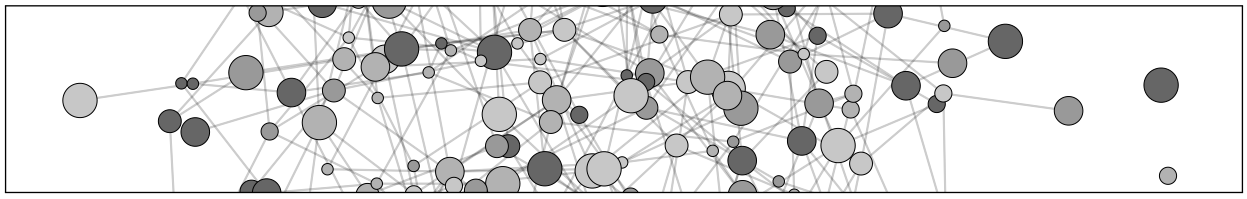
Moving Beyond Descriptives: An Introduction to Basic Network Statistics with `statnet`

Statnet Workshop Series · Sunbelt XXXIII · *21 May, 2013*



Workshop Presenters:

Lorien Jasny
Post-doctoral Researcher
Department of Environmental Science and Policy
University of California, Davis
ljasny@ucdavis.edu

Ryan Acton
Assistant Professor
Department of Sociology
University of Massachusetts Amherst
racton@soc.umass.edu

Contents

1 Getting Started

1.1 Warning

The document is best viewed on a Mac using Acrobat Reader!

1.2 Basic resources

- `R` website: `http://www.r-project.org`
- Helpful `R` tutorials: `http://cran.r-project.org/other-docs.html`
- `statnet` website: `http://statnet.org`
- `statnet` help: `statnet_help@statnet.org`

1.3 Typographical conventions

Text in a grey box, as below, represents code for you to type (or copy & paste into `R`).

```
gplot(g,gmode="graph",vertex.cex=1.5)
```

The lighter text inside the grey box represents comments. Comments are ignored by `R`, but are typically useful for humans.

All other text is meant to provide background or guidance.

1.4 Download and install the latest version of R

1. Go to http://cran.r-project.org/, and select Mirrors from the left-hand menu.
2. Select a location near you.
3. From the "Download and Install R " section, select the link for your operating system.
4. Follow the instructions on the relevant page.
5. Note that you need to download only the base distribution, not the contributed packages.
6. Once you've downloaded the installation file, follow the instructions for installation.

1.5 Download and attaching statnet and associated packages

1. Open R.
2. Install the statnet installer. At the R cursor, type:

```
install.packages("statnet")
```

3. Now, and in the future, you can install/update statnet at any point using the installer that comes with statnet. The previous step is only necessary the first time you wish to use statnet but does not need to be repeated each time. At the R cursor, type:

```
update.packages("statnet")
```

   Follow the directions; feel free to say no to any optional packages, although we recommend saying yes. The first choice provided is to install all the required and optional packages.

4. Attach statnet to your R session by typing:

```
library(statnet)
```

1.6 Set a specific working directory for this tutorial if you wish

1. If you are using Windows, go to File ¿ Change Dir and choose the directory.
2. On a Mac, go to Misc ¿ Change Working Directory and choose the directory.
3. Otherwise, use the setwd() command:

```
setwd("full.path.for.the.folder")
```

1.7 A few facts to remember about R

- R mostly runs through the command line or through batch files.
- Everything in R is an object, including data, output, functions—everything.
- Objects that are created during a session are saved in the "global environment" by default, which is stored as a single file (".RData ") in the working directory.
- R is case sensitive.
- R comes with a set of pre-loaded functions. Others can be added by downloading from the R project website. Downloaded packages to be used must be attached using the library() command during any R session in which they are to be used. For instance:

```
install.packages("coda")    # install
    package from CRAN
library(coda)      # attach installed
    package
```

2 Node level indices, network covariates, and network regression

2.1 Network visualization and basic descriptives

Make sure you're in the correct file directory, navigate to it using setwd() or insert the correct path into the following command.

```
load("basicNetworkStatistics.Rdata")
```

```
data(emon)        # Load Drabek et al. data
```

Begin by examining the 'emon' dataset

```
?emon
class(emon)
class(emon[[1]])
emon[[1]]
emon[[1]]%v%"vertex.names"        # Display vertex names
```

