Package 'xergm.common'

April 7, 2020

Version 1.7.8

Date 2020-04-06

Title Common Infrastructure for Extensions of Exponential Random Graph Models

Description Datasets and definitions of generic functions used in dependencies of the 'xergm' package.

URL http://github.com/leifeld/xergm.common

Depends R (>= 2.14.0), ergm (>= 3.5.1), network (>= 1.13.0)

Imports methods, utils, stats

License GPL (>= 2)

NeedsCompilation no

Author Philip Leifeld [aut, cre],
Skyler J. Cranmer [ctb],
Bruce A. Desmarais [ctb]

Maintainer Philip Leifeld <philip.leifeld@essex.ac.uk>

Repository CRAN

Date/Publication 2020-04-07 09:50:02 UTC

R topics documented:

adjust *Adjust the dimensions of a matrix to the dimensions of another matrix*

Description

Adjust the dimensions of a matrix to the dimensions of another matrix.

Usage

```
adjust(source, target, remove = TRUE, add = TRUE, value = NA,
    returnlabels = FALSE)
```

Arguments

source	A matrix, network, list or data.frame object or a vector which should be adjusted.
target	A matrix, network, list or data.frame object or a vector to which the source object is compared with regard to its labels.
remove	Should rows and columns that are not present in the target object be removed?
add	Should rows and columns that are present in the target object but not in the source object be added to the source object?
value	The value to be inserted if a new row or column is added. By default, new cells are filled with NA values, but other sensible values may include -Inf or 0.
returnlabels	Return a list of added and removed row and column labels rather than the actual matrix, vector, or network object?

Details

An adjacency matrix (the source matrix) is compared to another adjacency matrix (the target matrix) by matching the row or column labels. If the target matrix contains rows/columns which are not present in the source matrix, new rows and columns with the corresponding labels and NA values in the cells are inserted into the source matrix. If the source matrix contains rows/columns which are not present in the target matrix, these rows and columns are removed from the source matrix. In addition to adjacency matrices, two-mode matrices, network objects (also with vertex attributes), and vectors are supported.

See Also

handleMissings

Examples

```
# create sociomatrix a with 13 vertices a to m
vertices <- letters[1:13]
a <- matrix(rbinom(length(vertices)^2, 1, 0.1), nrow = length(vertices))
rownames(a) <- colnames(a) <- vertices
```

```
# create sociomatrix b with the same vertices except f and k, but additional n
vertices <- c(vertices[-c(6, 11)], "n")
b <- matrix(rbinom(length(vertices)^2, 1, 0.1), nrow = length(vertices))
rownames(b) <- colnames(b) <- vertices

# check dimensions
dim(a)  # 13 x 13
dim(b)  # 12 x 12

# adjust a to b: add n and fill up with NAs; remove f and k
adjust(a, b, add = TRUE, remove = TRUE)

# more complex example with additional attributes stored in the network object;
# convert a to network objects with additional vertex and network attributes
nw <- network(a)
vertices <- letters[1:13]
nwattrib1 <- matrix(rbinom(length(vertices)^2, 1, 0.1), nrow = length(vertices))
nwattrib2 <- nwattrib1
rownames(nwattrib1) <- colnames(nwattrib1) <- vertices
set.network.attribute(nw, "nwattrib1", nwattrib1)
set.network.attribute(nw, "nwattrib2", nwattrib2)
set.vertex.attribute(nw, "vattrib", 1:length(vertices))

# check presence of the two attributes
list.network.attributes(nw)  # nwattrib1 and nwattrib2 are listed
get.network.attribute(nw, "nwattrib1")  # returns sociomatrix with labels
get.network.attribute(nw, "nwattrib2")  # returns sociomatrix without labels
list.vertex.attributes(nw)  # vattrib is listed
get.vertex.attribute(nw, "vattrib")  # returns numeric vector 1:13

# adjust the network including the two attributes
nw.adjusted <- adjust(nw, b, add = TRUE, remove = TRUE)
as.matrix(nw.adjusted)  # note that the order of nodes may have changed
get.network.attribute(nw.adjusted, "nwattrib1")  # returns adjusted sociomatrix
get.network.attribute(nw.adjusted, "nwattrib2")  # returns adjusted sociomatrix
get.vertex.attribute(nw.adjusted, "vattrib")  # returns adjusted vector
```

