Package 'tnam'

December 14, 2015

Version 1.6

Date 2015-12-09

Title Temporal Network Autocorrelation Models (TNAM)

Description Temporal and cross-sectional network autocorrelation models (TNAM).

Depends R (>= 2.14.0), xergm.common (>= 1.6)

Imports methods, utils, stats, network, sna, igraph, vegan, lme4 (>= 1.0), Rcpp (>= 0.11.0)

Suggests xergm, texreg, statnet

License GPL (>= 2)

LinkingTo Rcpp

NeedsCompilation yes

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

Maintainer Philip Leifeld <philip.leifeld@eawag.ch>

Repository CRAN

Date/Publication 2015-12-14 18:02:24

R topics documented:

tnam-package *Temporal Network Autocorrelation Models (TNAM)*

Description

Temporal Network Autocorrelation Models (TNAM).

Details

The **tnam** package implements temporal and cross-sectional network autocorrelation models (TNAM).

Author(s)

Philip Leifeld (http://www.philipleifeld.de)

Skyler J. Cranmer (http://www.skylercranmer.net)

tnam *Fit (temporal) network autocorrelation models*

Description

Fit (temporal) network autocorrelation models.

Usage

```
tnam(formula, family = gaussian, re.node = FALSE,
    re.time = FALSE, time.linear = FALSE, time.quadratic = FALSE,
    center.y = FALSE, na.action = na.omit, ...)

tnamdata(formula, center.y = FALSE)
```

Arguments

formula	A formula where the left-hand side specifies either a vector containing the outcome variable (for a cross-sectional model) or a list of such vectors (for modeling the outcome at multiple time steps) or a data frame with one time step per column (also for longitudinal models of behavior). The right-hand side of the formula consists of tnam-specific model terms like netlag, structsim and other terms which are described on the help page of tnam-terms.
family	The link function for fitting the generalized linear model or the mixed effects model, for example gaussian or binomial. The options are the same as in the glm and glmer functions. For details on the family argument, see the family help page.
re.node	If multiple time steps are present: should a random effect for the nodes be added to the model? This results in the estimation of a mixed effects model.

re.time	If multiple time steps are present: should a random effect for the time steps be added to the model? This results in the estimation of a mixed effects model.
time.linear	If multiple time steps are present: should a linear effect for time be added to the model? This can be estimated in the standard GLM framework.
time.quadratic	If multiple time steps are present: should a squared effect for time be added to the model? This can be estimated in the standard GLM framework.
center.y	Center the dependent variable by subtracting the mean from the actual value within each time step?
na.action	How should missing values be treated? By default, they are omitted. See the na.omit help page for details.
...	Further arguments that should be passed to the glm, lmer, or glmer function, which is used under the hood for estimating the model.

Details

The tnam function serves to estimate temporal or cross-sectional network autocorrelation models. Model terms such as spatial lags, temporal lags, spatio-temporal lags, centrality etc. can be specified in the formula argument. Details on the model terms can be found on the tnam-terms help page.

The tnamdata function accepts a formula (like in the tnam function) and returns a data frame with the response variable and the covariates for estimation with any estimation function. tnam first calls tnamdata internally and then hands over the resulting data structure to a glm, lmer, or nlmer call. If models such as tobit, multinomial or multilevel models should be estimated, one can leave out the estimation step and feed the results of tnamdata manually into any type of model.