Extract ties from the Cheyenne EMON communicating at least "every few hours"

```
g<-as.sociomatrix(emon[[1]],"Frequency")         # Need to get the frequency info
g<-symmetrize((g>0)&(g<4))         # Note the reverse coding!
```

Get some potential covariates

```
drs<-emon[[1]]%v%"Decision.Rank.Score"         # Get decision rank (see man page)
crs<-emon[[1]]%v%"Command.Rank.Score"         # Get command rank
```

Plot Cheyenne EMON communications

```
gplot(emon[[1]])
gplot(emon[[1]], label.cex=0.5, label.col=4, label=network.vertex.names(emon[[1]]))        # Basic
    display, with labels
```

Calculate some basic centrality measures

```
deg<-degree(g,gmode="graph")
bet<-betweenness(g,gmode="graph")
clo<-closeness(g,gmode="graph")
```

Raw Correlations

```
cor(cbind(deg,bet,clo),cbind(drs,crs))
```

Classical tests (using asymptotic t distribution)

```
cor.test(deg,drs)
cor.test(bet,drs)
cor.test(clo,drs)
```

2.2 Testing Correlations

Let's build a permutation test

```
deg
drs
c.obs<-cor(deg,drs)
c.obs
```

Permute one of the data sets

```
sample(drs)
cor(deg,sample(drs))
```

write a for loop to permute one of the data sets many times

```
c.rep<-vector(length=1000)
for(niter in 1:1000){
  c.rep[niter]<-cor(deg,sample(drs))
}
```

```
c.rep        # look at what we've created
hist(c.rep)        # a histogram is a better display
abline(v=c.obs,col="red")        #compare to empirical data
```

Put it all together in a function!

```
perm.cor.test<-function(x,y,niter=1000)
{
  c.obs<-cor(x,y)
  c.rep<-vector()
  for(i in 1:niter)
    c.rep[i]<-cor(x,sample(y))
  c.rep
}
```

use it and look at the results

```
new.c.rep<-perm.cor.test(deg,drs)
hist(new.c.rep)
abline(v=c.obs,col="red")
```

calculate quantiles

```
mean(new.c.rep>=c.obs)
mean(new.c.rep<=c.obs)
mean(abs(new.c.rep)>=abs(c.obs))
```

make the output display these quantiles

```
perm.cor.test<-function(x,y,niter=1000){
  c.obs<-cor(x,y)
  c.rep<-vector()
  for(i in 1:niter)
    c.rep[i]<-cor(x,sample(y),use="complete.obs")
  cat("Vector Permutation Test:\n\tObserved correlation: ",c.obs,"\tReplicate quantiles
     (niter=",niter,")\n",sep="")
  cat("\t\tPr(rho>=obs):",mean(c.rep>=c.obs),"\n")
  cat("\t\tPr(rho<=obs):",mean(c.rep<=c.obs),"\n")
  cat("\t\tPr(|rho|>=|obs|):",mean(abs(c.rep)>=abs(c.obs)),"\n")
  invisible(list(obs=c.obs,rep=c.rep))
}
```

examine the fancy output

```
fancy<-perm.cor.test(deg,drs)
attributes(fancy)
fancy$obs
fancy$c.rep
```

For more information...

```
?cor.test
```

```
?t.test
```

```
?sample
```

2.3 Using NLIs as regression covariates

```
pstaff<-emon[[1]]%v%"Paid.Staff" # Get more EMON covariates
vstaff<-emon[[1]]%v%"Volunteer.Staff"
govt<-((emon[[1]]%v%"Sponsorship")!="Private")
```

Simple model: decision rank is linear in size, degree, and govt status

```
mod<-lm(drs~deg+pstaff+vstaff+govt)
summary(mod)
anova(mod) #Some useful lm tools
AIC(mod)
```

Try with alternative measures...

```
mod2<-lm(drs~bet+pstaff+vstaff+govt) #Betweenness
summary(mod2)
mod3<-lm(drs~clo+pstaff+vstaff+govt) #Closeness
summary(mod3)
AIC(mod,mod2,mod3) #Closeness wins!
```

For more information...

```
?lm
```

```
?anova
```

```
?AIC
```

2.4 Graph correlation and bivariate QAP

Remember the Florentine families data

```
data(florentine)
gplot(flobusiness) # Examine business ties
gplot(flomarriage) # Examine marriage ties
```

