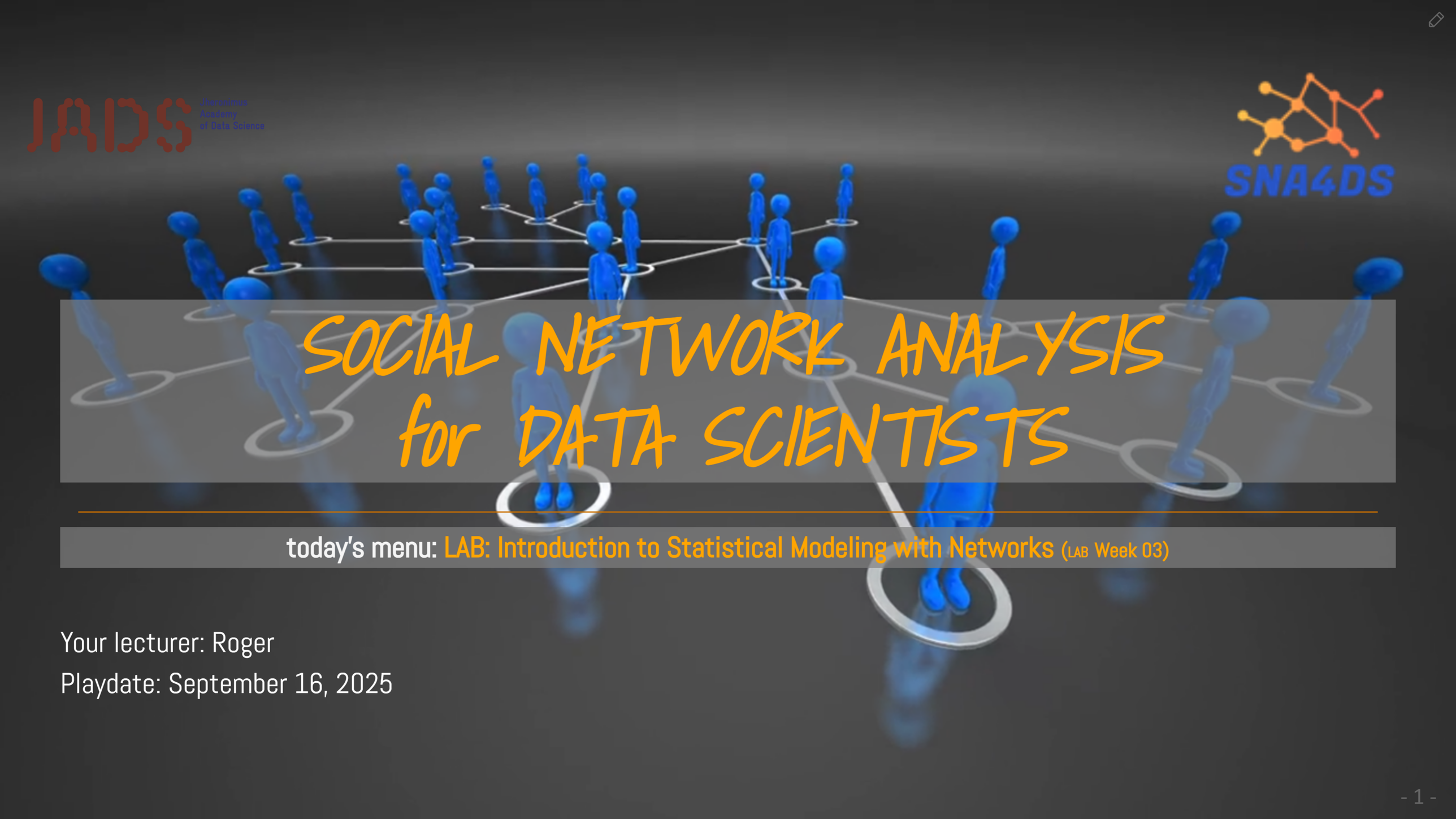


SOCIAL NETWORK ANALYSIS for DATA SCIENTISTS

today's menu: LAB: Introduction to Statistical Modeling with Networks (LAB Week 03)

Your lecturer: Roger

Playdate: September 16, 2025

Today's menu

1. Homeplay
2. QAP findings
3. CUG findings and extension
4. Assessment and practical overview of the methods

How did the tutorial and homeplay go?

All clear?

Challenges?

