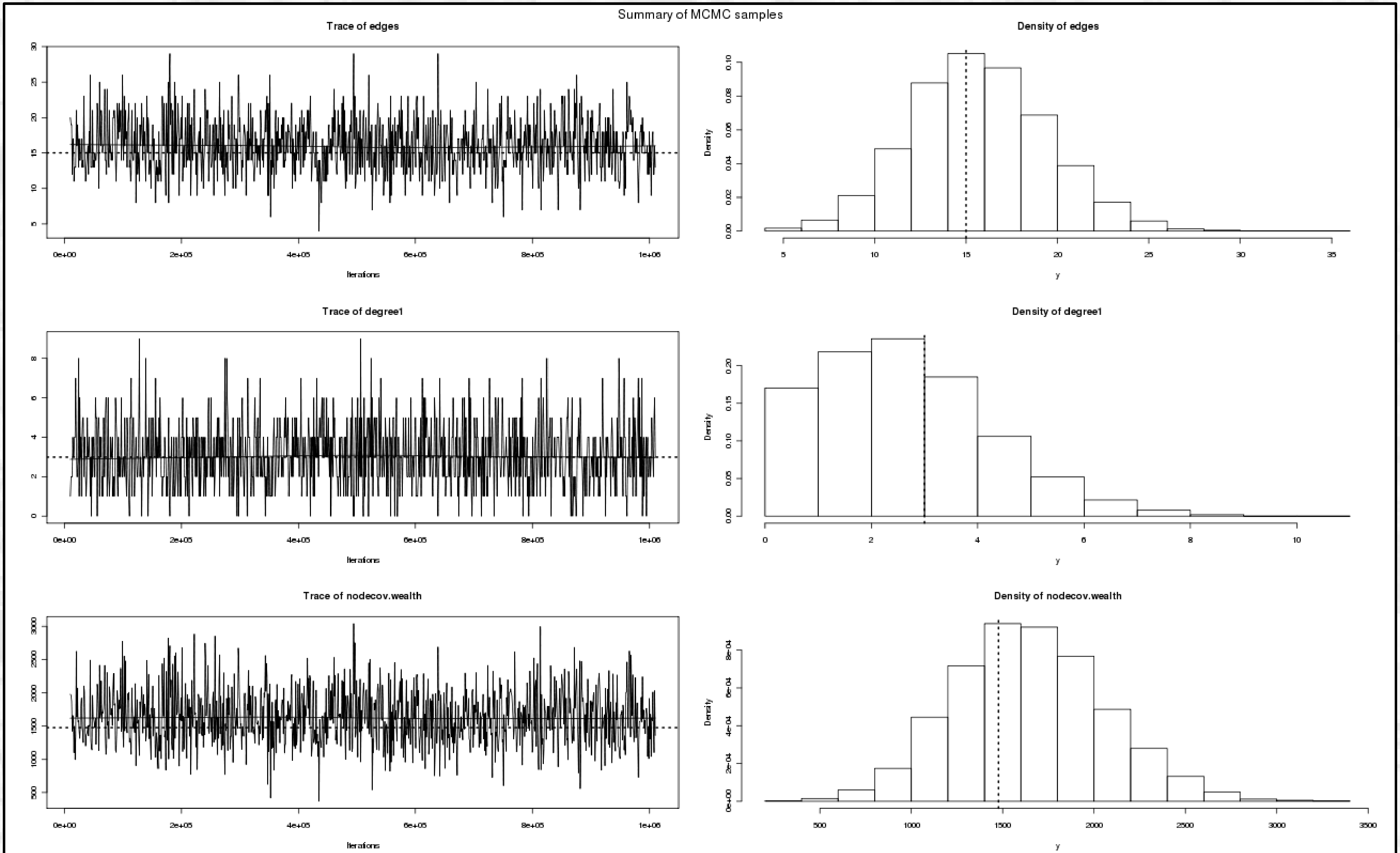


Example 2: flobusiness w/edges, 0/1 degree, wealth



```
> mcmc.diagnostics(fit, center=F)

Correlations of sample statistics:
, , cor

                    edges     degree0     degree1 nodecov.wealth
edges           1.0000000 -0.93797047 -0.32274203      0.9889985
degree0        -0.9379705  1.00000000  0.06218768     -0.9469523
degree1        -0.3227420  0.06218768  1.00000000     -0.2876239
nodecov.wealth  0.9889985 -0.94695232 -0.28762390      1.0000000

, , lag1

                    edges    degree0    degree1 nodecov.wealth
edges           0.9233889 -0.8690288 -0.3271741      0.9193287
degree0        -0.8697501  0.8624829  0.2606243     -0.8770274
degree1        -0.3317297  0.2634784  0.1741045     -0.3154555
nodecov.wealth  0.9176854 -0.8755447 -0.3097296      0.9238070


