Package ‘remify’

May 16, 2024

Type Package

Title Processing and Transforming Relational Event History Data

Version 3.2.6

Date 2024-05-15

Maintainer Giuseppe Arena <g.arena@tilburguniversity.edu>

Description Efficiently processes relational event history data and transforms them into formats suitable for other packages. The primary objective of this package is to convert event history data into a format that integrates with the packages in 'remverse' and is compatible with various analytical tools (e.g., computing network statistics, estimating tie-oriented or actor-oriented social network models). Second, it can also transform the data into formats compatible with other packages out of 'remverse'. The package processes the data for two types of temporal social network models: tie-oriented modeling framework (Butts, C., 2008, <doi:10.1111/j.1467-9531.2008.00203.x>) and actor-oriented modeling framework (Stadtfeld, C., & Block, P., 2017, <doi:10.15195/v4.a14>).

License MIT + file LICENSE

URL https://github.com/TilburgNetworkGroup/remify

BugReports https://github.com/TilburgNetworkGroup/remify/issues

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.8.3), igraph (>= 1.4.3)

Suggests knitr, rmarkdown, tinytest

LinkingTo Rcpp, RcppArmadillo,

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation yes

Author Giuseppe Arena [aut, cre] (<https://orcid.org/0000-0001-5204-3326>),
Rumana Lakdawala [ctb],
Marlyne Meijerink-Bosman [ctb],
Diana Karimova [ctb],

Fabio Generoso Vieira [ctb],
Mahdi Shafiee Kamalabad [ctb],
Roger Leenders [ctb],
Joris Mulder [ctb]

Repository CRAN

Date/Publication 2024-05-16 02:30:03 UTC

R topics documented:

`dim.remify` *dim.remify*

Description

A function that returns the dimension of the temporal network.

Usage

```
## S3 method for class 'remify'
dim(x)
```

Arguments

`x`	a `remify` object.

Value

vector of dimensions of the processed event sequence.

Examples

```
# processing the random network 'randomREHsmall'
library(remify)
data(randomREHsmall)
reh <- remify(edgelist = randomREHsmall$edgelist,
              model = "tie")

# dimensions of the processed 'remify' object
dim(reh)
```

`getActorID` *getActorID*

Description

A function that given a vector of actor names returns the corresponding vector of ID's.

Usage

```
getActorID(x, actorName = NULL)

## S3 method for class 'remify'
getActorID(x, actorName = NULL)
```

Arguments

x	a `remify` object.
actorName	a vector of actor names. The same names in the input edgelist.

Value

actor ID as integer value.

Methods (by class)

- `getActorID(remify)`: return actor's ID from actor's name

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# find actor ID from the actor name
getActorID(x = reh, actorName = c("Francesca","Kayla"))
```

---

getActorName *getActorName*

Description

A function that given a vector of actor ID's returns the corresponding vector of actor (input) names.

Usage

```
getActorName(x, actorID = NULL)

## S3 method for class 'remify'
getActorName(x, actorID = NULL)
```

Arguments

x	a remify object.
actorID	a vector of actor ID's. The ID value can range between 1 and N (number of actors in the network).

Value

character vector of actors' names.

Methods (by class)

- `getActorName(remify)`: return actor's name from actor's ID

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# find actor name from actor ID
getActorName(x = reh, actorID = c(1,2,8,12))
```

getDyad *getDyad*

Description

A function that given a vector of one or more dyad ID's returns the corresponding dyad composition of "actor1", "actor2" and "type" (if event types are present). The ID's to supply must range between 1 and D (largest risk set size).

Usage

```
getDyad(x, dyadID, active = FALSE)

## S3 method for class 'remify'
getDyad(x, dyadID, active = FALSE)
```

Arguments

`x`	a `remify` object.
`dyadID`	a vector of one or more dyad ID's, each one ranging from 1 to D (largest risk set size).
`active`	logical, whether to consider the input `dyadID` as a vector of ID's of active dyads (`active = TRUE`) or dyads from the full risk set (`active = FALSE`).

Value

a data.frame with "actor1", "actor2" and "type" names corresponding to the vector `dyadID`.

