Simple ERGM

Charles Costanzo

2023-10-05

```
# install "intergraph" package (only need to do once, so comment out)
# install.packages("intergraph")

# create a vector containing names of packages we want to load
packages <- c("intergraph","igraph","statnet","ergm")

# load in packages
lapply(packages, library, character.only = TRUE)
```

```
# read in igraph object "g"
path <- "/Users/charlescostanzo/College/Au 2023/Politsc 4998/Data/graphml_zscore_self_1_1_5_85_164/114.g
g <- read_graph(path, format = "graphml")

# print vertex attribute names
names(vertex_attr(g))
```

```
##  [1] "nokken_poole_dim2"            "nokken_poole_dim1"
##  [3] "conditional"                  "nominate_number_of_errors"
##  [5] "nominate_number_of_votes"     "nominate_geo_mean_probability"
##  [7] "nominate_log_likelihood"      "nominate_dim2"
##  [9] "nominate_dim1"                "died"
## [11] "born"                         "bioguide_id"
## [13] "bioname"                      "last_means"
## [15] "occupancy"                    "party_code"
## [17] "state_abbrev"                 "district_code"
## [19] "state_icpsr"                  "icpsr"
## [21] "chamber"                      "congress"
## [23] "id"
```

```
# set born attribute to numeric type
V(g)$born <- as.numeric(V(g)$born)
V(g)$died <- ifelse(V(g)$died == "NaN", NA, as.numeric(V(g)$died))

# convert igraph object "g" to network named "net"
net <- asNetwork(g)

# extract edgelist from "g"
el.g <- get.edgelist(g)

# extract edgelist from "net"
el.net <- as.matrix(net, "edgelist")
```

```r
# check if edge lists are identical
identical(as.numeric(el.g), as.numeric(el.net))
```

```
## [1] TRUE
```

```r
# set random seed for reproducibility
set.seed(1022)

# run very simple ergm
model <- ergm(net ~ edges +
                ostar(2:3) +
                nodemix("party_code"),
              control = control.ergm(MCMC.samplesize=5000,
                                     MCMC.burnin=5000,
                                     MCMLE.maxit=10)
)
```

```
## 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 10:

## 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 1.0000.

## The log-likelihood improved by 0.0262.

## Convergence test p-value: 0.1779. Not converged with 99% confidence; increasing sample size.
## Iteration 2 of at most 10:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0963.
## Convergence test p-value: 0.1492. Not converged with 99% confidence; increasing sample size.
## Iteration 3 of at most 10:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0622.
## Convergence test p-value: 0.1028. Not converged with 99% confidence; increasing sample size.
## Iteration 4 of at most 10:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0383.
## Convergence test p-value: 0.0099. 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.
```

```
# display model summary
summary(model)
```

```
## Call:
## ergm(formula = net ~ edges + ostar(2:3) + nodemix("party_code"),
##     control = control.ergm(MCMC.samplesize = 5000, MCMC.burnin = 5000,
##         MCMLE.maxit = 10))
##
## Monte Carlo Maximum Likelihood Results:
##
##                          Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                   -1.670e+00  6.359e-01      0  -2.626  0.00863 **
## ostar2                  -2.873e-02  5.099e-02      0  -0.563  0.57317
## ostar3                   8.308e-05  2.098e-03      0   0.040  0.96841
## mix.party_code.200.100 -5.637e-01  3.289e-02      0 -17.140  < 1e-04 ***
## mix.party_code.100.200 -9.724e-01  3.286e-02      0 -29.587  < 1e-04 ***
## mix.party_code.200.200 -3.514e-01  3.039e-02      0 -11.564  < 1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 261719  on 188790  degrees of freedom
##  Residual Deviance:  81854  on 188784  degrees of freedom
##
## AIC: 81866  BIC: 81927  (Smaller is better. MC Std. Err. = 1.138)
```