Package ‘goldfish’

October 13, 2022

Encoding UTF-8
Type Package
Title Statistical Network Models for Dynamic Network Data
Version 1.6.4
Date 2022-08-09
License GPL (>= 3)

Description Tools for fitting statistical network models to dynamic network data. Can be used for fitting both dynamic network actor models ('DyNAMs') and relational event models ('REMs').
Stadtfeld, Hollway, and Block (2017a) <doi:10.1177/0081175017709295>,
Stadtfeld, Hollway, and Block (2017b) <doi:10.1177/0081175017733457>,
Stadtfeld and Block (2017) <doi:10.15195/v4.a14>,

URL https://snlab-ch.github.io/goldfish/
BugReports https://github.com/snlab-ch/goldfish/issues
Depends R (>= 3.5.0)
Imports Rcpp (>= 1.0.1), changepoint, methods, utils, stats, generics,
ggplot2, tibble
LinkingTo Rcpp, RcppArmadillo
Suggests testthat, knitr, devtools, covr, rmarkdown, pixiedust,
igraph, ggraph, migraph, broom, lmtest

VignetteBuilder knitr
LazyData true
RoxygenNote 7.2.1

NeedsCompilation yes
Config/testthat/edition 3

Author James Hollway [cre, aut, dtc] (IHEID,
<https://orcid.org/0000-0002-8361-9647>),
Christoph Stadtfeld [aut, dtc],
Marion Hoffman [ctb],
Alvaro Uzaheta [ctb] (<https://orcid.org/0000-0003-4367-3670>),
Mirko Reul [ctb],
Timon Elmer [ctb],
Kieran Mepham [ctb],
Per Block [ctb],
Xiaolei Zhang [ctb],
Weigutian Ou [ctb],
Emily Garvin [ctb],
Siwei Zhang [ctb]

Maintainer James Hollway <james.hollway@graduateinstitute.ch>

Repository CRAN

Date/Publication 2022-08-24 08:30:06 UTC

R topics documented:

goldfish-package ........................................... 2
defineDependentEvents .................................... 3
defineGlobalAttribute ..................................... 5
defineGroups_interaction .................................. 5
defineNetwork ............................................. 7
defineNodes ................................................ 8
estimate ...................................................... 10
examine ...................................................... 14
Fisheries_Treaties_6070 ................................... 16
linkEvents ................................................... 18
logLik.result.goldfish ..................................... 20
print-method ............................................... 22
RFID_Viability_Study ...................................... 24
Social_Evolution .......................................... 25
update-method ............................................. 26

Index 28

goldfish-package goldfish package

Description

The goldfish Project is an R package that allows to fit statistical network models (such as DyNAM and REM) to dynamic network data.

Details

The goldfish package in R allows the study of time-stamped network data using a variety of models. In particular, it implements different types of Dynamic Network Actor Models (DyNAMs), a class of models that is tailored to the study of actor-oriented network processes through time. Goldfish also implements different versions of the tie-oriented Relational Event Model by Carter Butts.
defineDependentEvents

References


See Also

`estimate()`

defineDependentEvents  Define dependent events for a model

Description

The final step in defining the data objects is to identify the dependent events.

Usage

```r
defineDependentEvents(
  events,
  nodes,
  nodes2 = NULL,
  defaultNetwork = NULL,
  envir = environment()
)
```

Arguments

- `events`: a data frame containing the event list that should be considered as a dependent variable in models.
- `nodes`: a data frame or a `nodes.goldfish` object containing the nodes used in the event list.
- `nodes2`: a second nodeset in the case that the events occurs in a two-mode network.
- `defaultNetwork`: the name of a `network.goldfish` object.
- `envir`: An `environment` object where the nodes-set and default network objects are defined. The default value is `environment()`.
defineDependentEvents

Details

Before this step is performed, we have to define: the nodeset (defineNodes()), the network (defineNetwork()) and the link the event list to the network (linkEvents()).

During the definition as a dependent event, some checks are done to ensure consistency with the default network and the nodeset. In particular, consistency of the labels of nodes in the events with the nodes’ labels in the network and the nodeset is done.

It is possible to define as a dependent event a different set of events to the ones link to the default network. This is useful to model different type of events where the event dynamic is driven by different effects or its weight differs. Fisheries_Treaties_6070 has an example of it, the relational event modeled are fisheries treaties between countries. The bilatchanges data frame contains information of creation and dissolution of treaties. vignette(teaching2) shows how to model just the creation of treaties conditional on creation and dissolution.

Value

an object with additional class dependent.goldfish with attributes:

- nodes a character vector with the names of the nodes set that define the dimensions of the defaultNetwork. nodes and nodes2 arguments.

- defaultNetwork A character value with the name of the network object when this is present. defaultNetwork argument.

- type A character value that can take values monadic or dyadic depending on the arguments used during the definition.

The object can be modified using methods for data frames.

See Also

defineNodes(), defineNetwork(), linkEvents()

Examples

actors <- data.frame(
  actor = 1:5, label = paste("Actor", 1:5),
  present = TRUE, gender = sample.int(2, 5, replace = TRUE)
)
actors <- defineNodes(nodes = actors)
calls <- data.frame(
  time = c(12, 27, 45, 56, 66, 68, 87),
  sender = paste("Actor", c(1, 3, 5, 2, 3, 4, 2)),
  receiver = paste("Actor", c(4, 2, 3, 5, 1, 2, 5)), increment = rep(1, 7)
)
callNetwork <- defineNetwork(nodes = actors)
callNetwork <- linkEvents(
  x = callNetwork, changeEvent = calls, nodes = actors
)

# Defining the dependent events:
callDependent <- defineDependentEvents(
  events = calls, nodes = actors, defaultNetwork = callNetwork
)
defineGlobalAttribute  

**Define a global time-varying attribute**

**Description**

This function allows to define a global attribute of the nodeset (i.e a variable that is identical for each node but changes over time).

**Usage**

```r
defineGlobalAttribute(global)
```

**Arguments**

- `global` a data frame containing all the values this global attribute takes along time.