Methods (by class)

- `getDyad(remify)`: return dyad composition in actor1, actor2 and type from one (or more) dyad ID

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# find dyad composition (names of actor1, actor2 and type) from the dyad ID
getDyad(x = reh, dyadID = c(450,239,900))
```

getDyadID *getDyadID*

Description

A function that given a vector of names as to actor1, actor2 and type returns the corresponding dyad ID. The names to supply are the original input names of the edgelist before the processing via the function `remify::remify()`.

Usage

```
getDyadID(x, actor1, actor2, type)

## S3 method for class 'remify'
getDyadID(x, actor1, actor2, type)
```

Arguments

`x`	a `remify` object.
`actor1`	[character] name of actor1.
`actor2`	[character] name of actor2.
`type`	[character] name of type.

Value

dyad ID as integer value.

Methods (by class)

- `getDyadID(remify)`: return dyad's ID from dyad's composition

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# find dyad ID from dyad composition (names of actor1, actor2 and type)
getDyadID(x = reh, actor1 = "Francesca", actor2 = "Kayla", type = "conflict")
```

`getRiskset` *getRiskset*

Description

This function returns the processed risk set changes specified by the input ‘omit_dyad‘. In such a matrix: value 1 refers to the dyads in the risk set, and 0 otherwise (dyads excluded from the risk set). All the possible risk set modifications are described by row, and the columns identify the dyads. Note: This matrix is the output given by processing the input ‘omit_dyad‘, and the number of rows might be equal to or higher than the number of objects in ‘omit_dyad‘. This might happen because more than one modification of the risk set defined in the input could overlap over time with others. For more details about how the risk set is processed, see `vignette(package="remify",topic="riskset")`.

Usage

```
getRiskset(x)

## S3 method for class 'remify'
getRiskset(x)
```

Arguments

`x`	a `remify` object.

Value

list of objects describing the processed the risk set.

Methods (by class)

- `getRiskset(remify)`: manual riskset object

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# structure of the processed risk set
str(getRiskset(reh))
```

`getTypeID` *getTypeID*

Description

A function that given a vector of type names returns the corresponding vector of ID's.

Usage

```
getTypeID(x, typeName = NULL)

## S3 method for class 'remify'
getTypeID(x, typeName = NULL)
```

Arguments

`x`	a `remify` object.
`typeName`	a vector of type names. The same names in the input edgelist.

Value

type ID as integer value.

Methods (by class)

- `getTypeID(remify)`: return type's ID from type's name

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# find type ID from the type name
getTypeID(x = reh, typeName = c("conflict","cooperation"))
```

getTypeName *getTypeName*

Description

A function that given a vector of type ID's returns the corresponding vector of type (input) names.

Usage

```
getTypeName(x, typeID = NULL)

## S3 method for class 'remify'
getTypeName(x, typeID = NULL)
```

Arguments

x	a remify object.
typeID	a vector of type ID's. The ID value can range between 1 and C (number of event types in the network).

Value

character vector of types' names.

Methods (by class)

- getTypeName(remify): return type's name from type's ID

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# find type name from type ID
getTypeName(x = reh, typeID = c(1,3))
```

plot.remify *plot.remify*

Description

several plots that describe the network of relational events, both for directed and undirected relational events.

Usage