Could the two be related?

```
par(mfrow=c(1,2))
coords<-plot(flobusiness,label=flobusiness%v%"vertex.names",label.cex=.5,pad=1)
title("Business Ties")
plot(flomarriage,label=flomarriage%v%"vertex.names",label.cex=.5,pad=1,coord=coords)
title("Marriage Ties")
par(mfrow=c(1,1))
```

Let's try a graph correlation

```
gcor(flobusiness,flomarriage)
```

Why can't we use our previous permutation test? instead, use rmperm take a look

```
rmperm
```

```
par(mfrow=c(1,2))
plot(flobusiness,label=flobusiness%v%"vertex.names",label.cex=.5,pad=1,coord=coords)
title("Business Ties")
gplot(rmperm(flobusiness),label=flobusiness%v%"vertex.names",label.cex=.5,pad=1,coord=coords)
title("Permuted Business Ties")
par(mfrow=c(1,1))
```

now look at qaptest and use it

```
qaptest
qt<-qaptest(list(flobusiness,flomarriage),gcor,g1=1,g2=2)
summary(qt) # Examine the results
plot(qt) # Plot the QAP distribution
```

Testing against covariate effects

```
wealth<-sapply(flomarriage%v%"wealth",rep,network.size(flomarriage))
wealth
wealthdiff<-abs(outer(flomarriage%v%"wealth",flomarriage%v%"wealth","-"))
wealthdiff
qt1<-qaptest(list(flomarriage,wealth),gcor,g1=1,g2=2)
qt2<-qaptest(list(flomarriage,wealthdiff),gcor,g1=1,g2=2)
summary(qt1) # Do wealthy families have more ties?
summary(qt2) # Is there a wealth difference effect?
```

For more information...

```
?qaptest
?gcor
```

```
?outer
?sapply
```

```
?rep
```

2.5 Network Regression

Combine the previous tests (takes a while to perform QAP test)

```
marriagefit<-netlm(flomarriage,list(flobusiness,wealth,wealthdiff))
summary(marriagefit) # Examine the results
```

Another example: we begin by preparing the response variable. We will use the Cheyenne EMON in valued form, but need to recode the frequency data

```
Y<-as.sociomatrix(emon[[1]], "Frequency") # Extract frequencies
Y[Y>0]<-5-Y[Y>0] # Now, higher -> more frequent
```

Extract some vertex attributes

```
crk<-emon[[1]]%v% "Command.Rank.Score" # Command rank
spon<-emon[[1]]%v%"Sponsorship" # Type of organization
```

Form some predictor matrices (X variables)

```
Xcr<-sapply(crk,rep,length(crk)) # Column effects for command rank
Xcr
Xsp<-outer(spon,spon,"!=") # Dyadic effect for type difference
Xsp
```

Fit the model using QAP for standard errors (takes a while to perform QAP test)

```
cmfit<-netlm(Y,list(Xcr,Xsp))
summary(cmfit) # Examine the results
```

Fit the model using OLS for standard errors

```
cmfitB<-netlm(Y,list(Xcr,Xsp),nullhyp="classical")
summary(cmfitB) # Examine the results
```

For more information...

```
?outer
```

```
?netlm
```

3 Graph Level Indices and Conditional Uniform Graph Tests

3.1 Permutation tests for GLI/graph-level covariate association

Here we consider the famous Sampson monastery data:

```
par(mfrow=c(4,3),mar=c(2,1,4,1))
for(i in 1:length(sampson))
  gplot(sampson[[i]],displaylabel=TRUE,boxed.label=FALSE,main=names(sampson)[i])
par(mfrow=c(1,1),mar=c(5,4,4,2)+.01)
```

Are positive relations more reciprocal (relative to density) than negative ones? Let's try a simple permutation test:

```
r4<-grecip(sampson,measure="edgewise.lrr")        # Calculate reciprocity levels for each network
r4
```

Categorize relationships by positive and negative

```
ispos<-c(TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
```

```
obs<-sum(r4[ispos])-sum(r4[!ispos])         # Calculate the empirical relationship
obs         # But what does this mean?
```

```

reps<-vector()        # Run a permutation test
for(i in 1:1e4){
  temp<-sample(ispos)
  reps[i]<-sum(r4[temp])-sum(r4[!temp])
}
```

calculate statistics and plot

```
mean(reps>=obs)         #Upper tail p-value
mean(abs(reps)>=abs(obs))         # Two-sided version
hist(reps)
abline(v=obs,col=2,lwd=3)         # Visualize it
```