r=0.0125 and 0.9875:

Quantile (q) = 0.025
Accuracy (r) = +/- 0.0125
Probability (s) = 0.95

                Burn-in  Total   Lower bound  Dependence enough   enough
                (M)      (N)     (Nmin)       factor (I) burn-in? samples?
 edges          5400     5767200 600          9610       yes       no
 degree0        8800     2923800 600          9900       yes       no
 degree1        900      1529100 600          2550       yes       no
 nodecov.wealth 5400     5767200 600          9610       yes       no
```

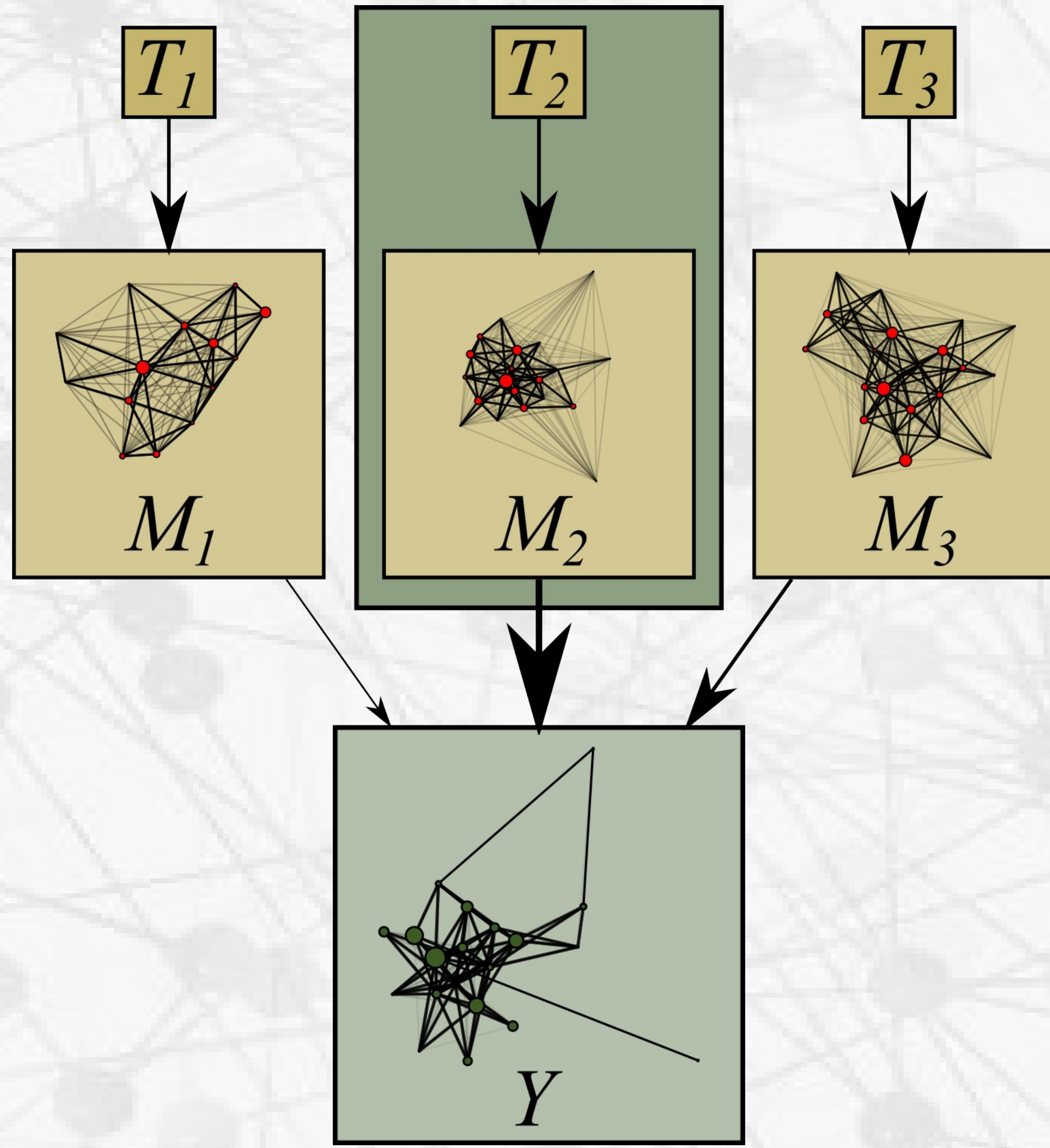
Example 3: flobusiness w/edges, 1-degree, wealth



```
> mcmc.diagnostics(fit, center=F)

Correlations of sample statistics:
, , cor

                    edges    degree1 nodecov.wealth
edges           1.0000000 -0.4665689      0.8779387
degree1        -0.4665689  1.0000000     -0.3933709
nodecov.wealth  0.8779387 -0.3933709      1.0000000

, , lag1

                     edges     degree1 nodecov.wealth
edges           0.09104357 -0.02522841     0.09795248
degree1        -0.02686050  0.01524997    -0.03244706
nodecov.wealth  0.10869165 -0.03262686     0.13800652


r=0.0125 and 0.9875:

Quantile (q) = 0.025
Accuracy (r) = +/- 0.0125
Probability (s) = 0.95

                Burn-in  Total Lower bound  Dependence enough   enough
                (M)      (N)   (Nmin)       factor (I) burn-in? samples?
 edges          200      78700 600          300        yes      yes
 degree1        200      89500 600          300        yes      yes
 nodecov.wealth 200      62100 600          300        yes      yes
```