```
## S3 method for class 'remify'
plot(
  x,
  which = c(1:5),
  breaks = 15L,
  palette = "Purples",
  n_intervals = 4L,
  rev = TRUE,
  actors = attr(x, "dictionary")$actors$actorName,
  pch.degree = 20,
  igraph.edge.color = "#4daa89",
  igraph.vertex.color = "#5AAFC8",
  ...
)
```

Arguments

`x`	is a `remify` object.
`which`	one or more numbers between 1 and 5. Plots described in order: (1) distribution of the inter-event times (histogram), (2) tile plot titled 'activity plot', with in-degree and out-degree activity line plots on the sides (or total-degree on the top side if the network is undirected). Tiles' color is scaled based on the count of the directed (or undirected) dyad, (3) for directed networks two plots of normalized out-degree and in-degree (values ranging in [0,1]) over a set of `n_intervals` (evenly spaced). For undirected networks one plot of normalized total-degree over the `n_intervals` (also here values ranging in [0,1]). The normalization is calculated in each interval as the `(degree-min(degree))/(max(degree)-min(degree)))` for each actor considering minimum and maximum degree (in-, out- or total-) observed in the interval (opacity and size of the points is proportional to the normalized measure), (4) four plots: (i) number of events (# events) per time interval, (ii) proportion of observed dyads (# dyads / x$D) per time interval, (iii) and (iv) (for directed network only) proportion of active senders and receivers per time interval (calculated as # senders/ x$N and # receiver/x$N per interval), (5) two networks: (i) network of events where edges are considered undirected (edges' opacity is proportional to the counts of the undirected events, vertices' opacity is proportional to the total-degree of the actors), (ii) visualization of

	directed network (edges' opacity is proportional to the counts of the directed events, vertices' opacity is proportional to the in-degree of the actors).
`breaks`	default is `15L` and it describes the number of cells of the histogram plot for the inter-event times. It can be specified in the same way as the argument used by the function `graphics::hist()` (see ?graphics::hist for more details).
`palette`	a palette from `grDevices::hcl.pals()` (default is the `"Purples"` palette).
`n_intervals`	number of time intervals for time plots (default is `10`).
`rev`	default is TRUE (reverse order of the color specified in `palette`)
`actors`	default is the set of actors in the network (see `attr(x,"dictionary")[["actors"]]`). The user can specify a subset of actors on which to run the descriptive plots. If the set contains more than 50 actors, then the function will select the 50 most active actors from the set provided.
`pch.degree`	default is 20. Shape of the points for the degree plots (in-degree, out-degree, total-degree).
`igraph.edge.color`	color of the edges in visualization of the network with vertices and nodes. The user can specify the hex value of a color, the color name or use the functiongr`Devices::rgb()` which returns the hex value.
`igraph.vertex.color`	color of the vertices in visualization of the network with vertices and nodes. The user can specify the hex value of a color, the color name or use the function `grDevices::rgb()` which returns the hex value.
`...`	other graphical parameters

Details

Generic plot method

Value

no return value, called for plotting descriptives on the relational event history data.

`print.remify` *print.remify*

Description

print a summary of the event history.

Usage

```
## S3 method for class 'remify'
print(x, ...)
```

Arguments

`x`	a `remify` object.
`...`	further arguments.

Value

displays the same information provided by the summary method.

Examples

```
# processing the random network 'randomREHsmall'
library(remify)
data(randomREHsmall)
reh <- remify(edgelist = randomREHsmall$edgelist,
              model = "tie")

# printing a summary of the processed 'remify' object
print(reh)
```

`randomREH` *Random Relational Event History*

Description

A randomly generated sequence of relational events with 20 actors and 9915 events. Each event type is associated to one of the three following sentiments: *conflict*, *competition* and *cooperation*.

Usage

```
randomREH
```

Format

`data(randomREH)` will load a list containing following objects:

- `edgelist` a `data.frame` that contains the random sequence of events. Columns of the edgelist are:
  - `time` the timestamp indicating the time at which each event occurred;
  - `actor1` the name of the actor that generated the relational event;
  - `actor2` the name of the actor that received the relational event;
  - `type` the type of the relational event.
- `actors` names of actors interacting in the dynamic network.
- `types` names of event types observed in the network and describing the sentiment of the interaction (*conflict*, *competition* and *cooperation*).
- `origin` starting time point (`t_0`) prior to the first observed event (`t_1`), the class of this object must be the same as the one of the time column in the edgelist.
- `omit_dyad` a list where each element describes an alteration of the riskset which takes place at specific time points and for certain actors and/or types.

Examples

```
data(randomREH)

# actors names
randomREH$actors

# types names
randomREH$types

# looking into the first modification of the riskset: omit_dyad[[1]]
## the data.frame `dyad` specifies which dyads will be omitted from the riskset
## (all the dyads that expressed a `conflict` between actor won't be part of the riskset):
randomREH$omit_dyad[[1]]$dyad

## the vector `time` specifies the time points when this exclusion takes place
head(randomREH$omit_dyad[[1]]$time) # (printing out only the first 10 time points)

# run the preprocessing function reh() by supplying the loaded objects.
edgelist_reh <- remify(edgelist = randomREH$edgelist,
                    actors = randomREH$actors,
                    types = randomREH$types,
                    directed = TRUE,
                    ordinal = FALSE,
                    origin = randomREH$origin,
                    omit_dyad = randomREH$omit_dyad,
                    model = "tie")

# `edgelist_reh` is an object of class `reh`
class(edgelist_reh)

# names of objects inside `edgelist_reh`
names(edgelist_reh)
```

`randomREHsmall` *Random Relational Event History (small)*

Description

A subset from the randomly generated sequence of relational events ‘randomREH‘, with 5 actors and 586 events (without event types).

Usage