**Details**

For instance, seasonal climate changes could be defined as a changing global attribute. Then, this global attribute can be linked to the nodeset by using `linkEvents()`.

**Value**

an object of class `global.goldfish`

**Examples**

```r
seasons <- defineGlobalAttribute(data.frame(time = 1:12, replace = 1:12))
```

---

**defineGroups_interaction**

*To define the second mode of a DyNAM-i model*

**Description**

This function creates all objects necessary to the estimation of a DyNAM-i model model = "DyNAMi" from dyadic interaction records and an actor set. It first creates a nodeset for the second mode of the interaction network that will be modeled, i.e. the interaction groups set, and an event list that indicates when groups are present or not through time. It then creates a list of interaction events, between actors and groups, in which an actor either joins or leaves a group. It is decomposed in an list of dependent events (that should be modeled) and a list of exogenous events (that should not be modeled). For example when an actor leaves a group and joins her own singleton group, only the leaving event is modeled but not the joining one, and vice versa when an actor belonging to a singleton group joins another group.
defineGroups_interaction

Usage

```r
defineGroups_interaction(
  records,
  actors,
  seed.randomization,
  progress = getOption("progress")
)
```

Arguments

- **records**: an object of class `data.frame` that is a list of rows of type `node A, nodeB, Start, End`, where `nodeA` and `nodeB` indicate the actors involved in a dyadic interaction, and `Start` and `End` indicating the starting and ending time of their interaction.
- **actors**: a object of class `nodes.goldfish` that defines the actors interacting (labels in `records` and `actors` should be identical).
- **seed.randomization**: an integer used whenever there should be some random choice to be made.
- **progress**: logical weather detailed information of intermediate steps should be printed in the console.

Details

It is important to notice that sometimes some random decisions have to be made regarding who joined or left a group, for example when two actors start interacting but we do not know who initiated the interaction. To test for the robustness of such a procedure, one can use different randomization seeds and run the model several times.

Value

A list with the following data frames:

- **interaction.updates**: containing all joining and leaving events
- **groups**: containing the nodeset corresponding to interaction groups (the second mode of the network)
- **dependent.events**: for the events that should be modeled
- **exogenous.events**: that are not modeled (for example when an actor leaves a group and joins its own singleton group, only the leaving event is modeled but not the joining event)
- **composition.changes**: that is an events list that should be attached to the groups nodeset to indicate when a group is present or not
The function defines a network object either from a nodeset or from a matrix (sociomatrix or adjacency matrix). If a matrix is used as input, `defineNetwork()` returns a network filled with the same values as the ones present in the provided network. If the nodeset is the only argument, `defineNetwork()` returns an empty network with the number of columns and rows corresponding to the size of the nodeset. These networks are static, but they can be turned into dynamic networks by linking dynamic events to the network object using `linkEvents()`.

**Usage**

```r
defineNetwork(
  matrix = NULL,
  nodes,
  nodes2 = NULL,
  directed = TRUE,
  envir = environment()
)
```

**Arguments**

- `matrix` An initial matrix (optional), and object of class `matrix`.
- `nodes` A node-set (see `defineNodes()`).
- `nodes2` A second optional node-set for the definition of two-mode networks.
- `directed` A logical value indicating whether the network is directed.
- `envir` An `environment` object where the nodes-set objects are defined. The default value is `environment()`.

**Details**

If a matrix is used as input, its dimension names must be a subset of the nodes in the nodeset as defined with the `defineNodes()` and the order of the labels in rows and columns must correspond to the order of node labels in the nodeset. The matrix can be directed or undirected (as specified with the directed argument).

If the network is updated over time (e.g., a new wave of friendship data is collected), these changes can be added with the `linkEvents()` - similar to link changing attribute events to a nodeset. This time, the user needs to provide the network and the associated nodeset. If no matrix is provided, goldfish only considers the nodeset and assumes the initial state to be empty (i.e., a matrix containing only 0s).
Value

an object with additional class network.goldfish with attributes:

- **nodes** a character vector with the names of the nodes set objects used during the definition. `nodes` and `nodes2` arguments.
- **directed** Logical value indicating whether the network is directed. `directed` argument
- **events** An empty character vector. `linkEvents()` is used to link event data frames.

The object can be modified using methods for matrix.

See Also

`defineNodes()`, `linkEvents()`

Examples

```r
# If no initial matrix is provided
data("Social_Evolution")
callNetwork <- defineNetwork(nodes = actors)

# If a initial matrix is provided
data("Fisheries_Treaties_6070")
bilatnet <- defineNetwork(bilatnet, nodes = states, directed = FALSE)
```

---

defineNodes

*Defining a node set with (dynamic) node attributes.*

Description

The `defineNodes()` function processes and checks the data.frame passed to the `nodes` argument. This is a recommended step before the definition of the network.

Usage

defineNodes(nodes)

Arguments

- **nodes** a data.frame object with the nodes attributes with the following reserved names
  - `label` character variable containing the nodes labels (mandatory)
  - `present` logical variable indicating if the respective node is present at the first time-point (optional)
**Details**

Additional variables in the nodes data frame object are considered as the initial values of the nodes attributes. Those variables must be of class numeric, character, logical.

It is important that the initial definition of the node set contain all the nodes that could be potential senders or receivers of events. In case that all the nodes are not available at all times, the present variable can be used to define compositional changes. Therefore, the initial node set would contain all the potential senders and receivers nodes and the variable present will indicate all the nodes present at the beginning as senders or receivers. Using `linkEvents()` is possible to link events where the composition of available nodes changes over time, see vignette("teaching2").

For the attributes in the nodeset to become dynamic, them can be linked to a dynamic event-list data frames in the initial state object by using the `linkEvents()`. A new call of `linkEvents()` is required for each attribute that is dynamic.

Objects of class tbl_df from the tibble package frequently use in the tidyverse ecosystem and objects of class data.table will produce errors in later steps for goldfish. Current implementation of goldfish relies on the subsetting behavior of data frames objects. The previous mentioned objects classes change this behavior producing errors.