See Also

tnam-package tnam-terms knecht

Examples

```
# The following example models delinquency among adolescents at
# multiple time steps as a function of (1) their nodal attributes
# like sex or religion, (2) their peers' delinquency levels, (3)
# their own and their peers' past delinquency behavior, and (4)
# their structural position in the network. See ?knecht for
# details on the dataset. Before estimating the model, all data
# should be labeled with the names of the nodes such that tnam
# is able to merge the information on multiple nodes across time
# points.

library("tnam")
data("knecht")

# prepare the dependent variable y
delinquency <- as.data.frame(delinquency)
rownames(delinquency) <- letters

# replace structural zeros (denoted as 10) and add row labels
friendship[[3]][friendship[[3]] == 10] <- NA
```

```
friendship[[4]][friendship[[4]] == 10] <- NA
for (i in 1:length(friendship)) {
  rownames(friendship[[i]]) <- letters
}

# prepare the covariates sex and religion
sex <- demographics$sex
names(sex) <- letters
sex <- list(t1 = sex, t2 = sex, t3 = sex, t4 = sex)
religion <- demographics$religion
names(religion) <- letters
religion <- list(t1 = religion, t2 = religion, t3 = religion,
    t4 = religion)

# Estimate the model. The first term is the sex of the respondent,
# the second term is the religion of the respondent, the third
# term is the previous delinquency behavior of the respondent,
# the fourth term is the delinquency behavior of direct friends,
# the fifth term is the delinquency behavior of indirect friends
# at a path distance of 2, the sixth effect is the past delinquency
# of direct friends, the seventh term indicates whether the
# respondent has any contacts at all, and the last term captures
# the effect of the betweenness centrality of the respondent on
# his or her behavior. Apparently, previous behavior, being an
# isolate, and religion seem to have an effect on delinquency in
# this dataset. There is also a slight positive trend over time,
# and direct friends exert a minor effect (not significant).
# Note that a linear model may not be the best specification for
# modeling the ordered categorical delinquency variable, but it
# suffice here for illustration purposes.

model1 <- tnam(
    delinquency ~
    covariate(sex, coefname = "sex") +
    covariate(religion, coefname = "religion") +
    covariate(delinquency, lag = 1, exponent = 1) +
    netlag(delinquency, friendship) +
    netlag(delinquency, friendship, pathdist = 2, decay = 1) +
    netlag(delinquency, friendship, lag = 1) +
    degreedummy(friendship, deg = 0, reverse = TRUE) +
    centrality(friendship, type = "betweenness"),
    re.node = TRUE, time.linear = TRUE
)
summary(model1)

# for nice table output, use the texreg package
library("texreg")
screenreg(model1)
```

---

tnam-terms *Terms used in (Temporal) Network Autocorrelation Models (tnam)*

Description

The function tnam is used to fit (temporal) network autocorrelation models.

The function tnamdata can be used alternatively to create a data frame containing all the data ready for estimation. This may be useful when a non-standard model should be estimated, like a tobit model or a model with zero inflation, for example.

Both functions accept a formula containing several model terms. The model terms are themselves functions which can be called separately. For example, one model term is called netlag. This model term can be part of the formula handed over to the tnam function, or netlag can be called directly in order to create a single variable.

This help page describes the different model terms available in (temporal) network autocorrelation models. See the tnam help page for details on the model.

Usage

```
attribsim(y, attribute, match = FALSE, lag = 0,
    normalization = c("no", "row", "column"), center = FALSE,
    coefname = NULL)

centrality(networks, type = c("indegree", "outdegree", "freeman",
    "betweenness", "flow", "closeness", "eigenvector",
    "information", "load", "bonpow"), directed = TRUE, lag = 0,
    rescale = FALSE, center = FALSE, coefname = NULL, ...)

cliquelag(y, networks, k.min = 2, k.max = Inf, directed = TRUE,
    lag = 0, normalization = c("no", "row", "column"),
    center = FALSE, coefname = NULL)

clustering(networks, directed = TRUE, lag = 0, center = FALSE,
    coefname = NULL, ...)

covariate(y, lag = 0, exponent = 1, center = FALSE,
    coefname = NULL)

degreedummy(networks, deg = 0, type = c("indegree", "outdegree",
    "freeman"), reverse = FALSE, directed = TRUE, lag = 0,
    center = FALSE, coefname = NULL, ...)

interact(x, y, lag = 0, center = FALSE, coefname = NULL)

netlag(y, networks, lag = 0, pathdist = 1, decay = pathdist^-1,
    normalization = c("no", "row", "column", "complete"),
    reciprocal = FALSE, center = FALSE, coefname = NULL, ...)

structsim(y, networks, lag = 0, method = c("euclidean",
    "minkowski", "jaccard", "binary", "hamming"), center = FALSE,
    coefname = NULL, ...)
```

```
weightlag(y, networks, lag = 0, normalization = c("no", "row",
    "column"), center = FALSE, coefname = NULL)
```