```
randomREHsmall
```

Format

`data(randomREHsmall)` will load a list containing following objects:

`edgelist` a `data.frame` that contains the random sequence of events. Columns of the edgelist are:

- `time` the timestamp indicating the time at which each event occurred;
- `actor1` the name of the actor that generated the relational event;
- `actor2` the name of the actor that received the relational event;

`actors` names of actors interacting in the dynamic network.

`origin` starting time point (`t_0`) prior to the first observed event (`t_1`), the class of this object must be the same as the one of the time column in the edgelist.

`omit_dyad` a list where each element describes an alteration of the riskset which takes place at specific time points and for certain actors and/or types.

Examples

```
data(randomREHsmall)

# actors names
randomREHsmall$actors

# types names
randomREHsmall$types


# run the preprocessing function reh() by supplying the loaded objects.
small_edgelist_reh <- remify(edgelist = randomREHsmall$edgelist,
                    actors = randomREHsmall$actors,
                    directed = TRUE,
                    ordinal = FALSE,
                    origin = randomREHsmall$origin,
                    omit_dyad = randomREHsmall$omit_dyad,
                    model = "tie")

# `small_edgelist_reh` is an object of class `reh`
class(small_edgelist_reh)

# names of objects inside `small_edgelist_reh`
names(small_edgelist_reh)
```

---

rehshape   *Transform processed remify objects to different formats*

Description

A function that transforms a `remify` object into one of the possible formats that suit external packages. The function can convert, at the moment, the data structure from an object of class `remify` to a data structure required by the function `relevent::rem()` or by the function `relevent::rem.dyad()` from the 'relevent' package (Butts, C.T. 2023).

Usage

```
rehshape(
  data,
  output_format = c("relevent-rem", "relevent-rem.dyad"),
  ncores = 1L,
  optional_arguments = NULL
)
```

Arguments

`data`	an object of class 'remify' (see function `remify::remify()`).
`output_format`	a character indicating the output format which the input data has to be converted to. It can assume two values: `"relevent-rem"` , `"relevent-rem.dyad"`. Default value is `"relevent-rem"`.
`ncores`	number of cores used to parallelize internal algorithms
`optional_arguments`	vector of arguments names from relevent::rem or relevent::rem.dyad() that the user might want to process and have in the output object of rehshape (e.g., the pre-computed structures required by relevent::rem.dyad)

Value

an object of class specified in the `output_format` argument. The output class object 'relevent-rem' contains a list of objects named after the arguments of the function `relevent::rem()`: 'eventlist' (mandatory), 'supplist' (optional), 'timing'(mandatory). The output class object 'relevent-rem.dyad' contains a list of objects named after the arguments of the function `relevent::rem.dyad()`: 'edgelist' (mandatory), 'n' (mandatory), 'ordinal'(optional).

Examples

```
# processing the random network 'randomREH'
library(remify)
data(randomREH)
reh <- remify(edgelist = randomREH$edgelist,
              model = "tie",
              riskset = "manual",
              omit_dyad = randomREH$omit_dyad)

# convert 'remify' object to output_format = "relevent-rem"
relevent_rem_obj <- rehshape(data = reh, output_format = "relevent-rem")

str(relevent_rem_obj)

# convert 'remify' object to output_format = "relevent-rem.dyad"
relevent_rem.dyad_obj <- rehshape(data = reh, output_format = "relevent-rem.dyad")

summary(relevent_rem.dyad_obj)
```

`remify` *Process a Relational Event History*

Description

A function that processes raw relational event history data and returns a S3 object of class 'remify' which is used as input in other functions inside 'remverse'.

Usage