**Value**

an object with an additional class nodes.goldfish with attributes:

- **events**
  An empty character vector. `linkEvents()` is used to link event data frames.

- **dynamicAttributes**
  An empty character vector. `linkEvents()` is used to link event data frames and their related attribute.

The object can be modified using methods for data frames.

**See Also**

`defineNetwork()`, `linkEvents()`

**Examples**

```r
nodesAttr <- data.frame(
  label = paste("Actor", 1:5),
  present = c(TRUE, FALSE, TRUE, TRUE, FALSE),
  gender = c(1, 2, 1, 1, 2)
)
nodesAttr <- defineNodes(nodes = nodesAttr)

# Social evolution nodes definition
data("Social_Evolution")
actors <- defineNodes(actors)

# Fisheries treaties nodes definition
data("Fisheries_Treaties_6070")
states <- defineNodes(states)
```
**estimate**

*Estimate a model*

**Description**

Estimates parameters for a dynamic network model via maximum likelihood implementing the iterative Newton-Raphson procedure as describe in Stadtfeld and Block (2017).

**Usage**

```r
estimate(
  x,
  model = c("DyNAM", "REM", "DyNAMi"),
  subModel = c("choice", "rate", "choice_coordination"),
  estimationInit = NULL,
  preprocessingInit = NULL,
  preprocessingOnly = FALSE,
  progress = getOption("progress"),
  verbose = getOption("verbose")
)
```

**Arguments**

- `x`: a formula that defines at the left-hand side the dependent network (see `defineDependentEvents()`) and at the right-hand side the effects and the variables for which the effects are expected to occur (see `vignette("goldfishEffects")`).
- `model`: a character string defining the model type. Current options include "DyNAM", "DyNAMi" or "REM".
  - **DyNAM** Dynamic Network Actor Models (Stadtfeld, Hollway and Block, 2017 and Stadtfeld and Block, 2017)
  - **DyNAMi** Dynamic Network Actor Models for interactions (Hoffman et al., 2020)
  - **REM** Relational Event Model (Butts, 2008)
- `subModel`: a character string defining the submodel type. Current options include "choice", "rate" or "choice_coordination"
  - **choice** a multinomial receiver choice model model = "DyNAM" (Stadtfeld and Block, 2017), or the general Relational event model model = "REM" (Butts, 2008). A multinomial group choice model model = "DyNAMi" (Hoffman et al., 2020)
  - **choice_coordination** a multinomial-multinomial model for coordination ties model = "DyNAM" (Stadtfeld, Hollway and Block, 2017)
  - **rate** A individual activity rates model model = "DyNAM" (Stadtfeld and Block, 2017). Two rate models, one for individuals joining groups and one for individuals leaving groups, jointly estimated model = "DyNAMi" (Hoffman et al., 2020)
estimationInit a list containing lower level technical parameters for estimation. It may contain:

- **initialParameters** a numeric vector. It includes initial parameters of the estimation. Default is set to NULL.
- **fixedParameters** a numeric vector. It specifies which component of the coefficient parameters (intercept included) is fixed and the value it takes during estimation, e.g., if the vector is \(c(2, \text{NA})\) then the first component of the parameter is fixed to 2 during the estimation process. Default is set to NULL, i.e. all parameters are estimated. Note that it must be consistent with initialParameters.
- **maxIterations** maximum number of iterations of the Gauss/Fisher scoring method for the estimation. Default is set to 20.
- **maxScoreStopCriterion** maximum absolute score criteria for successful convergence. Default value is 0.001
- **initialDamping** a numeric vector used to declare the initial damping factor for each parameter. It controls the size of the update step during the iterative estimation process. The default is set to 30 when the formula has windowed effects or 10 in another case, see vignette("goldfishEffects").
- **dampingIncreaseFactor** a numeric value. It controls the factor that increases the damping of the parameters when improvements in the estimation are found.
- **dampingDecreaseFactor** a numeric value. Controls the factor that decreases the damping of the parameters when no improvements in the estimation are found.
- **returnIntervalLogL** a logical value. Whether to keep the log-likelihood of each event from the final iteration of the Gauss/Fisher estimation method.
- **engine** a string indicating the estimation engine to be used. Current options include "default", "default_c", and "gather_compute". The default value is "default", it is an estimation routine implemented in pure R code. "default_c" uses a C implementation of the "default" routine. "gather_compute" uses a C implementation with a different data structure that reduces the time but it can increase the memory usage.
- **startTime** a numerical value or a date-time character with the same time-zone formatting as the times in event that indicates the starting time to be considered during estimation. Note: it is only use during preprocessing
- **endTime** a numerical value or a date-time character with the same time-zone formatting as the times in event that indicates the end time to be considered during estimation. Note: it is only use during preprocessing
- **opportunitiesList** a list containing for each dependent event the list of available nodes for the choice model, this list should be the same length as the dependent events list (ONLY for choice models).

preprocessingInit a preprocessed.goldfish object computed for the current formula, allows skipping the preprocessing step.

preprocessingOnly logical indicating whether only preprocessed statistics should be returned rather than a result.goldfish object with the estimated coefficients.
progress logical indicating whether should print a minimal output to the console of the progress of the preprocessing and estimation processes.

verbose logical indicating whether should print very detailed intermediate results of the iterative Newton-Raphson procedure; slows down the routine significantly.

Details

Missing data is imputed during the preprocessing stage. For network data missing values are replaced by a zero value, it means that is assuming a not tie/event explicitly. For attributes missing values are replaced by the mean value, if missing values are presented during events updates they are replace by the mean of the attribute in that moment of time.

Value

returns an object of \texttt{class()} "result.goldfish" when \texttt{preprocessingOnly = FALSE} or a preprocessed statistics object of class "preprocessed.goldfish" when \texttt{preprocessingOnly = TRUE}.