Arguments

attribute	A vector, list of vectors or data frame with the same dimensions as y. Based on this attribute, the similarity between nodes i and j will be calculated, and the resulting similarity matrix is used to weight the y variable.
center	Should the model term be centered? That is, should the mean of the variable be subtracted from the actual value at each time step?
coefname	An additional name that is used as part of the coefficient label for easier identification in the summary output of the model.
decay	For each value in pathdist, the decay argument specifies the relative importance. By default, a geometric decay is used, that is, the behavior of nodes at path distance 2 is counted only half as much as the behavior of adjacent nodes. Alternatively, if both are equally important, it is possible to write pathdist = c(1, 2) and decay = c(1, 1).
deg	The degree (e.g., deg = 2) or degree range (e.g., deg = 1:3).
directed	Is the input matrix or network a directed network?
exponent	The exponent of a covariate. For example, exponent = 2 creates a squared variable. This may be helpful for modeling non-linear effects or for modeling a quadratic behavior shape.
k.max	Maximal clique size.
k.min	Minimal clique size.
lag	The temporal lag. The default value 0 means there is no lag. A value of 1 would specify a single-period lag, that is, current behavior is modeled conditional on previous influence. A value of 2 would specify a two-period lag, that is, current behavior is modeled conditional on pre-previous influence, etc.
match	If match = FALSE, a similarity matrix is computed by subtracting node j's attribute value from node i's attribute value, standardizing the resulting distance between 0 and 1, and converting it into a similarity by subtracting it from 1. This similarity matrix is used as a weight matrix to compute a spatial lag. If match = TRUE is specified, the weight matrix contains values of 1 whenever node i and j have the same attribute value and 0 otherwise.
method	The distance function used for computing structural similarity. Possible values are "euclidean", "minkowski", "jaccard", "binary", and "hamming".
networks	The network(s) for computing the peer influence, also known as the weight matrix. This can be a matrix or a network object (for a single time step) or a list of matrices or network objects (for multiple time steps).
normalization	Possible values: "no" for switching off normalization, "row" for row normalization of the weight matrix, "column" for column normalization of the weight matrix, and "complete" for complete normalization. If "no" is selected, this corresponds to the total similarity or the sum of all influences of tied alters.

If "row" is selected, this corresponds to the average alter effect. If i is the row node and j is the column node, row normalization computes the peer influence of j on i as a fraction of i's overall number of outgoing ties (i.e., i's row sum or outdegree centrality). The theoretical intuition is that other nodes' influences on i are not cumulative; i rather perceives the average influence of his or her peers. Note that row normalization does not necessarily entail that the values are standardized between 0 and 1.

If "column" is selected, this corresponds to the peer influence of node j on node i as a fraction of the number of incoming ties j has (i.e., j's column sum or indegree centrality). This captures the theoretical effect that j may distribute his or her influence among many nodes, in which case j's influence on i is relatively weak. Thus the "exerted influence of j on i decreases with the number of actors j influences" (Leenders 2002). Note that column normalization does not necessarily entail that the values are standardized between 0 and 1.

If "complete" is selected, this captures the peer influence of node j on node i as a fraction of all nodes' cumulative outcome values (including non-tied dyads; except i's own outcome value). In other words, complete normalization corresponds to the actual exposure of i to the influence of his or her tied alters j over the the exposure i could receive if i were tied to all other nodes in the network. If a decay <= 1 is used and if only direct friends are considered, this effectively standardizes the influence scores between 0 and 1.

`pathdist` An integer or a vector of integers. For example, if pathdist = 1 is used, this computes the sum of the behavior of adjacent nodes. If pathdist = 2 is specified, this computes the effect of indirect paths of length 2 ("friends of friends"). If pathdist = 1:2 is set, both directly connected nodes' behavior and the behavior of nodes at a path distance of 2 from the focal node are counted. Arbitrary (sets of) path distances can be used. See also the decay argument.

`reciprocal` If reciprocal = TRUE is specified, only the behavior of nodes to which a reciprocal relation exists is counted (that is, a link in both directions).

`rescale` Should the centrality index be rescaled between 0 and 1?

`reverse` Reverse the selection of degrees. For example, when deg = 0 and reverse = FALSE are specified, resulting values of 1 indicate that a node has no connections, whereas the combination deg = 0 and reverse = TRUE results in the value 1 representing nodes which have a degree of at least 1.

`type` The type of centrality measure. Possible values are "indegree", "outdegree", "freeman", "betweenness", "flow", "closeness", "eigenvector", "information", "load", and "bonpow".

`x` A variable that should be interacted with y. Either a vector or a list of vectors or another model term (this is the preferred way).

`y` The outcome or behavior variable. Either a vector (for a single time step) or a list of vectors with named elements in each vector (for multiple time steps) or a data frame with row names where each column is one time step (for multiple time steps).

`...` Additional arguments to be handed over to subroutines.

Model terms for `tnam`

attribsim *Spatial lag based on attribute similarity* The `attribsim` model term computes a similarity matrix based on the `attribute` argument and uses this similarity matrix to construct a spatial lag by multiplying the similarity matrix and the outcome vector `y`. The intuition behind this model term is that node i's behavior may be influenced by node j's behavior if nodes i and j are similar on another dimension. For example, if i and j both smoke while k does not smoke, j's alcohol consumption may affect i's alcohol consumption to a larger extent than node k's alcohol consumption. In this example, the `y` outcome variable is alcohol consumption and the `attribute` argument is smoking. If `match = FALSE`, the absolute similarity between i and j is computed by subtracting j's attribute value from i's attribute value and taking the absolute value to construct the similarity matrix. If `match = TRUE`, the function computes a matrix containing values of 1 if i and j have the same attribute value and 0 otherwise. A scenario where the `attribsim` model term makes sense is degree assortativity: if i and j have the same degree centrality, they may be inclined to learn from each other's behavior, even in the absence of a direct connection between them.