`alliances` *Longitudinal international defense alliance network, 1981–2000*

Description

The alliances dataset contains the international defense alliance network among 164 countries, covering the years 1981–2000. In addition to the yearly defense alliance network, it contains data on military capabilities, governing regime type, geographic contiguity and international conflict. This is an excerpt from a dataset that has been used in two published analyses. The full dataset (Cranmer, Desmarais and Menninga 2012; Cranmer, Desmarais and Kirlkand 2012) contains a large number of countries and a much longer time series.

Usage

```
data("alliances")
```

Format

`allyNet` is a list of network objects at 20 time points, 1981–2000, containing undirected defense alliance networks. In addition to the alliance ties, each network object contains three vertex attributes. `cinc` is the "CINC" or Composite Index of National Capability score (see https://correlatesofwar.org/data-sets/national-material-capabilities). `polity` is the "polity score" of each country in the respective year. Quoting the online description, "the Polity Score captures this regime authority spectrum on a 21-point scale ranging from -10 (hereditary monarchy) to +10 (consolidated democracy)," (see http://www.systemicpeace.org/polityproject.html). `year` is simply the year recorded as a vertex attribute.

`contigMat` is a 164 x 164 binary matrix in which a 1 indicates that two countries share a border.

`lNet` is a list of 20 matrices. Each element is the adjacency matrix from the previous year. This is used to model memory in the ties.

`LSP` is a list of 20 matrices. Each element is a matrix recording the number of shared partners between countries in the alliance network from the previous year.

`warNet` is a list of 20 matrices. Each element is a binary matrix that indicates whether two states were in a militarized interstate dispute in the respective year.

Source

The data were gathered by Skyler Cranmer and Bruce Desmarais in the process of writing Cranmer, Desmarais and Menninga (2012) and Cranmer, Desmarais and Kirlkand (2012).

Permission to redistribute this dataset along with this package was granted by Skyler Cranmer and Bruce Desmarais on December 15, 2015. Questions about the data should be directed to them.

References

Skyler J. Cranmer, Bruce A. Desmarais, and Justin H. Kirkland (2012): Toward a Network Theory of Alliance Formation. *International Ineractions* 38(3): 295–324.

Skyler J. Cranmer, Bruce A. Desmarais, and Elizabeth Menninga (2012): Complex Dependencies in the Alliance Network. *International Ineractions* 29(3):279–313.

Examples

```
## Not run:
library("xergm")
data("alliances")

# btergm formulas look very similar to ERGM formulas.
# Note the R argument; usually want R > 1000.
# Here it is set to 50 to limit computation time.
# First, set the seed for replicability.
set.seed(123)
model <- btergm(allyNet ~ edges + gwesp(0, fixed = TRUE)
    + edgecov(lNet) + edgecov(LSP) + edgecov(warNet)
```

```
    + nodecov("polity") + nodecov("cinc") + absdiff("polity")
    + absdiff("cinc") + edgecov(contigMat) + nodecov("year"),
    R = 50)

# View estimates and confidence intervals.
summary(model)

# Evaluate model fit. Simulate 100 networks for each time point.
# Calculate edgewise shared partners, degree and geodesic distance
# distance distributions.
alliance_gof <- gof(model, statistics = c(deg, esp, geodesic))

# Plot goodness of fit.
plot(alliance_gof)

## End(Not run)
```

`checkdegeneracy` *Generic function for checking whether a statistical network model is degenerate*

Description

Generic function for checking whether a statistical network model is degenerate.

Usage

```
checkdegeneracy(object, ...)
```

Arguments

`object`	An `mtergm` or `btergm` object.
`...`	Custom arguments to be handed over to subroutines.