An object of class "result.goldfish" is a list including:

- \texttt{parameters} a numeric vector with the coefficients estimates.
- \texttt{standardErrors} a numeric vector with the standard errors of the coefficients estimates.
- \texttt{logLikelihood} the log-likelihood of the estimated model
- \texttt{finalScore} a vector with the final score reach by the parameters during estimation.
- \texttt{finalInformationMatrix} a matrix with the final values of the negative Fisher information matrix. The inverse of this matrix gives the variance-covariance matrix for the parameters estimates.
- \texttt{convergence} a list with two elements. The first element (\texttt{isConverged}) is a logical value that indicates the convergence of the model. The second element (\texttt{maxAbsScore}) reports the final maximum absolute score in the final iteration.
- \texttt{nIterations} an integer with the total number of iterations performed during the estimation process.
- \texttt{nEvents} an integer reporting the number of events considered in the model.
- \texttt{names} a matrix with a description of the effects used for model fitting. It includes the name of the object used to calculate the effects and additional parameter description.
- \texttt{formula} a formula with the information of the model fitted.
- \texttt{model} a character value of the model type.
- \texttt{subModel} a character value of the subModel type.
- \texttt{rightCensored} a logical value indicating if the estimation process considered right-censored events. Only it is considered for \texttt{DyNAM-rate (model = "DyNAM" and subModel = "rate")} or \texttt{REM (model = "REM")} models, and when the model includes the intercept.
DyNAM

The actor-oriented models that the goldfish package implements have been called Dynamic Network Actor Models (DyNAMs). The model is a two-step process. In the first step, the waiting time until an actor \( i \) initiates the next relational event is modeled (model = "DyNAM" and subModel = "rate") by an exponential distribution depending on the actor activity rate. In the second step, the conditional probability of \( i \) choosing \( j \) as the event receiver is modeled (model = "DyNAM" and subModel = "choice") by a multinomial probability distribution with a linear predictor. These two-steps are assumed to be conditionally independent given the process state (Stadtfeld, 2012), due to this assumption is possible to estimate these components by different calls of the estimate function.

Waiting times

When DyNAM-rate (model = "DyNAM" and subModel = "rate") model is used to estimate the first step component of the process, or the REM model = "REM" model is used. It is important to add a time intercept to model the waiting times between events, in this way the algorithm considers the right-censored intervals in the estimation process.

In the case that the intercept is not included in the formula. The model reflects the likelihood of an event being the next in the sequence. This specification is useful for scenarios where the researcher doesn’t have access to the exact interevent times. For this ordinal case the likelihood of an event is merely a multinomial probability (Butts, 2008).

References


See Also

defineDependentEvents(), defineGlobalAttribute(), defineNetwork(), defineNodes(), linkEvents()

Examples

```r
# A multinomial receiver choice model
data("Social_Evolution")
callNetwork <- defineNetwork(nodes = actors, directed = TRUE)
callNetwork <- linkEvents(x = callNetwork, changeEvent = calls, 
                           nodes = actors)
callsDependent <- defineDependentEvents(events = calls, nodes = actors, 
                                          defaultNetwork = callNetwork)
```
mod01 <- estimate(callsDependent ~ inertia + recip + trans,
                   model = "DyNAM", subModel = "choice",
                   estimationInit = list(engine = "default_c"))
summary(mod01)

# A individual activity rates model
mod02 <- estimate(callsDependent ~ 1 + nodeTrans + indeg + outdeg,
                   model = "DyNAM", subModel = "rate",
                   estimationInit = list(engine = "default_c"))
summary(mod02)

# A multinomial-multinomial choice model for coordination ties
data("Fisheries_Treaties_6070")
states <- defineNodes(states)
states <- linkEvents(states, sovchanges, attribute = "present")
states <- linkEvents(states, regchanges, attribute = "regime")
states <- linkEvents(states, gdpchanges, attribute = "gdp")
bilatnet <- defineNetwork(bilatnet, nodes = states, directed = FALSE)
bilatnet <- linkEvents(bilatnet, bilatchanges, nodes = states)
contignet <- defineNetwork(contignet, nodes = states, directed = FALSE)
contignet <- linkEvents(contignet, contigchanges, nodes = states)
createBilat <- defineDependentEvents(
    events = bilatchanges[bilatchanges$increment == 1, ],
    nodes = states, defaultNetwork = bilatnet)
partnerModel <- estimate(
    createBilat ~
    inertia(bilatnet) +
    indeg(bilatnet, ignoreRep = TRUE) +
    trans(bilatnet, ignoreRep = TRUE) +
    tie(contignet) +
    alter(states$regime) +
    diff(states$regime) +
    alter(states$gdp) +
    diff(states$gdp),
    model = "DyNAM", subModel = "choice_coordination",
    estimationInit = list(initialDamping = 40, maxIterations = 30))
summary(partnerModel)
Description

Provide diagnostic functions for an object of class result.goldfish. outliers helps to identify outliers events. changepoints helps to identify where a change point in the events sequence is presented using the log-likelihood.

Usage

```r
examineOutliers(
  x,
  method = c("Hampel", "IQR", "Top"),
  parameter = 3,
  window = NULL
)
```

```r
examineChangepoints(
  x,
  moment = c("mean", "variance"),
  method = c("PELT", "AMOC", "BinSeg"),
  window = NULL,
  ...
)
```

Arguments

- **x**: an object of class result.goldfish output from an estimate call.
- **method**: Choice of "AMOC", "PELT" or "BinSeg". For a detail description see cpt.mean or cpt.var. The default value is "PELT".
- **parameter**: An integer that represents the number of absolute outliers to identify, the threshold for the Hampel filter, i.e. parameter * MAD, or the threshold beyond the interquartile range halved, i.e. parameter/2 * IQR.
- **window**: The window half-width for the Hampel filter. By default it is half the width of the event sequence.
- **moment**: character argument to choose between "mean" or "variance". See section Change point for details.
- **...**: additional arguments to be passed to the functions in the changepoint package.

Value

NULL if neither outliers nor change points are identified. An object of class ggplot object from a call of ggplot2::ggplot(). It can be modified using the ggplot2 syntax.