From Diagnostics to Model Assessment

- **Recall logic ERG inference**
  - Theories imply probability models
  - Observed data more probable under certain models
  - Theories which account for data preferred
- **Model adequacy**
  - Model should reproduce relevant properties of observed data
  - Compare w/CUG tests, baseline models



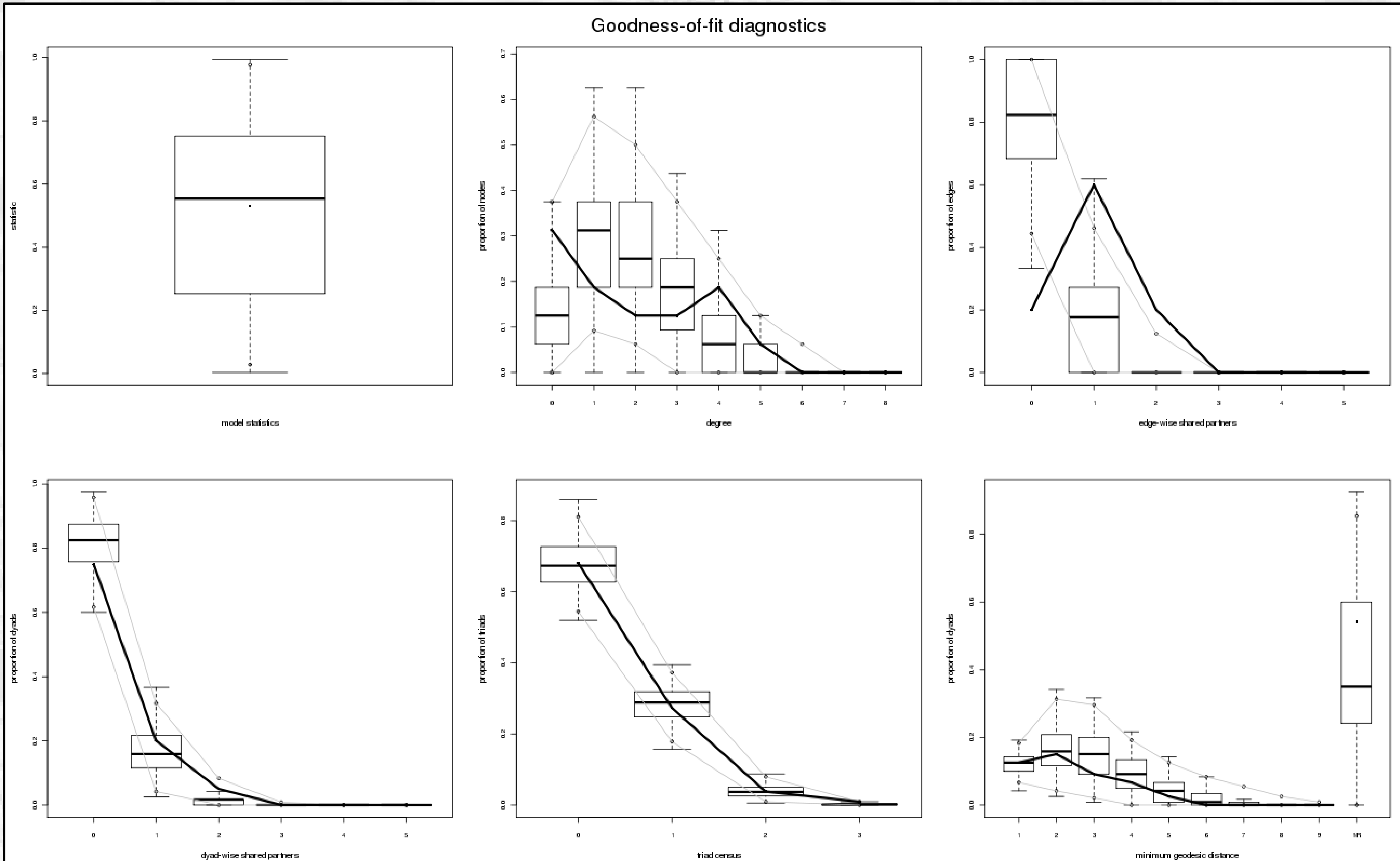
Assessing Adequacy

- **How does one assess model adequacy? Simulation!**
  - Simulate draws from fitted model
  - Compare observed graph to simulated graphs on measures of interest
  - Verify that observed properties are well-covered by simulated ones (e.g., not in 5% tails)
- **What properties should be considered?**
  - This is application-specific – no single uniform answer
  - Start with “in-model” statistics; ERG must get means right, but should still verify non-pathological distributions
  - “Out-of-model” statistics can be common low-level properties (e.g., degree, triad census), or theoretically motivated quantities
  - One approach: treat much as you would a CUG test, in terms of identifying meaningful stats to check