Details

The `checkdegeneracy` function permits degeneracy checking for statistical network models (mtergm and btergm objects). See the specific methods in the **btergm** package.

chemnet *German Toxic Chemicals Policy Network in the 1980s (Volker Schneider)*

Description

The chemnet dataset contains network and attribute data and for the 30 most influential political actors with regard to toxic chemicals regulation in Germany in 1983/1984. While the original dataset contains up to 47 actors, this dataset contains the "complete influence core" of mutually relevant actors. The data are cross-sectional. There are no missing data; the response rate was 100 percent. Volker Schneider (University of Konstanz) collected this dataset for his dissertation (Schneider 1988). The dataset was later re-used for a journal publication on information exchange in policy networks (Leifeld and Schneider 2012).

The chemnet dataset contains network relations on political/strategic and technical/scientific information exchange, influence attribution, and membership in policy committees/forums, as well as nodal attributes on the actor type and opinions about the six most salient issues related to the political process that was leading to a new chemicals law at the time being.

Usage

```
data(chemnet)
```

Format

`pol` is a directed 30 x 30 adjancency matrix indicating which row actor sends political/strategic information to which column actor. `1` indicates an information exchange tie, and `0` indicates the absence of a network tie.

`scito` is a directed 30 x 30 adjacency matrix indicating which row actor sends technical/scientific information to which column actor. `1` indicates an information exchange tie, and `0` indicates the absence of a network tie. In contrast to political/strategic information exchange, two separate survey questions were asked about technical/scientific information exchange: sending information, and receiving information. The two matrices contain the same relation but one time from the sender's perspective and one time from the receiver's perspective. By combining the two matrices, one can create a "confirmed" technical/scientific information exchange relation. The `scito` matrix contains ties from the sender's perspective.

`scifrom` is a directed 30 x 30 adjacency matrix indicating which row actor receives technical/scientific information from which column actor. `1` indicates an information exchange tie, and `0` indicates the absence of a network tie. In contrast to political/strategic information exchange, two separate survey questions were asked about technical/scientific information exchange: sending information, and receiving information. The two matrices contain the same relation but one time from the sender's perspective and one time from the receiver's perspective. By combining the two matrices, one can create a "confirmed" technical/scientific information exchange relation. The `scifrom` matrix contains ties from the receiver's perspective.

`infrep` is a directed 30 x 30 adjancency matrix indicating which row actor deems which column actor "particularly influential". `1` indicates such a tie, and `0` indicates the absence of an influence attribution tie.

`committee` is a 30 x 20 two-mode (bipartite) network matrix indicating which row actor is a member of which policy committee/forum (as indicated by the column labels). `1` indicates a membership tie, and `0` indicates non-membership.

`types` is a one-column data.frame where the `type` variable contains the actor type of each node. The following values are possible:

- `gov` (government actor, e.g., a federal ministry)
- `ig` (interest group
- `io` (international organization)
- `par` (political party)
- `sci` (scientific organization)

intpos is a 30 x 6 matrix containing the interest positions of the 30 political actors on the six most salient political issues related to a pending new chemicals law. `-1` indicates a negative stance, i.e., the actor rejects the proposal; `1` indicates a positive stance, i.e., the actor supports the proposal; and `0` indicates a neutral or absent opinion.

Source

The data were collected using paper-based questionnaires. The questionnaires were administered in personal interviews (PAPI). Further information, including the actual survey, data on additional actors, the full names of the policy committees/forums, and the full list of unabbreviated actor names can be found online at http://hdl.handle.net/1902.1/17004 in the replication archive of Leifeld and Schneider (2012).

- Replication archive: http://hdl.handle.net/1902.1/17004
- AJPS publication: http://dx.doi.org/10.1111/j.1540-5907.2011.00580.x

The dataset is publicly available. Questions about the data or the original study should be directed to Volker Schneider (volker.schneider@uni-konstanz.de), the author of the original study and person who collected the data.

References

Leifeld, Philip and Volker Schneider (2012): Information Exchange in Policy Networks. *American Journal of Political Science* 53(3): 731–744. http://dx.doi.org/10.1111/j.1540-5907.2011.00580.x.

Schneider, Volker (1988): *Politiknetzwerke der Chemikalienkontrolle. Eine Analyse einer transnationalen Politikentwicklung*. Walter de Gruyter: Berlin/New York.