Outliers

examineOutliers creates a plot with the log-likelihood of the events in the y-axis and the event index in the x-axis, identifying observations with labels indicating the sender and recipient.
Change point

The parameter moment controls which method from the package changepoint is used:

"mean" It uses the cpt.mean function to investigate optimal positioning and (potentially) number of change points for the log-likelihood of the events in mean.

"variance" It uses the cpt.var function to investigate optimal positioning and (potentially) number of change points for the log-likelihood of the events in variance.

The function call creates a plot with the log-likelihood of the events in the y-axis and the event index in the x-axis, highlighting the change point sections identified by the method.

Examples

```r
# A multinomial receiver choice model
data("Social_Evolution")
callNetwork <- defineNetwork(nodes = actors, directed = TRUE)
callNetwork <- linkEvents(
  x = callNetwork, changeEvent = calls,
  nodes = actors
)
callsDependent <- defineDependentEvents(
  events = calls, nodes = actors,
  defaultNetwork = callNetwork
)
mod01 <- estimate(callsDependent ~ inertia + recip + trans,
                   model = "DyNAM", subModel = "choice",
                   estimationInit = list(returnIntervalLogL = TRUE,
                                          engine = "default_c")
)
examineOutliers(mod01)
examineChangepoints(mod01)
```

Description

An abbreviated version of the international fisheries agreements dataset, including only bilateral agreements, fewer variables, and ranging only between 1960 and 1970 inclusive. This data set is only meant for testing, and not for inference. It provides an example of an undirected, weighted (by integer/increment) network, with composition change and both monadic and dyadic covariates. Monadic variables include the dates states gain or lose sovereign status, their polity score, and their GDP. Dyadic variables include bilateral fisheries agreements between states, and states’ contiguity with one another over time.
Usage

```r
data(Fisheries_Treaties_6070)

bilatchanges
bilatnet
contigchanges
contignet
gdpchanges
regchanges
sovchanges
states
```

Format

The data includes several dataframes: states (154 rows, 4 columns, monadic), sovchanges (62 rows, 3 columns, monadic), regchanges (145 rows, 3 columns, monadic), gdpchanges (979 rows, 3 columns, monadic), bilatchanges (77 rows, 4 columns, dyadic), contigchanges (139 rows, 4 columns, dyadic). See below for variables and formats.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>states$label</td>
<td>Node identifier labels</td>
<td>character</td>
</tr>
<tr>
<td>states$present</td>
<td>Node present in dataset</td>
<td>boolean</td>
</tr>
<tr>
<td>states$regime</td>
<td>Placeholder for regime variable</td>
<td>numeric (NA)</td>
</tr>
<tr>
<td>states$gdp</td>
<td>Placeholder for GDP variable</td>
<td>numeric (NA)</td>
</tr>
<tr>
<td>sovchanges$time</td>
<td>Date of state sovereignty update</td>
<td>POSIXct</td>
</tr>
<tr>
<td>sovchanges$node</td>
<td>Node for state sovereignty update</td>
<td>integer</td>
</tr>
<tr>
<td>sovchanges$replace</td>
<td>State sovereignty update</td>
<td>boolean</td>
</tr>
<tr>
<td>regchanges$time</td>
<td>Date of regime update</td>
<td>POSIXct</td>
</tr>
<tr>
<td>regchanges$node</td>
<td>Node for regime update</td>
<td>integer</td>
</tr>
<tr>
<td>regchanges$replace</td>
<td>Regime update</td>
<td>integer (-10–10)</td>
</tr>
<tr>
<td>gdpchanges$time</td>
<td>Date of GDP update</td>
<td>POSIXct</td>
</tr>
<tr>
<td>gdpchanges$node</td>
<td>Node for GDP update</td>
<td>integer</td>
</tr>
<tr>
<td>gdpchanges$replace</td>
<td>GDP update</td>
<td>numeric</td>
</tr>
<tr>
<td>bilatchanges$time</td>
<td>Date of bilateral change</td>
<td>POSIXct</td>
</tr>
<tr>
<td>bilatchanges$sender</td>
<td>First bilateral change node</td>
<td>integer</td>
</tr>
<tr>
<td>bilatchanges$receiver</td>
<td>Second bilateral change node</td>
<td>integer</td>
</tr>
<tr>
<td>bilatchanges$increment</td>
<td>Create or dissolve action</td>
<td>numeric (-1 or 1)</td>
</tr>
<tr>
<td>contigchanges$time</td>
<td>Date of contiguity change</td>
<td>POSIXct</td>
</tr>
<tr>
<td>contigchanges$sender</td>
<td>First contiguity change node</td>
<td>integer</td>
</tr>
<tr>
<td>contigchanges$receiver</td>
<td>Second contiguity change node</td>
<td>integer</td>
</tr>
<tr>
<td>contigchanges$replace</td>
<td>New contiguity value</td>
<td>numeric</td>
</tr>
</tbody>
</table>
An object of class \texttt{data.frame} with 77 rows and 4 columns. 
An object of class \texttt{matrix} (inherits from \texttt{array}) with 154 rows and 154 columns. 
An object of class \texttt{data.frame} with 139 rows and 4 columns. 
An object of class \texttt{matrix} (inherits from \texttt{array}) with 154 rows and 154 columns. 
An object of class \texttt{data.frame} with 979 rows and 3 columns. 
An object of class \texttt{data.frame} with 145 rows and 3 columns. 
An object of class \texttt{data.frame} with 62 rows and 3 columns. 
An object of class \texttt{data.frame} with 154 rows and 4 columns. 

Reference 


### \texttt{linkEvents} 

#### Description 

Link dynamic events to a nodeset or a network. 