Useful function to extract vertex attributes

```
attrs <- snafun::extract_all_vertex_attributes(louis)
attrs
```

	black_political_equality	democratic	Intercept	perc_black	perc_cath	perc_urban	vertex.names
1	80.7	75.8	1	19.7	83.0	55.8	1
2	79.0	59.8	1	24.8	45.4	33.3	2
3	68.6	74.8	1	31.9	88.2	33.4	3
4	68.2	71.7	1	41.2	95.9	0.0	4
5	42.8	76.0	1	27.8	87.1	25.1	5
6	55.3	48.7	1	22.4	11.0	37.5	6
7	1.0	17.6	1	49.4	0.2	15.2	7
8	16.5	25.2	1	24.9	9.0	66.0	8
9	20.5	24.8	1	36.5	10.2	80.8	9
10	69.4	64.4	1	20.9	52.6	73.9	10
11	7.9	34.0	1	27.8	1.3	0.0	11
12	75.0	83.1	1	6.4	78.8	0.0	12
13	23.9	26.2	1	35.2	0.0	0.0	13
14	1.0	12.7	1	50.3	1.5	39.7	14
15	14.5	23.1	1	46.3	10.4	43.4	15
16	16.2	26.7	1	57.5	10.4	24.1	16
17	43.4	46.6	1	31.8	36.6	85.1	17
18	0.0	23.9	1	61.2	27.7	40.1	18

R code NAM NAM Output

```
adj <- snafun::to_matrix(louis)
w_adj <- adj/rowSums(adj)

mod <- snafun::stat_nam(democratic ~ perc_black +
                          perc_cath + perc_urban + black_political_equality,
                          data = attrs, W = w_adj)
```

```
Call:spatialreg::lagsarlm(formula = formula, data = data, listw = W,
    na.action = na.action, Durbin = Durbin, quiet = quiet, zero.policy = zero.policy)

Residuals:
      Min        1Q    Median        3Q       Max
-22.68996  -4.80434   0.58857   4.05038  17.69156

Type: lag
Coefficients: (asymptotic standard errors)
                          Estimate Std. Error z value  Pr(>|z|)
(Intercept)             13.8735681  4.6729365  2.9689  0.002988
perc_black              -0.0040543  0.0718224 -0.0564  0.954984
perc_cath                0.2255374  0.0464735  4.8530 1.216e-06
perc_urban              -0.0979634  0.0365971 -2.6768  0.007433
black_political_equality  0.2957095  0.0589806  5.0137 5.340e-07

Rho: 0.30601, LR test value: 7.8801, p-value: 0.0049982
Asymptotic standard error: 0.096259
    z-value: 3.179, p-value: 0.0014778
Wald statistic: 10.106, p-value: 0.0014778

Log likelihood: -216.5846 for lag model
ML residual variance (sigma squared): 49.81, (sigma: 7.0576)
Number of observations: 64
```

Today's menu

Homeplay

What to
use when?

General approach

When	Which approach	function
Statistic on a single network	Conditionally Uniform Graph	sna::cug.test
Association between two networks	QAP	sna::qaptest
A valued dependent network and one or more explanatory networks	QAP linear model	sna::netlm
A binary dependent network and one or more explanatory networks	QAP logistic model	sna::netlogit

Transitivity: conditioning on edges

R code | Output | Output plot

```
trans_f <- function(x, directed = FALSE) {
  x <- snafun::fix_cug_input(x, directed = directed)
  snafun::g_transitivity(x)
}

cug_trans_15 <- sna::cug.test(fifa2015, mode = "graph",
                              FUN = trans_f,
                              cmode = "edges", reps = 1000)
cug_trans_15
```

OR

```
cug_trans_15 <- sna::cug.test(fifa2015,
                              FUN = sna::gtrans,
                              cmode = "edges",
                              reps = 1000,
                              FUN.args = list(mode = "graph"))
cug_trans_15
```