Schneider, Volker and Philip Leifeld (2009): Ueberzeugungssysteme, Diskursnetzwerke und politische Kommunikation: Ein zweiter Blick auf die deutsche Chemikalienkontrolle der 1980er Jahre. In: Volker Schneider, Frank Janning, Philip Leifeld and Thomas Malang (editors): *Politiknetzwerke. Modelle, Anwendungen und Visualisierungen*. Pages 139–158. Wiesbaden: VS Verlag fuer Sozialwissenschaften. http://dx.doi.org/10.1007%2F978-3-531-91883-9_6.

Examples

```
## Not run:
# Replication code for Leifeld and Schneider (2012), AJPS.
```

```
# Note that the estimates can only be reproduced approximately
# due to internal changes in the statnet package.

# preparatory steps
library("statnet")
library("xergm")
library("texreg")
seed <- 12345
set.seed(seed)
data("chemnet")

# create confirmed network relation
sci <- scito * t(scifrom)  # equation 1 in the AJPS paper
prefsim <- dist(intpos, method = "euclidean")  # equation 2
prefsim <- max(prefsim) - prefsim  # equation 3
prefsim <- as.matrix(prefsim)
committee <- committee %*% t(committee)  # equation 4
diag(committee) <- 0 # the diagonal has no meaning
types <- types[, 1]  # convert to vector

# create network objects and store attributes
nw.pol <- network(pol) # political/stratgic information exchange
set.vertex.attribute(nw.pol, "orgtype", types)
set.vertex.attribute(nw.pol, "betweenness",
    betweenness(nw.pol)) # centrality

nw.sci <- network(sci) # technical/scientific information exchange
set.vertex.attribute(nw.sci, "orgtype", types)
set.vertex.attribute(nw.sci, "betweenness",
    betweenness(nw.sci)) # centrality

# ERGM: model 1 in the AJPS paper; only preference similarity
model1 <- ergm(nw.pol ~ edges + edgecov(prefsim),
    control = control.ergm(seed = seed))
summary(model1)

# ERGM: model 2 in the AJPS paper; complete model
model2 <- ergm(nw.pol ~
    edges +
    edgecov(prefsim) +
    mutual +
    nodemix("orgtype", base = -7) +
    nodeifactor("orgtype", base = -1) +
    nodeofactor("orgtype", base = -5) +
    edgecov(committee) +
    edgecov(nw.sci) +
    edgecov(infrep) +
    gwesp(0.1, fixed = TRUE) +
    gwdsp(0.1, fixed = TRUE),
    control = control.ergm(seed = seed)
)
summary(model2)
```

```
# ERGM: model 3 in the AJPS paper; only preference similarity
model3 <- ergm(nw.sci ~ edges + edgecov(prefsim),
    control = control.ergm(seed = seed))
summary(model3)

# ERGM: model 4 in the AJPS paper; complete model
model4 <- ergm(nw.sci ~
    edges +
    edgecov(prefsim) +
    mutual +
    nodemix("orgtype", base = -7) +
    nodeifactor("orgtype", base = -1) +
    nodeofactor("orgtype", base = -5) +
    edgecov(committee) +
    edgecov(nw.pol) +
    edgecov(infrep) +
    gwesp(0.1, fixed = TRUE) +
    gwdsp(0.1, fixed = TRUE),
    control = control.ergm(seed = seed)
)
summary(model4)

# regression table using the texreg package
screenreg(list(model1, model2, model3, model4))

# goodness of fit using the btergm package
gof2 <- gof(model2, roc = FALSE, pr = FALSE)
gof2  # print gof output
plot(gof2)  # visual inspection of GOF

gof4 <- gof(model4, roc = FALSE, pr = FALSE)
gof4
plot(gof4)

# MCMC diagnostics
pdf("diagnostics2.pdf")
mcmc.diagnostics(model2)
dev.off()

pdf("diagnostics4.pdf")
mcmc.diagnostics(model4)
dev.off()

## End(Not run)
```

`getformula` *Retrieve a formula object from a model object*

Description

Retrieve a formula object from a model object.

Usage

```
getformula(x)
```

Arguments

x	A network model.

Details

The generic getformula function serves to extract a formula from a model object.

---

gof *Goodness of fit for inferential network models*

Description

Assess goodness of fit and degeneracy of inferential network models.