#### Usage 

\begin{verbatim}
linkEvents(x, ...) 
## S3 method for class 'nodes.goldfish'
linkEvents(x, changeEvents, attribute, ...) 
## S3 method for class 'network.goldfish'
linkEvents(x, changeEvents, nodes = NULL, nodes2 = NULL, ...) 
## Default S3 method: 
linkEvents(x, ...) 
\end{verbatim}

#### Arguments 

- \texttt{x} 
  Either a nodeset (\texttt{nodes.goldfish} object) or a network (\texttt{network.goldfish} object) 
- \texttt{...} 
  additional arguments to be passed to the method. 
- \texttt{changeEvents} 
  The name of a data frame that represents a valid events list.
**linkEvents**

**attribute**  
a character vector indicating the names of the attributes that should be updated by the specified events (ONLY if the object is a nodeset).

**nodes**  
a nodeset (data.frame or nodes.goldfish object) related to the network (ONLY if x is a network).

**nodes2**  
an optional nodeset (data.frame or nodes.goldfish object) related to the network (ONLY if x is a network).

**Details**

The data frame that contains the events must contain variables with specific names depending if they refer to dynamic attributes or dynamic networks.

For dynamic networks stored in object of class network.goldfish the changeEvents data frame must contain the following variables:

**time**  
numeric or POSIXct (see base::as.Date()) variable containing the time-stamps when the event happen.

**sender**  
character variable indicating the label of the sender of the event.

**receiver**  
character variable indicating the label of the receiver of the event.

See the bilatchanges and contigchanges data frames in the Fisheries_Treaties_6070 datasets for examples of event data frames that relate with dynamic networks.

For dynamic attributes stored in object of class nodes.goldfish the changeEvents data frame must contain the following variables:

**time**  
numeric or POSIXct (see base::as.Date()) variable containing the time-stamps when the event happen.

**label**  
character variable indicating the label of the node for which the attribute changes.

See sovchanges, regchanges and gdpchanges data frames in the Fisheries_Treaties_6070 datasets for examples of event data frames that relate with dynamic attributes.

For both cases an additional variable indicates the change in value of either the ties or attributes. The class of this variable must be the same as the tie value or attribute value that will be updated, i.e., when the present variable is dynamic the updating values must be logical (see defineNodes()) for a description of this variable. There are two possibilities on how to specify those changes but only one can be used at a time:

**increment**  
with a numerical value that represent the increment (when it’s positive value) or the decrement (when it’s a negative value) of the dynamic element from their past value (with respect to the time value).

In the Social_Evolution dataset the calls data frame contains the calling events between students where the increment represent a new call. With every new call the dyad (sender-receiver) increase the count of calls that had happen in the past.

**replace**  
contains the value that would replace at point-time time the attribute or tie value. It is usually the way to represent changes in node attributes.

In the Fisheries_Treaties_6070 dataset the sovchanges, regchanges and gdpchanges data frames are examples where the replace variable is used to specify attribute changes and their class match with the variable in the node set.
Dynamic network attributes can be also defined using the `replace` variable. The `contigchanges` data frame in the `Fisheries_Treaties_6070` dataset, and `friendship` data frame in the `Social_Evolution` are examples of this.

**Value**

an object with the same class as the object `x`. For objects of class `network.goldfish` the attribute `events` with the name of the event data frame passed through with the argument `changeEvents`. For objects of class `nodes.goldfish` attributes `events` and `dynamicAttribute` are modified with name of the objects passed through with the arguments `changeEvents` and `attribute` respectively.

**See Also**

`defineNodes()`, `defineNetwork()`

**Examples**

```r
actors <- data.frame(
  actor = 1:5, label = paste("Actor", 1:5),
  present = TRUE, gender = sample.int(2, 5, replace = TRUE)
)
actors <- defineNodes(nodes = actors)
callNetwork <- defineNetwork(nodes = actors)

# Link events to a nodeset
compositionChangeEvents <- data.frame(
  time = c(14, 60),
  node = "Actor 4",
  replace = c(FALSE, TRUE)
)
actorsnew <- linkEvents(
  x = actors, attribute = "present", changeEvents = compositionChangeEvents
)

# Link events to a Network
calls <- data.frame(
  time = c(12, 27, 45, 56, 66, 68, 87),
  sender = paste("Actor", c(1, 3, 5, 2, 3, 4, 2)),
  receiver = paste("Actor", c(4, 2, 3, 5, 1, 2, 5)),
  increment = rep(1, 7)
)
callNetwork <- linkEvents(
  x = callNetwork, changeEvent = calls, nodes = actors
)
```

---

*logLik.result.goldfish*

*Extract log-likelihood from a fitted model object*
logLik.result.goldfish

Description

This function extract the log-likelihood from the output of a `estimate` call. The extracted log-likelihood correspond to the value in the last iteration of the `estimate` call, users should check convergence of the Gauss/Fisher scoring method before using the log-likelihood statistic to compare models.

Usage

## S3 method for class 'result.goldfish'
logLik(object, ..., avgPerEvent = FALSE)

Arguments

- **object**: an object of class `result.goldfish` output from an `estimate` call with a fitted model.
- **...**: additional arguments to be passed.
- **avgPerEvent**: a logical value indicating whether the average likelihood per event should be calculated.

Details

Users might use `stats::AIC()` and `stats::BIC()` to compute the Information Criteria from one or several fitted model objects. An information criterion could be used to compare models with respect to their predictive power.

Alternatively, `lmtest::lrtest()` can be used to compare models via asymptotic likelihood ratio tests. The test is designed to compare nested models. i.e., models where the model specification of one contains a subset of the predictor variables that define the other.

Value

Returns an object of class `logLik` when `avgPerEvent = FALSE`. This is a number with the extracted log-likelihood from the fitted model, and with the following attributes:

- **df**: degrees of freedom with the number of estimated parameters in the model
- **nobs**: the number of observations used in estimation. In general, it corresponds to the number of dependent events used in estimation. For a `subModel = "rate"` or `model = "REM"` with intercept, it corresponds to the number of dependent events plus right-censored events due to exogenous or endogenous changes.

When `avgPerEvent = TRUE`, the function returns a number with the average log-likelihood per event. The total number of events depends on the presence of right-censored events in a similar way that the attribute nobs is computed when `avgPerEvent = FALSE`. 
Methods for goldfish objects.

Description

Printing functions for goldfish objects.