We can look at transitivity as well. How does transitivity compare to density? (Log-odds method)

```
log(gtrans(sampson)/gden(sampson))
```

Are positive relations more transitive (relative to density) than negative ones? Let's try another vector permutation test:

```
ltr<-log(gtrans(sampson)/gden(sampson))
obs<-sum(ltr[ispos])-sum(ltr[!ispos])
reps<-vector()
for(i in 1:1e4){
  temp<-sample(ispos)
  reps[i]<-sum(ltr[temp])-sum(ltr[!temp])
}
mean(reps>=obs) # Upper tail p-value
mean(abs(reps)>=abs(obs)) # Two-sided version
hist(reps)
abline(v=obs,col=2,lwd=3) # Visualize it
```

3.2 Comparing graphs via the triad census

Let's get the triad census for each network

```
tc<-triad.census(sampson)
tc
```

Cool trick: two-way correspondence analysis of graphs and their triad census scores (aka a "Faust diagram"). Networks here appear close to the triad types they contain at excess frequency (distances are chi-squared based; see references in ?corresp for more detail).

```
library(MASS) # Requires the MASS package
plot(corresp(tc,nf=2)) # Plot network/triad association
```

What if this data were symmetric? We can symmetrize to illustrate.

```
tc<-triad.census(symmetrize(sampson),mode="graph") #Need to use mode="graph" here
rownames(tc)<-names(sampson)
plot(corresp(tc,nf=2)) # Plot undirected network/triad association
```

For more information...

```
?gtrans
?triad.census
```

```
?corresp
?symmetrize
```

3.3 Simple univariate conditional uniform graph tests

The cug.test function provides a simple interface for univariate CUG tests. Let's try testing some data on trade in complex manufactured goods to see if overall activity (density) is greater then would be expected from size alone.

```
cug.test(ctrade,gden) # Call cug.test with the gden (density) function
```

Is there more reciprocity than density would suggest? Let's see.

```
cug.test(ctrade,grecip,cmode="edges") # Conditioning on edges, calling grecip
```

Given biases in density and reciprocity, we might look to see if there is a bias in transitivity, too. This time, let's condition on all of the above.

```
cug.test(ctrade,gtrans,cmode="dyad") # Conditioning on dyad census
```

We are not limited to simple commands. Note that we here must pass not only arguments to cug.test, but also to centralization and degree! Let's try indegree centralization:

```
ct<-cug.test(ctrade,centralization,cmode="dyad",FUN.arg=list(FUN=degree, cmode="indegree"))
```

```
ct # Print the result
plot(ct) # Can also plot it!
```

4 Multivariate Analysis of Graph Sets

4.1 Distance based methods: clustering and scaling

Start by calculating Hamming distances for the Sampson data

```
samphd<-hdist(sampson)
samphd
```

Now, try an MDS solution

```
sampmds<-cmdscale(samphd)
sampmds
plot(sampmds,type="n") # Plot the results
text(sampmds,label=names(sampson))
```

MDS suggests a three-cluster solution; let's try hclust

```
samphc<-hclust(as.dist(samphd))
plot(samphc,labels=names(sampson)) # Very clear solution
rect.hclust(samphc,k=3)
```

Examine central graphs for each cluster

```
sampcg<-gclust.centralgraph(samphc,k=3,sampson)
par(mfrow=c(2,2))
gplot(sampcg[1,,],main="Positive CG")
gplot(sampcg[2,,],main="Negative CG")
gplot(sampcg[3,,],main="Liking CG")
par(mfrow=c(1,1))
```

More fun - can plot stats by cluster!

```
gclust.boxstats(samphc,k=3,grecip(sampson,measure="edgewise.lrr"),
    names=c("Positive","Negative","Liking"), main="Edgewise LRR Reciprocity by Relational Type")
gclust.boxstats(samphc,k=3,gtrans(sampson), names=c("Positive","Negative","Liking"),
    main="Transitivity by Relational Type")
```

For more information...

```
?hdist
?cmdscale
```

```
?hclust
?rect.hclust
```

```
?gclust.centralgraph
?gdist.plotstats
```