Usage

```
gof(object, ...)
```

Arguments

object	A network model.
...	Arbitrary further arguments are handed over to the simulate.formula function or the siena07 function. For details, refer to the help page of these functions.

Details

The generic gof function provides goodness-of-fit measures and degeneracy checks for inferential network models. See the specific help pages in the **btergm** package and other packages.

See Also

simulate.formula

handleMissings *Handle missing data in matrices.*

Description

Handle missing data in matrices.

Usage

```
handleMissings(mat, na = NA, method = "remove", logical = FALSE)
```

Arguments

mat	A matrix object.
na	The value that missing data are coded as. Usually NA, sometimes 9 or 10.
method	What should be done with the missing data? If method = "remove" is set, the function determines how many missing entries are in each row and column and iteratively removes rows or columns with the largest amount of missing data until no missing data are left in the matrix. If method = "fillmode" is set, the modal value of the matrix is identified (usually 0 in network matrices) and missing cells are imputed by filling in this modal value. method = "zero" replaces NAs by 0s.
logical	Return a matrix with logical values indicating which cells should be removed? By default the manipulated matrix is returned.

Details

This function deals with missing data in matrices or network objects used for inferential network analysis. It can either remove missing rows and/or columns iteratively (rows and columns with more NA values first, then successively rows and columns with fewer NA entries) or replace missing values by the modal value of the matrix or by 0. The function can return either the manipulated matrix or a matrix with logical values indicating which of the cells should be removed.

See Also

adjust

interpret *Generic interpretation function for statistical network models*

Description

Generic interpretation function for statistical network models.

Usage

```
interpret(object, ...)
```

Arguments

object	An ergm or btergm object.
...	Custom arguments to be handed over to subroutines.

Details

The interpret function facilitates interpretation of statistical network models at the micro level, as described in Desmarais and Cranmer (2012). See the specific methods in the **btergm** package.

References

Desmarais, Bruce A. and Skyler J. Cranmer (2012): Micro-Level Interpretation of Exponential Random Graph Models with Application to Estuary Networks. *The Policy Studies Journal* 40(3): 402–434.

is.mat.onemode *Check whether a matrix is one-mode or directed*

Description

Check whether a matrix represents a one-mode network (as opposed to a two-mode network) and whether a network represented by a matrix is directed (as opposed to undirected).

Usage

```
is.mat.onemode(mat)

is.mat.directed(mat)
```

Arguments

mat	A matrix object containing zeros and ones.

Details

The `is.mat.onemode` function returns `TRUE` if the input matrix `mat` represents a one-mode network and `FALSE` otherwise. The `is.mat.directed` function returns `TRUE` if the input matrix `mat` represents a directed network and `FALSE` otherwise.

---

knecht — *Longitudinal classroom friendship network and behavior (Andrea Knecht)*

Description

The Knecht dataset contains the friendship network of 26 pupils in a Dutch school class measured at four time points along with several demographic and behavioral covariates like age, sex, ethnicity, religion, delinquency, alcohol consumption, primary school co-attendance, and school advice. Some of these covariates are constant while others vary over time.

The full dataset (see Knecht 2006 and 2008) contains a large number of classrooms while the dataset presented here is an excerpt based on one single classroom. This excerpt was first used in a tutorial for the software **Siena** and the corresponding R package **RSiena** (Snijders, Steglich and van de Bunt 2010). The following description was largely copied from the original data description provided on the homepage of the **Siena** project (see below for the URL).

The data were collected between September 2003 and June 2004 by Andrea Knecht, supervised by Chris Baerveldt, at the Department of Sociology of the University of Utrecht (NL). The entire study is reported in Knecht (2008). The project was funded by the Netherlands Organisation for Scientific Research NWO, grant 401-01-554. The 26 students were followed over their first year at secondary school during which friendship networks as well as other data were assessed at four time points at intervals of three months. There were 17 girls and 9 boys in the class, aged 11–13 at the beginning of the school year. Network data were assessed by asking students to indicate up to twelve classmates which they considered good friends. Delinquency is defined as a rounded average over four types of minor delinquency (stealing, vandalism, graffiti, and fighting), measured in each of the four waves of data collection. The five-point scale ranged from 'never' to 'more than 10 times', and the distribution is highly skewed. In a range of 1–5, the mode was 1 at all four waves, the average rose over time from 1.4 to 2.0, and the value 5 was never observed.