Usage

```r
## S3 method for class 'result.goldfish'
print(
  x,
  ..., 
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  complete = FALSE
)
```

```r
## S3 method for class 'summary.result.goldfish'
print(
  x,
  ..., 
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  complete = FALSE
)
```

```r
## S3 method for class 'nodes.goldfish'
print(x, ..., full = FALSE, n = 6)
```

```r
## S3 method for class 'network.goldfish'
print(x, ..., full = FALSE, n = 6L)
```

```r
## S3 method for class 'dependent.goldfish'
print(x, ..., full = FALSE, n = 6)
```

```r
## S3 method for class 'preprocessed.goldfish'
print(x, ..., width = getOption("width"))
```

Arguments

- **x**: an object of class `result.goldfish`, `summarize.result.goldfish`, `nodes.goldfish`, `network.goldfish`, `dependent.goldfish`, or `preprocessed.goldfish`.
- **...**: further arguments to be passed to the respective default method.
- **digits**: minimal number of significant digits, see `print.default()`.
width controls the maximum number of columns on a line used in printing summary.result.goldfish and preprocessed.goldfish, see print.default().

complete logical. Indicates whether the parameter coefficients of effects fixed during estimation using fixedParameters should be printed. The default value is FALSE. 

Note: applies for objects of class result.goldfish and summary.result.goldfish.

full logical. Indicates whether the complete matrix/data.frame should be printed. The default value FALSE.

n number of rows for data.frame, and rows and columns for matrix to be printed.

Value

Not value, called for printing side effect.

For objects of class result.goldfish and summary.result.goldfish print the estimated coefficients when complete = FALSE, otherwise it includes also the fixed coefficients. For summary.result.goldfish print:

Effect details:
a table with additional information of the effects. The information corresponds to the values of the effects arguments when they are modified and if they where fixed during estimation, see vignette("goldfishEffects") for the complete list of arguments, and estimate() on how to fix coefficients during estimation.

Coefficients: a table with the estimated coefficients, their approximate standard error obtain from the inverse of the negative Fisher information matrix, z-value and the p-value of the univariate two-tailed Wald test to test the hypothesis that the parameter is 0.

Convergence and Information Criteria:
Information about the convergence of the iterative Newton-Raphson procedure and the score value in the last iteration. Information criteria as the AIC, BIC and the AIC corrected for small sample size AICc are reported.

Model and subModel:
the values set during estimation.

For objects of class nodes.goldfish print information of the total number of nodes in the object, the number of nodes present at the beginning of preprocessing, a table with the linked attributes with their respective events data frame and a printing of the first rows in the nodes data frame. See defineNodes().

For objects of class network.goldfish print information of the dimensions of the network, number of ties presented at the beginning of the preprocessing, the nodes data frames linked to it, information about their definition as a one-mode and directed network, linked events data frame to it and a printing of the first rows and columns in the array. See defineNetwork().

For objects of class dependent.goldfish print information of the total number of events in the object, linked nodes set(s), linked default network and a printing of the first rows in the events data frame. See defineDependentEvents().
**RFID Validity Study**

**Description**

Dataset collected at ETH Zürich by Timon Elmer and colleagues in order to test the accuracy of Radio Frequency Identification (RFID) badges for measuring social interactions. Social interactions of 11 individuals (from the university staff) were recorded with RFID badges in an informal setting. They were then compared to the interactions observed by two confederates who watched the video recording of the event. The RFID data went through the data processing procedure detailed in the original article. See Elmer et al, 2019 for more details, and the OSF platform for all details on the dataset.

**Usage**

```r
data(RFID_Validity_Study)

rfid

video

known.before

participants
```

**Format**

3 dataframes:

- **participants** (11 rows, 7 columns): attributes of the experiment’s participants
- **rfid** (1011 rows, 4 columns): dyadic interactions detected by the RFID badges (after data processing)
- **video** (219 rows, 4 columns): dyadic interactions detected by the video rating and one network:

**known.before** (11 rows, 11 columns): network of previous acquaintances

See below for variables and formats.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>participants$actor</td>
<td>Identifier of the actor</td>
<td>integer</td>
</tr>
<tr>
<td>participants$label</td>
<td>(Anonymized) name</td>
<td>Factor</td>
</tr>
<tr>
<td>participants$present</td>
<td>Presence of the actor (all actors are present)</td>
<td>logical</td>
</tr>
</tbody>
</table>
Description

An abbreviated version of the MIT Reality Commons Social Evolution dataset, spanning a reduced time period and with fewer variables. Dyadic variables include binary friendships at time of survey, and time-stamped phone call occurrences. Individual variables include the floor of the dormitory on which the student resides, and the grade type of each student including freshmen, sophomore, junior, senior, or graduate tutors.

Usage

data(Social_Evolution)

actors

calls

friendship

Source

https://osf.io/rrhxe/

References

Format

3 dataframes: actors (84 rows, 4 columns), calls (439 rows, 4 columns), friendship (766 rows, 4 columns). See below for variables and formats.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>actors$label</td>
<td>Actor identifier labels</td>
<td>character</td>
</tr>
<tr>
<td>actors$present</td>
<td>Actor present in dataset</td>
<td>boolean</td>
</tr>
<tr>
<td>actors$floor</td>
<td>Floor of residence actor lives on</td>
<td>numeric (1-9)</td>
</tr>
<tr>
<td>actors$gradeType</td>
<td>Degree level</td>
<td>numeric (1-5)</td>
</tr>
<tr>
<td>calls$time</td>
<td>Time and date of call</td>
<td>numeric from POSIXct</td>
</tr>
<tr>
<td>calls$sender</td>
<td>Initiator of phone call</td>
<td>character</td>
</tr>
<tr>
<td>calls$receiver</td>
<td>Recipient of phone call</td>
<td>character</td>
</tr>
<tr>
<td>calls$increment</td>
<td>Indicates call number increment (all 1s)</td>
<td>numeric (1)</td>
</tr>
<tr>
<td>friendship$time</td>
<td>Time and date of friend nomination</td>
<td>numeric from POSIXct</td>
</tr>
<tr>
<td>friendship$sender</td>
<td>Nominator of friendship</td>
<td>character</td>
</tr>
<tr>
<td>friendship$receiver</td>
<td>Nominee of friendship</td>
<td>character</td>
</tr>
<tr>
<td>friendship$replace</td>
<td>Indicates friendship value at $time</td>
<td>numeric</td>
</tr>
</tbody>
</table>

An object of class data.frame with 84 rows and 4 columns.
An object of class data.frame with 439 rows and 4 columns.
An object of class data.frame with 766 rows and 4 columns.