Transitivity: conditioning on edges

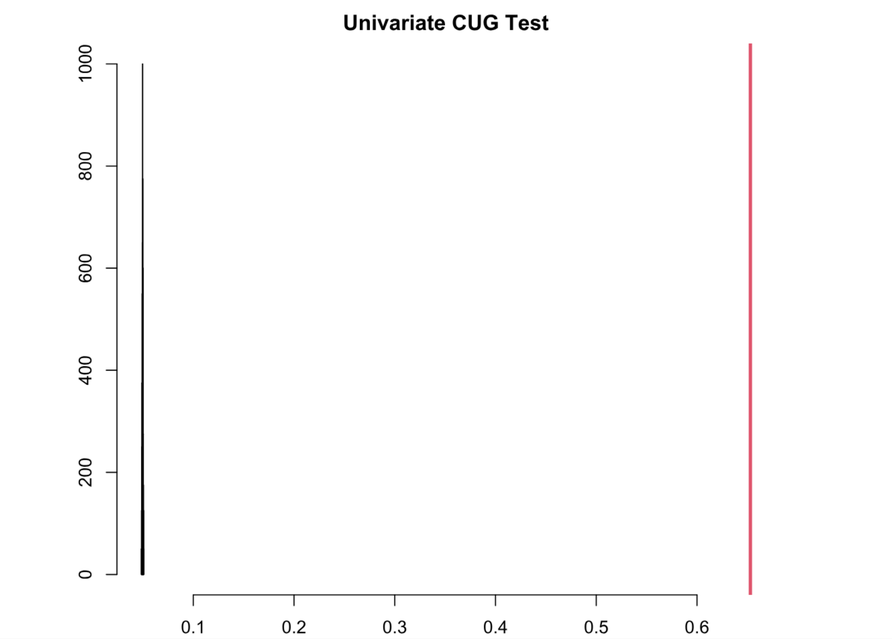
R code | Output | Output plot

```
Univariate Conditional Uniform Graph Test

Conditioning Method: edges
Graph Type: digraph
Diagonal Used: FALSE
Replications: 200

Observed Value: 0.6529858
Pr(X>=Obs): 0
Pr(X<=Obs): 1
```

Transitivity: conditioning on edges

R code | Output | Output plot

```
plot(cug_trans_15)
```



Exceptionally strong friend-of-a-friend tendencies!

Betweenness centralization

R code | Output | Plot output

```
### Conditioning on the dyad census

betw_f <- function(x, directed = FALSE) {  # note: directed = FALSE!
  x <- snafun::fix_cug_input(x, directed = directed)
  snafun::g_centralize(x, measure = "betweenness", directed = directed)$centralization
}

cug_betw_06_ds <- sna::cug.test(fifa2006,
                                mode = "graph", FUN = betw_f,
                                cmode = "dyad.census", reps = 200)
cug_betw_06_ds
```

OR

```
cug_betw_06_ds <- sna::cug.test(fifa2006, FUN = sna::centralization,
                                          FUN.arg=list(FUN = sna::betweenness),
                                          mode="graph",
                                          reps = 200,
                                          cmode="dyad.census")
```

Betweenness centralization

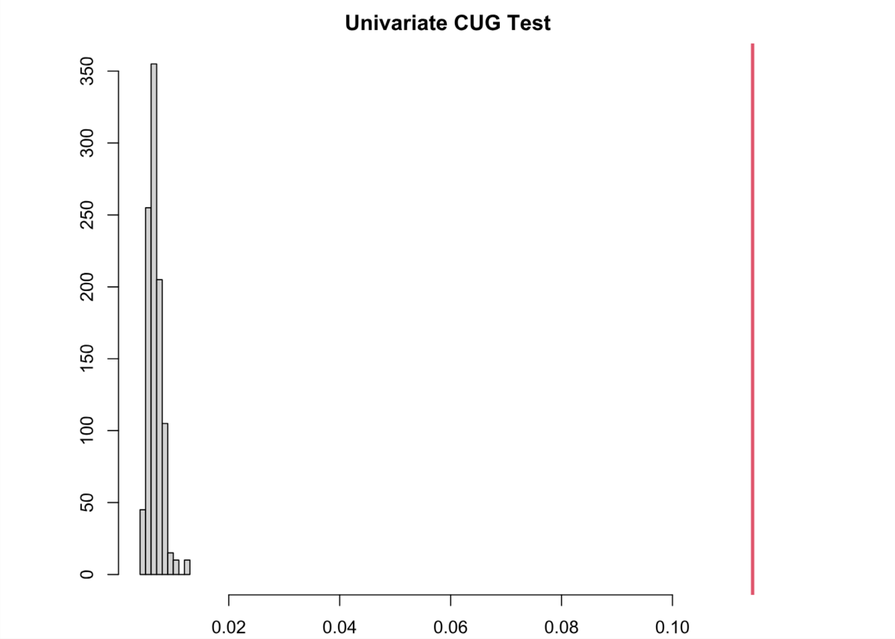
R code | Output | Plot output

```
Univariate Conditional Uniform Graph Test

Conditioning Method: dyad.census
Graph Type: graph
Diagonal Used: FALSE
Replications: 200

Observed Value: 0.1144484
Pr(X>=Obs): 0
Pr(X<=Obs): 1
```

Betweenness centralization

R code | Output | Plot output

```
plot(cug_betw_06_ds)
```



We see exceptionally strong betweenness centralization in FIFA.