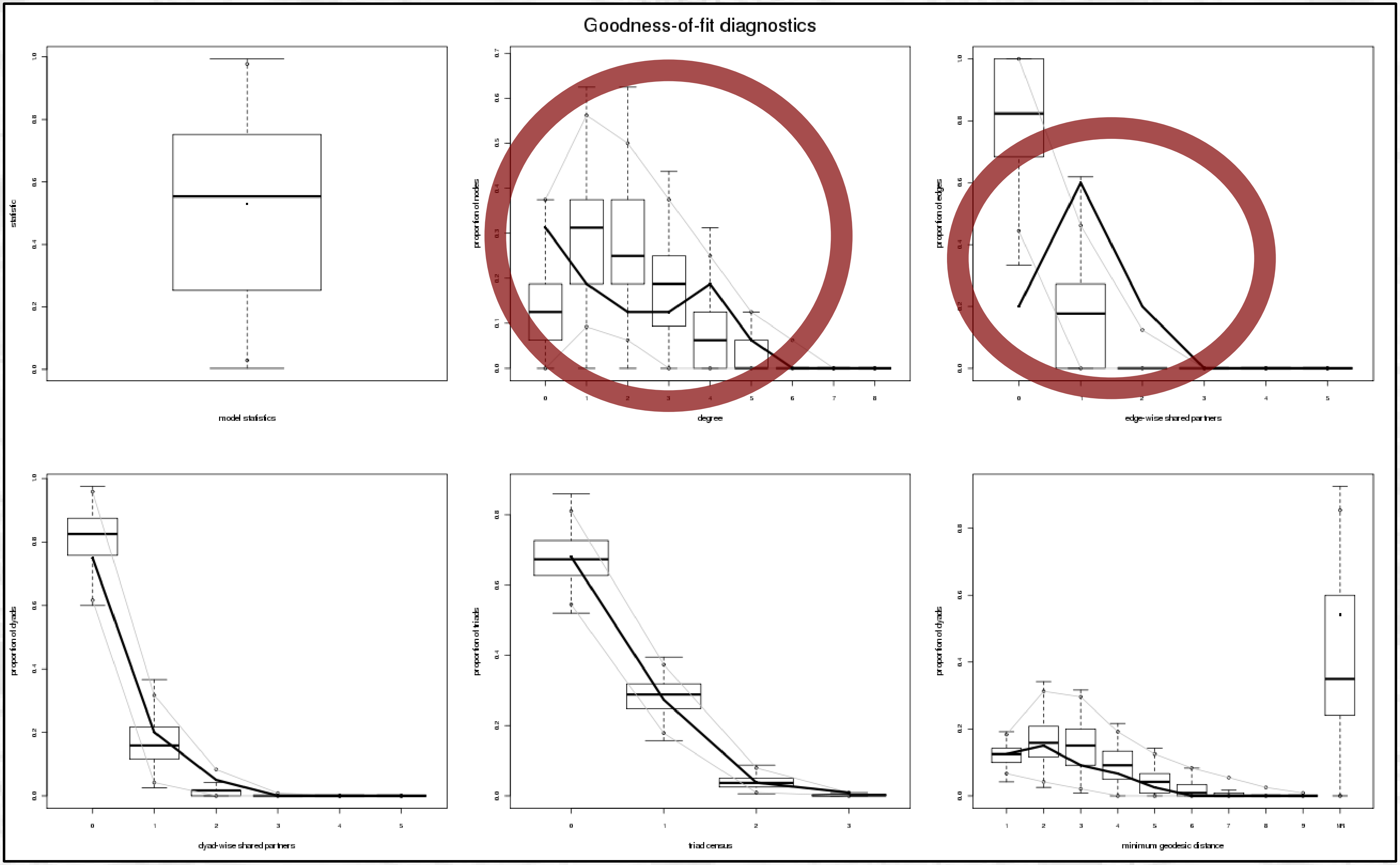
Checking Adequacy w/`gof`

- **`ergm` facilitates adequacy checking with the gof routine**
  - Simulates draws from an ERG model, calculates stats
  - Basic syntax: `gof(fit,GOF=~term1+term2)`
    - `fit` is an `ergm` object (i.e., a fitted model)
    - `term1`, `term2`, etc. indicate statistics to be used (e.g., `degree`, `distance`, `triadcensus`, `model`)
      - `model` includes all "in-model" stats – a good reality check!
  - Has `print`, `plot`, `summary` methods (`plot` especially useful)
  - Note: still uses MCMC, so check convergence....