Usage

```
data(knecht)
```

Format

Note: the data have to be transformed before they can be used with btergm and related packages (see examples below).

`friendship` is a list of adjacency matrices at four time points, containing friendship nominations of the column node by the row node. The following values are used: `0` = no, `1` = yes, `NA` = missing, `10` = not a member of the classroom (structural zero).

`demographics` is a data frame with 26 rows (the pupils) and four demographic variables about the pupils:

- `sex` (1 = girl, 2 = boy)
- `age` (in years)
- `ethnicity` (1 = Dutch, 2 = other, 0 = missing)
- `religion` (1 = Christian, 2 = non-religious, 3 = non-Christian religion, 0 = missing)

`primary` is a 26 x 26 matrix indicating whether two pupils attended the same primary school. 0 = no, 1 = yes.

`delinquency` is a data frame with 26 rows (the pupils) and four columns (the four time steps). It contains the rounded average of four items (stealing, vandalizing, fighting, graffiti). Categories: frequency over last three months, 1 = never, 2 = once, 3 = 2–4 times, 4 = 5–10 times, 5 = more than 10 times; 0 = missing.

`alcohol` is a data frame with 26 rows (the pupils) and 3 columns (waves 2, 3, and 4). It contains data on alcohol use ("How often did you drink alcohol with friends in the last three months?"). Categories: 1 = never, 2 = once, 3 = 2–4 times, 4 = 5–10 times, 5 = more than 10 times; 0 = missing.

`advice` is a data frame with one variable, "school advice", the assessment given at the end of primary school about the school capabilities of the pupil (4 = low, 8 = high, 0 = missing)

Source

The data were gathered by Andrea Knecht, as part of her PhD research, building on methods developed by Chris Baerveldt, initiator and supervisor of the project. The project is funded by the Netherlands Organisation for Scientific Research NWO, grant 401-01-554, and is part of the research program "Dynamics of Networks and Behavior" with principle investigator Tom A. B. Snijders.

- Complete original data: https://easy.dans.knaw.nl/ui/datasets/id/easy-dataset:48665
- This excerpt in Siena format: http://www.stats.ox.ac.uk/~snijders/siena/klas12b.zip
- Siena dataset description: http://www.stats.ox.ac.uk/~snijders/siena/tutorial2010_data.htm

Permission to redistribute this dataset along with this package was granted by Andrea Knecht on April 17, 2014. Questions about the data or the original study should be directed to her.

References

Knecht, Andrea (2006): *Networks and Actor Attributes in Early Adolescence* [2003/04]. Utrecht, The Netherlands Research School ICS, Department of Sociology, Utrecht University. (ICS-Codebook no. 61).

Knecht, Andrea (2008): *Friendship Selection and Friends' Influence. Dynamics of Networks and Actor Attributes in Early Adolescence*. PhD Dissertation, University of Utrecht. http://dspace.library.uu.nl/bitstream/handle/1874/25950/full.pdf.

Knecht, Andrea, Tom A. B. Snijders, Chris Baerveldt, Christian E. G. Steglich, and Werner Raub (2010): Friendship and Delinquency: Selection and Influence Processes in Early Adolescence.

Social Development 19(3): 494–514. http://dx.doi.org/10.1111/j.1467-9507.2009.00564.x.

Leifeld, Philip and Skyler J. Cranmer (2014): A Theoretical and Empirical Comparison of the Temporal Exponential Random Graph Model and the Stochastic Actor-Oriented Model. Paper presented at the 7th Political Networks Conference, McGill University, Montreal, Canada, May 30. http://arxiv.org/abs/1506.06696.

Leifeld, Philip, Skyler J. Cranmer and Bruce A. Desmarais (2017): Temporal Exponential Random Graph Models with btergm: Estimation and Bootstrap Confidence Intervals. *Journal of Statistical Software*.