Source

http://realitycommons.media.mit.edu/socialevolution.html

References


update-method

Methods to update a nodes or network object

Description

Methods to create a data frame from an object of class nodes.goldfish (see defineNodes()) or a matrix from an object of class network.goldfish (see defineNetwork()) with the attributes or the network ties updated according with the events linked to the object using the linkEvents() function.
Usage

## S3 method for class 'nodes.goldfish'
as.data.frame(x, ..., time = -Inf, startTime = -Inf)

## S3 method for class 'network.goldfish'
as.matrix(x, ..., time = -Inf, startTime = -Inf)

Arguments

- **x**: an object of class nodes.goldfish for as.data.frame() method or network.goldfish for as.matrix() method.
- **...**: Not further arguments are required.
- **time**: a numeric value or a calendar date value (see as.Date()) to update the state of the object x until this time value (event time < time).
- **startTime**: a numeric as.Date format value; prior events are disregarded.

Value

The respective object updated accordingly to the events link to it. For nodes.goldfish object the attributes are updated according to the events linked to them. For network.goldfish object the network ties are updated according to the events linked to it.

See Also

defineNetwork(), defineNodes(), linkEvents()

Examples

data("Fisheries_Treaties_6070")
states <- defineNodes(states)
states <- linkEvents(states, sovchanges, attribute = "present")
states <- linkEvents(states, regchanges, attribute = "regime")
states <- linkEvents(states, gdpchanges, attribute = "gdp")

bilatnet <- defineNetwork(bilatnet, nodes = states, directed = FALSE)
bilatnet <- linkEvents(bilatnet, bilatchanges, nodes = states)

updateStates <- as.data.frame(states,
    time = as.numeric(as.POSIXct("1965-12-31")))

updateNet <- as.matrix(bilatnet, time = as.numeric(as.POSIXct("1965-12-31")))
Index

* RFID
  RFID_Viability_Study, 24
* datasets
  Fisheries_Treaties_6070, 16
  RFID_Viability_Study, 24
  Social_Evolution, 25
* dynamic
  Fisheries_Treaties_6070, 16
* evolution
  Social_Evolution, 25
* fisheries
  Fisheries_Treaties_6070, 16
* interactions
  RFID_Viability_Study, 24
* network
  Fisheries_Treaties_6070, 16
  Social_Evolution, 25
* political
  Fisheries_Treaties_6070, 16
* social
  RFID_Viability_Study, 24
  Social_Evolution, 25
* states
  Fisheries_Treaties_6070, 16
* study
  RFID_Viability_Study, 24
* validity
  RFID_Viability_Study, 24
actors (Social_Evolution), 25
as.data.frame.nodes.goldfish
  (update-method), 26
as.Date(), 27
as.matrix.network.goldfish
  (update-method), 26
base::as.Date(), 19
bilatchanges (Fisheries_Treaties_6070), 16
bilatnet (Fisheries_Treaties_6070), 16
calls (Social_Evolution), 25
change point, 15, 16
class(), 12
contigchanges
  (Fisheries_Treaties_6070), 16
contignet (Fisheries_Treaties_6070), 16
cpt.mean, 15, 16
cpt.var, 15, 16
defineDependentEvents, 3
defineDependentEvents(), 10, 13, 23
defineGlobalAttribute, 5
defineGlobalAttribute(), 13
defineGroups_interaction, 5
defineNetwork, 7
defineNetwork(), 4, 9, 13, 20, 23, 26, 27
defineNodes, 8
defineNodes(), 4, 7, 8, 13, 19, 20, 23, 26, 27
defineNodes_interaction, 16
defineNodes_interaction(), 13, 20, 23, 26, 27
defineNodes_network, 22
defineNodes_network(), 4, 9, 13, 20, 23, 26, 27
developGlobalAttribute, 5
developGlobalAttribute(), 13
developGroups_interaction, 5
developNetwork, 7
developNetwork(), 4, 9, 13, 20, 23, 26, 27
developNodes, 8
developNodes(), 4, 7, 8, 13, 19, 20, 23, 26, 27
developNodes_interaction, 16
developNodes_interaction(), 13, 20, 23, 26, 27
developNodes_network, 22
developNodes_network(), 4, 9, 13, 20, 23, 26, 27
determineEvents, 18
determineEvents(), 10, 13, 23
displayGlobalAttribute, 5
displayGlobalAttribute(), 13
displayGroups_interaction, 5
displayNetwork, 7
displayNetwork(), 4, 9, 13, 20, 23, 26, 27
displayNodes, 8
displayNodes(), 4, 7, 8, 13, 19, 20, 23, 26, 27
participants (RFID_Viability_Study), 24
print-method, 22
print.default(), 22, 23
print.dependent.goldfish
 (print-method), 22
print.network.goldfish (print-method), 22
print.nodes.goldfish (print-method), 22
print.preprocessed.goldfish
 (print-method), 22
print.result.goldfish (print-method), 22
print.summary.result.goldfish
 (print-method), 22
regchanges (Fisheries_Treaties_6070), 16
rfid (RFID_Viability_Study), 24
RFID_Viability_Study, 24
Social_Evolution, 19, 20, 25
sovchanges (Fisheries_Treaties_6070), 16
states (Fisheries_Treaties_6070), 16
stats::AIC(), 21
stats::BIC(), 21

tbl_df, 9
update-method, 26
video (RFID_Viability_Study), 24