Example: flobusiness w/edges



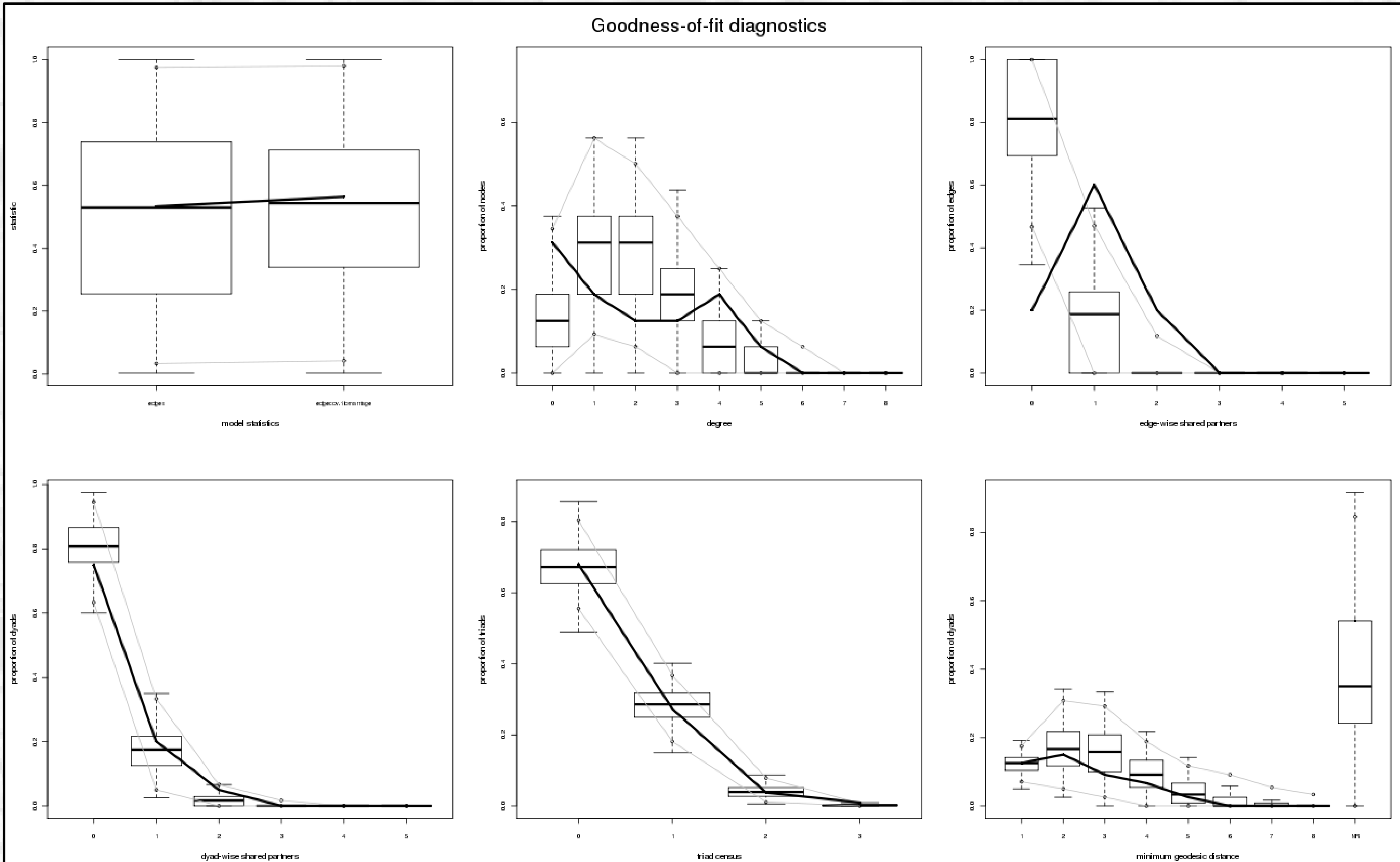
Example: flobusiness w/edges



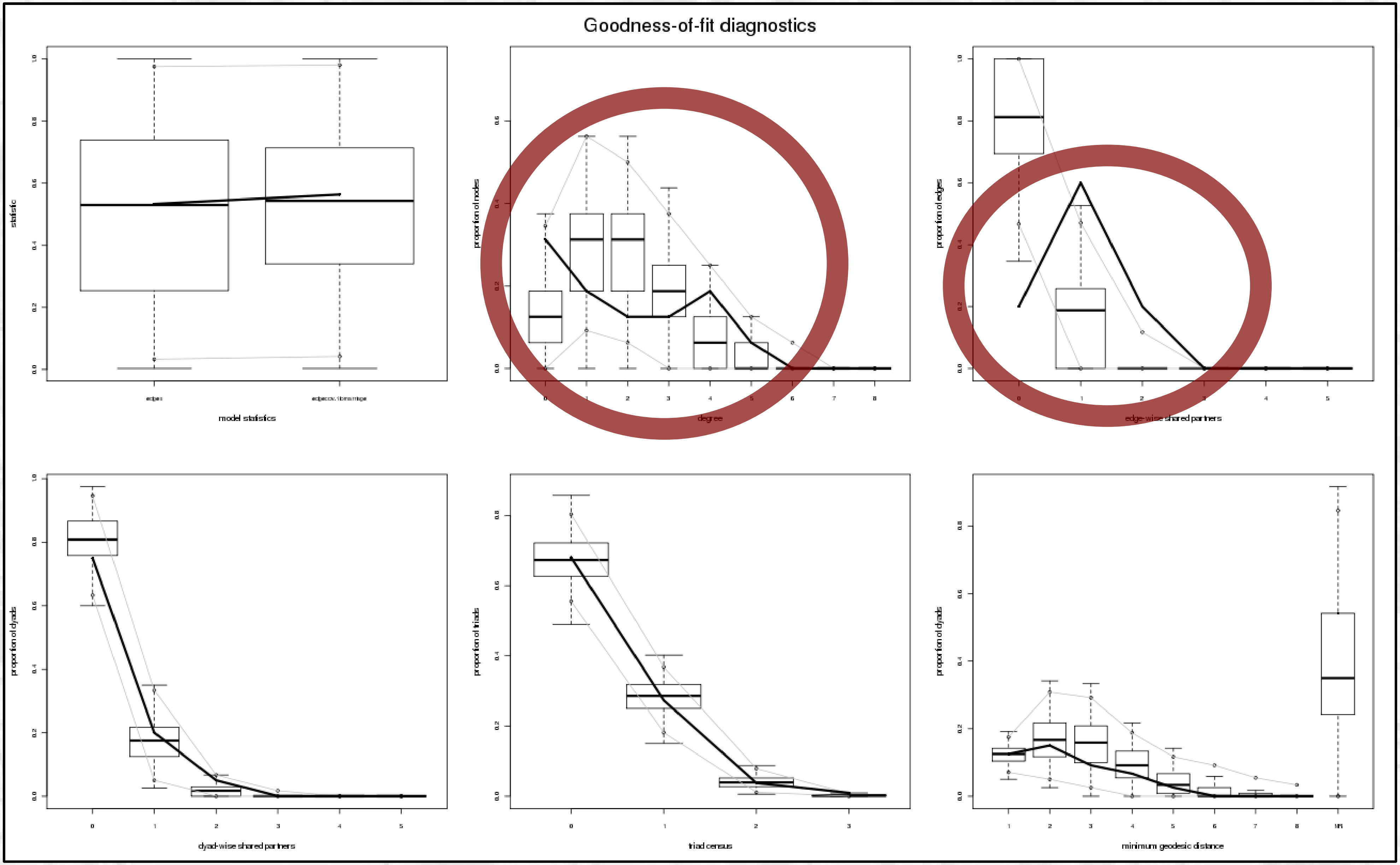
Coping with Inadequacy

- **What do we do if our model appears inadequate?**
  - Option 1: Add terms
    - Which features are poorly captured? Is there a term which would add in such effects (ideally minimally)?
  - Option 2: Switch terms
    - Can you replace an existing term with a similar one more likely to succeed? (E.g., sociality or degree terms versus *k*-stars)
  - Option 3: Do nothing
    - Is the type of inadequacy a problem for your specific question? Can it be tolerated in this case? How good is the overall fit?
- **As always, the right approach depends upon your goal – what do you need to predict/explain? (No model gets it all!)**