Snijders, Tom A. B., Christian E. G. Steglich, and Gerhard G. van de Bunt (2010): Introduction to Actor-Based Models for Network Dynamics. *Social Networks* 32: 44–60. http://dx.doi.org/10.1016/j.socnet.2009.02.004.

Steglich, Christian E. G. and Andrea Knecht (2009): Die statistische Analyse dynamischer Netzwerkdaten. In: Stegbauer, Christian and Roger Haeussling (editors), *Handbuch der Netzwerkforschung*, Wiesbaden: Verlag fuer Sozialwissenschaften.

Examples

```
## Not run:
# ========================================================================
# The following example was taken from the JSS article about btergm
# that is referenced above (Leifeld, Cranmer and Desmarais 2017).
# ========================================================================

require("texreg")
require("sna")
require("xergm")
require("RSiena")
data("knecht")

# step 1: make sure the network matrices have node labels
for (i in 1:length(friendship)) {
  rownames(friendship[[i]]) <- 1:nrow(friendship[[i]])
  colnames(friendship[[i]]) <- 1:ncol(friendship[[i]])
}
rownames(primary) <- rownames(friendship[[1]])
colnames(primary) <- colnames(friendship[[1]])
sex <- demographics$sex
names(sex) <- 1:length(sex)

# step 2: imputation of NAs and removal of absent nodes:
friendship <- handleMissings(friendship, na = 10, method = "remove")
friendship <- handleMissings(friendship, na = NA, method = "fillmode")

# step 3: add nodal covariates to the networks
for (i in 1:length(friendship)) {
  s <- adjust(sex, friendship[[i]])
  friendship[[i]] <- network(friendship[[i]])
  friendship[[i]] <- set.vertex.attribute(friendship[[i]], "sex", s)
  idegsqrt <- sqrt(degree(friendship[[i]], cmode = "indegree"))
```

```
  friendship[[i]] <- set.vertex.attribute(friendship[[i]],
      "idegsqrt", idegsqrt)
  odegsqrt <- sqrt(degree(friendship[[i]], cmode = "outdegree"))
  friendship[[i]] <- set.vertex.attribute(friendship[[i]],
      "odegsqrt", odegsqrt)
}
sapply(friendship, network.size)

# step 4: plot the networks
pdf("knecht.pdf")
par(mfrow = c(2, 2), mar = c(0, 0, 1, 0))
for (i in 1:length(friendship)) {
  plot(network(friendship[[i]]), main = paste("t =", i),
  usearrows = TRUE, edge.col = "grey50")
}
dev.off()

# step 5: estimate TERGMS without and with temporal dependencies
model.2a <- btergm(friendship ~ edges + mutual + ttriple +
    transitiveties + ctriple + nodeicov("idegsqrt") +
    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
    edgecov(primary), R = 100)

model.2b <- btergm(friendship ~ edges + mutual + ttriple +
    transitiveties + ctriple + nodeicov("idegsqrt") +
    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
    edgecov(primary) + delrecip + memory(type = "stability"),
    R = 100)

# step 6: alternatively, estimate via MCMC-MLE:
model.2d <- mtergm(friendship ~ edges + mutual + ttriple +
    transitiveties + ctriple + nodeicov("idegsqrt") +
    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
    edgecov(primary) + delrecip + memory(type = "stability"),
    control = control.ergm(MCMC.samplesize = 5000, MCMC.interval = 2000))

# step 7: GOF assessment with out-of-sample prediction
model.2e <- btergm(friendship[1:3] ~ edges + mutual + ttriple +
    transitiveties + ctriple + nodeicov("idegsqrt") +
    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
    edgecov(primary) + delrecip + memory(type = "stability"),
    R = 100)

gof.2e <- gof(model.2e, nsim = 100, target = friendship[[4]],
    formula = friendship[3:4] ~ edges + mutual + ttriple +
    transitiveties + ctriple + nodeicov("idegsqrt") +
    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
    edgecov(primary) + delrecip + memory(type = "stability"),
```

```
    coef = coef(model.2b), statistics = c(esp, dsp, geodesic,
    deg, triad.undirected, rocpr))
pdf("gof-2e.pdf", width = 8, height = 6)
plot(gof.2e)
dev.off()

## End(Not run)
```

Index