What does this imply about how FIFA operates?

Advanced CUG use

Use your own measure for sna::cug.test

Intuition | Challenge | R code | Output | Plot output

You can define your own measures to use inside `sna::cug.test`, so you are not confined to only the measures implemented in `sna`. For example, you can test if the number of walktrap communities in the network is statistically significant.

Advanced CUG use

Use your own measure for sna::cug.test

Intuition | Challenge | R code | Output | Plot output

Test if the number of walktrap communities in the network is statistically significant for the `fifa2015` network.

You can use previous code as a template:

```
trans_f <- function(x, directed = FALSE) {
  x <- snafun::fix_cug_input(x, directed = directed)
  snafun::g_transitivity(x)
}

sna::cug.test(fifa2015, mode = "graph", FUN = trans_f, cmode = "edges", reps = 1000)
```

06:00

Advanced CUG use

Use your own measure for sna::cug.test

Intuition | Challenge | R code | Output | Plot output

```
walktrap_num_f <- function(x, directed = FALSE) {  # note: directed = FALSE!
  x <- snafun::fix_cug_input(x, directed = directed)
  snafun::extract_comm_walktrap(x) |> length()
}

fifa_coms <- sna::cug.test(fifa2015, FUN = walktrap_num_f, mode = "graph",
                           diag = FALSE, cmode = "edges", reps = 1000)

fifa_coms
plot(fifa_coms)
```

Advanced CUG use

Use your own measure for sna::cug.test

Intuition | Challenge | R code | Output | Plot output

```
print(fifa_coms)
```

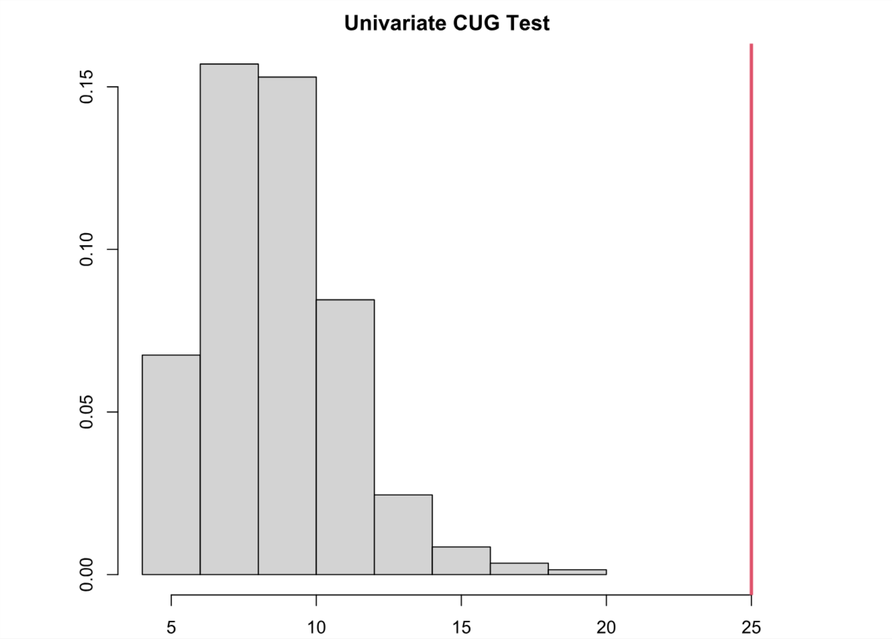
```
Univariate Conditional Uniform Graph Test

Conditioning Method: edges
Graph Type: graph
Diagonal Used: FALSE
Replications: 1000

Observed Value: 25
Pr(X>=Obs): 0
Pr(X<=Obs): 1
```

➔ There are more information-trapping communities (statistically significant) than you would expect of a network of this size and density.

Advanced CUG use

Use your own measure for sna::cug.test

Intuition | Challenge | R code | Output | Plot output



Statistically modeling networks

Testing hypotheses

- use the simplest statistical model that suits your hypothesis test
  - QAP, MRQAP, Logistic QAP, and CUG are often great for simple hypotheses
  - use ERGM's to test more complex expectations

Estimating parameters

- MRQAP, Logistic QAP, and ERGM's allow for parameter estimation

Uncovering what the drivers are of the observed network

- ERGM's model the network as a whole and allow for much more modeling flexibility and options
  - this comes at the challenge of fitting a more complex model