centrality *Node centrality* The `centrality` model term computes a centrality index for the nodes in a network or matrix. This can capture important structural effects because being central often implies certain constraints or opportunities more peripheral nodes do not have. For example, central nodes in a network of employees might be able to perform better.

cliquelag *Spatial lag of k-clique co-members* The `cliquelag` model term computes a clique co-membership matrix and multiplies this matrix with the outcome variable. The intuition behind this is that in some settings individuals may be influenced to a particularly strong extent by peers in the same cliques. A clique is defined as a maximal connected subgraph of size k. For example, a deviant behavior of a person may be conditioned by the deviant behavior of the person's friends – but only if these friends are tied to each other as well so that a clique among these persons exists. A minimal and a maximal `k` may be defined, where `k` is the size of the cliques. In the clique co-membership matrix, all cliques with `k.min <= k <= k.max` are included.

clustering *Local clustering coefficient*, or *transitivity* The `clustering` model term computes the local clustering coefficient, which is also known as transitivity. This index has high values if the direct neighborhood of a node is densely interconnected. For example, if one's friends are friends with each other, this may have repercussions on ego's behavior.

covariate *Exogenous nodal covariate* The `covariate` model term adds an exogenous nodal covariate to the model. For example, when performance of employees is modeled, a covariate could be seniority of these employees. It is possible to add lagged covariates to model the effect of past nodal attributes on current behavior. Similarly, this model term can be used to add autoregressive terms, that is, the effect of previous behavior on current behavior.

degreedummy *Dummy variable for degree centrality values* The `degreedummy` model term controls for specific degree centralities or ranges of degree centrality. For example, do nodes with a degree of 0 (isolates) show different behavior than nodes who are connected? Or do nodes with a degree centrality larger than three exert different behavior?

interact *Interactions between other model terms* The `interact` model term adds an interaction effect between two other model terms by multiplying the result vectors of these two model terms. When using interaction terms, centering the result is recommended. Note that only the interaction term is created; the main effects must be introduced to the model using the other model terms.

netlag *Spatial network lag* The `netlag` model term captures the autocorrelation inherent in networks. For example, when political actors are members of a policy network, their success of achieving policy outcomes is not independent from each other. Most likely, being connected to policy winners increases the success rate. In many settings, indirect effects may be important as well: how does the behavior of my friends' friends affect my own behavior? In some contexts, spatio-temporal lags are useful: how does the past behavior of my friends affect my current behavior? The `netlag` model term is designed for binary networks because things like indirect effects, restriction to reciprocal dyads, decay of indirect relations etc. is possible. For weighted networks, the `weightlag` term is recommended. If no other arguments are specified and loops are absent and a binary matrix is used, both model terms produce the same results.

structsim *Structural similarity* The `structsim` model term computes the structural similarity with other nodes in the network and multiplies this similarity matrix with the outcome variable. The intuition is that behavior is sometimes affected by comparison with structurally similar nodes. For example, a worker may be impressed by the performance of other workers who are embedded in the same team or who report to the same bosses. As with the other model terms, temporal lags are possible.

weightlag *Weighted spatial lag* The `weightlag` model term captures spatial autocorrelation in weighted networks. For example, the GDP per capita of a country may be affected by the GDP of proximate other countries or by the GDP of trade partners. In these cases, indirect contacts etc. do not make any sense, therefore the distinction between the `weightlag` and the `netlag` model term. The weight matrix is multiplied by the outcome variable, possibly after row or column normalization.

References

Leenders, Roger Th. A. J. (2002): Modeling Social Influence through Network Autocorrelation: Constructing the Weight Matrix. *Social Networks* 24: 21–47. http://dx.doi.org/10.1016/S0378-8733(01)00049-1.

Daraganova, Galina and Garry Robins (2013): Autologistic Actor Attribute Models. In: Lusher, Dean, Johan Koskinen and Garry Robins, "Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications", Cambridge University Press, chapter 9: 102–114.

Hays, Jude C., Aya Kachi and Robert J. Franzese Jr. (2010): A Spatial Model Incorporating Dynamic, Endogenous Network Interdependence: A Political Science Application. *Statistical Methodology* 7: 406–428. http://dx.doi.org/10.1016/j.stamet.2009.11.005

See Also

tnam-package tnam tnamdata knecht

Index