4.2 Network PCA

To begin, let's get the graph correlation matrix for the Sampson nets

```
sampcor<-gcor(sampson)
sampcor
```

Now, we compute the eigendecomposition (could also have used gcov above)

```
sampeig<-eigen(sampcor)
```

Eigenvalues contain variance explained, to whit:

```
evals<-sampeig$value # Extract eigenvalues
evals/sum(evals) # Variance explained
```

Show this as a scree plot

```
barplot(evals/sum(evals),names.arg=1:length(evals))
```

Examine loadings (eigenvectors); looks like first 3 are key

```
load<-sampeig$vector[,1:3]
rownames(load)<-names(sampson)
load
```

Can rotate using varimax

```
varimax(load)
```

Try plotting the first two components

```
plot(load[,1:2],type="n",asp=1,xlab="PC 1",ylab="PC 2")
abline(h=0,v=0,lty=3)
arrows(0,0,load[,1],load[,2],col=2) # Should be read angularly!
text(load[,1:2],label=names(sampson))
```

Finally, extract scores

```
S1<-apply(sweep(as.sociomatrix.sna(sampson),1,load[,1],"*"),c(2,3),sum)
S2<-apply(sweep(as.sociomatrix.sna(sampson),1,load[,2],"*"),c(2,3),sum)
S3<-apply(sweep(as.sociomatrix.sna(sampson),1,load[,3],"*"),c(2,3),sum)
```

Visualize a score graph (not too helpful in this case!)

```
coord<-gplot.layout.fruchtermanreingold(as.edgelist.sna(S1>0),NULL)
gplot(S1!=0,edge.col=sign(S1)+3,coord=coord)
```

For more information...

```
?gcor
?gcov
?eigen
```

```
?varimax
?abline
?arrows
```

```
?sweep
?gplot.layout
```

4.3 Network CCA

For this one, we're going to use the country trade data. Somewhat perversely, we're going to use yacca instead of netcancor (b/c this data set has missing data which netcancor doesn't handle, and b/c yacca currently has better visualizations).

```
install.packages("yacca")
library(yacca)
```

Perform a canonical correlation analysis between relations and attribute differences

```
trade.cca<-cca(gvectorize(trade),gvectorize(tradediff), standardize.scores=FALSE) # Turn off
    standardization b/c of NAs
summary(trade.cca)
```

```
plot(trade.cca) # Visualize the output
```

For more information...

```
?netcancor
?yacca
```

```
?cca
?gvectorize
```

4.4 Studying Qualitative Dynamics with Network MDS

Johnson collected 'liking' scores for researchers working in the South Pole on a monthly basis over two years

```
class(johnsonPolarY1)
dim(johnsonPolarY1)
johnsonPolarY1[1,,]
```

Visualize the data using weight or color to aid interpretation

```
gplot(johnsonPolarY1)
gplot(johnsonPolarY1[1,,],edge.lwd=johnsonPolarY1[1,,])
coords<-gplot(johnsonPolarY1[1,,],edge.col=rainbow(10)[johnsonPolarY1[1,,]], vertex.col="black")
par(ask=TRUE)
for(i in 1:8)
  gplot(johnsonPolarY1[i,,],edge.col=rainbow(10)[johnsonPolarY1[i,,]],
        vertex.col="black",coord=coords)
```

Calculate Hamming distances for each of the two Johnson data sets

```
jp1d <- hdist(johnsonPolarY1)
jp2d <- hdist(johnsonPolarY2)
```

Now try an MDS solution

```
jp1mds<-cmdscale(jp1d)
jp1mds
plot(jp1mds,type="l",lty=2) # Plot the results
text(jp1mds,label=rownames(johnsonPolarY1),font=2)
jp2mds<-cmdscale(jp2d)
jp2mds
plot(jp2mds,type="l",lty=2) # Plot the results
text(jp2mds,label=rownames(johnsonPolarY2),font=2)
```

5 Credits

5.1 The `statnet` core development team

- Carter T. Butts (UC Irvine)
- Steven M. Goodreau (University of Washington)
- Mark S. Handcock (UCLA)
- David R. Hunter (Penn State University)
- Martina Morris (University of Washington)

5.2 Document authorship credits

- Section 1
  - Steven M. Goodreau
  - Ryan M. Acton
- Section 2 to Section 4
  - Carter T. Butts
  - Lorien Jasny

5.3 Document LaTeX source code

- Ryan M. Acton