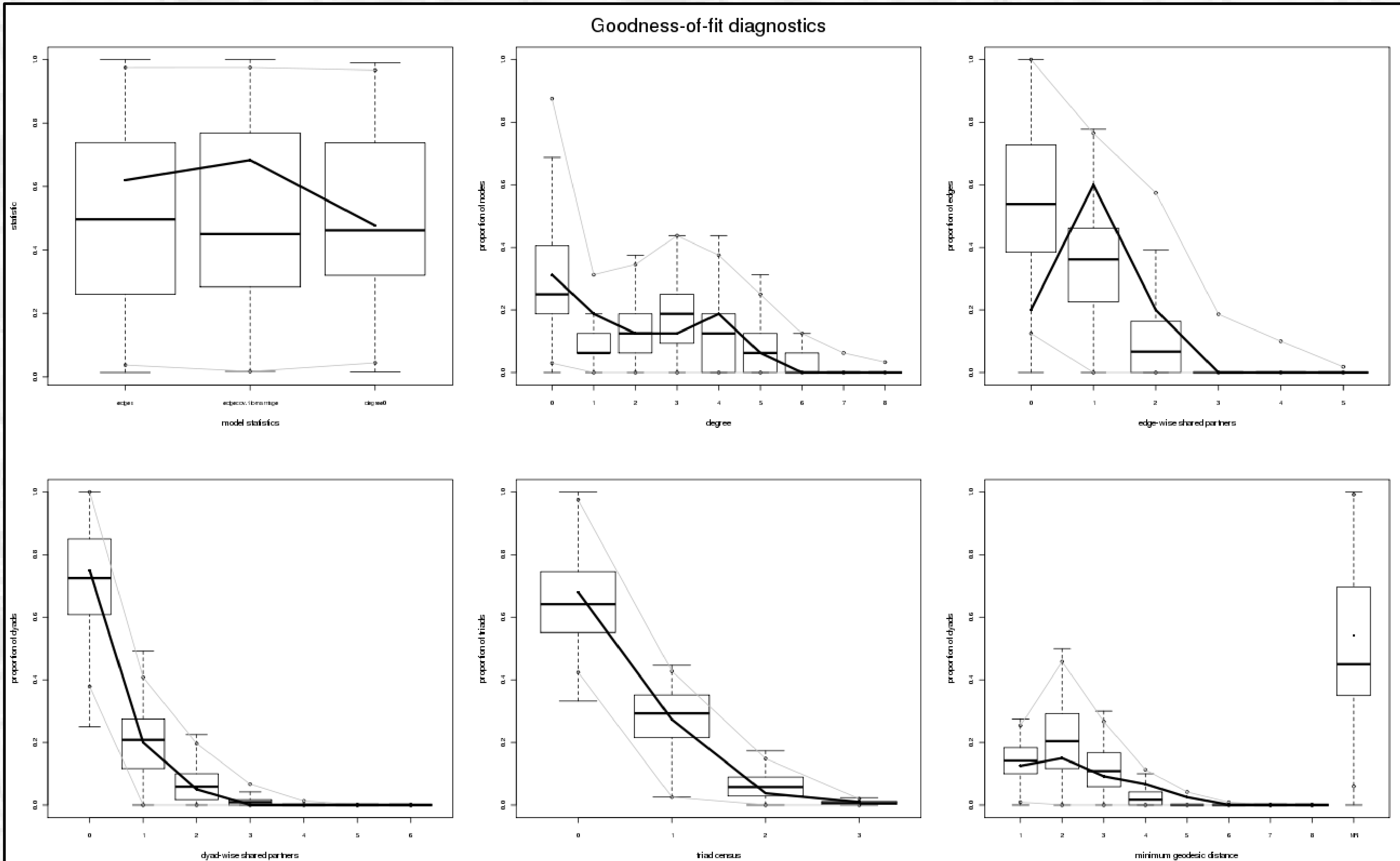
flobusiness w/edges, wealth



flobusiness w/edges, wealth



flobusiness w/edges, wealth, isolates (0-degree)



Final flobusiness Model

- **We now have a model in which we can be reasonably confident (MCMC diags also OK):**

```
==========================
Summary of model fit
==========================

Formula:   flobusiness ~ edges + edgecov(flomarriage) + degree(0)

Newton-Raphson iterations:  3
MCMC sample of size 10000

Monte Carlo MLE Results:
                    Estimate Std. Error MCMC s.e. p-value
edges                -1.7274     0.2887     0.023 < 1e-04 ***
edgecov.flomarriage   2.4852     0.4554     0.041 < 1e-04 ***
degree0               2.3828     0.7141     0.061 0.00114 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

    Null  Deviance: 166.355  on 120  degrees of freedom
 Residual Deviance:  64.377  on 117  degrees of freedom
          Deviance: 101.978  on   3  degrees of freedom

AIC: 70.377    BIC: 78.74
```