```
remify(
  edgelist,
  directed = TRUE,
  ordinal = FALSE,
  model = c("tie", "actor"),
  actors = NULL,
  types = NULL,
  riskset = c("full", "active", "manual"),
  origin = NULL,
  omit_dyad = NULL,
  ncores = 1L
)
```

Arguments

`edgelist`	the relational event history. An object of class `data.frame` with first three columns corresponding to time, and actors forming the dyad. The first three columns will be re-named "time", "actor1", "actor2" (where, for directed networks, "actor1" corresponds to the sender and "actor2" to the receiver of the relational event). Optional columns that can be supplied are: 'type' and 'weight'. If one or both exist in `edgelist`, they have to be named accordingly.
`directed`	logical value indicating whether events are directed (`TRUE`) or undirected (`FALSE`). (default value is `TRUE`)
`ordinal`	logical value indicating whether only the order of events matters in the model (`TRUE`) or also the waiting time must be considered in the model (`FALSE`). (default value is `FALSE`)
`model`	can be "tie" or "actor" oriented modeling. This argument plays a fundamental role when `omit_dyad` is supplied. Indeed, when actor-oriented modeling, the dynamic risk set will consist of two risk sets objects (senders' and dyads' risk sets). In the tie-oriented model the function will return a dynamic risk set referred at a dyad-level.
`actors`	[*optional*] character vector of actors' names that may be observed interacting in the network. If `NULL` (default), actors' names will be taken from the input edgelist.
`types`	[*optional*] character vector of event types that may occur in the network. If `NULL` (default), types' names will be taken from the input edgelist.

`riskset`	[*optional*] character value indicating the type of risk set to process: `riskset = "full"` (default) consists of all the possible dyadic events given the number of actors (and the number of event types) and it mantains the same structure over time. `riskset = "active"` considers at risk only the observed dyads and it mantains the same structure over time. `riskset = "manual"`, allows the risk set to have a structure that is user-defined, and it is based on the instructions supplied via the argument `omit_dyad`. This type of risk set allows for time-varying risk set, in which, for instance, subset of actors can interact only at specific time windows, or events of a specific type (sentiment) can't be observed within time intervals that are defined by the user.
`origin`	[*optional*] starting time point of the observaton period (default is `NULL`). If it is supplied, it must have the same class of the 'time' column in the input `edgelist`.
`omit_dyad`	[*optional*] list of lists. Each list refers to one risk set modification and must have two objects: a first object named 'time', that is a vector of two values defining the first and last time point of the time window where to apply the change to the risk set and a second object, named 'dyad', which is a `data.frame` where dyads to be removed are supplied in the format `actor1,actor2,type` (by row). The `NA` value can be used to remove multiple objects from the risk set at once with one risk set modification list (see Details).
`ncores`	[*optional*] number of cores used in the parallelization of the processing functions. (default is `1`).

Details

In `omit_dyad`, the `NA` value can be used to remove multiple objects from the risk set at once with one risk set modification list. For example, to remove all events with sender equal to actor "A" add a list with two objects `time = c(NA, NA)` and `dyad = data.frame(actor1 = A, actor2 = NA, type = NA)` to the `omit_dyad` list. For more details about

Value

'remify' S3 object, list of: number of events ('M'), number of actors ('N'), number of event types (if present, 'C'), number of dyads ('D', and also 'activeD' if 'riskset="active"'), vector of inter-event times (waiting times between two subsequent events), processed input edgelist as 'data.frame', processed 'omit_dyad' object as 'list'. The function returns also several attributes that make efficient the processing of the data for future analysis. For more details about the function, input arguments, output, attributes and methods, please read `vignette(package="remify",topic="remify")`.

Examples

```
# load package and random network 'randomREH'
library(remify)
data(randomREH)

# first events in the sequence
head(randomREH$edgelist)

# actor's names
```

```
randomREH$actors

# event type's names
randomREH$types

# start time of the study (origin)
randomREH$origin

# list of changes of the risk set: each one is a list of:
# 'time' (indicating the time window where to apply the risk set reduction)
# 'dyad' (a data.frame describing the dyads to remove from the risk set
# during the time window specified in 'time')
str(randomREH$omit_dyad)

# -------------------------------------- #
#  processing for tie-oriented modeling  #
# -------------------------------------- #

tie_randomREH <- remify(edgelist = randomREH$edgelist,
       directed = TRUE,
       ordinal = FALSE,
       model = "tie",
       actors = randomREH$actors,
       types = randomREH$types,
       riskset = "manual",
       origin = randomREH$origin,
       omit_dyad = randomREH$omit_dyad)

# summary
summary(tie_randomREH)

# dimensions of the processed network
dim(tie_randomREH)

# Which ID is assigned to the actors with names "Francesca" and "Kayla"?
getActorID(x = tie_randomREH, actorName = c("Francesca","Kayla"))

# Which ID is assigned to the event type "conflict"?
getTypeID(x = tie_randomREH, typeName = "conflict")

# Find dyad composition (names of actor1, actor2 and type) from the dyad ID: c(1,380,760,1140)
getDyad(x = tie_randomREH, dyadID = c(1,380,760,1140))

# visualize descriptive measures of relational event data
# plot(x = tie_randomREH)

# -------------------------------------- #
# processing for actor-oriented modeling #
# -------------------------------------- #

# loading network 'randomREHsmall'
data(randomREHsmall)
```

```
# processing small random network
actor_randomREH <- remify(edgelist = randomREHsmall$edgelist,
       directed = TRUE,
       ordinal = FALSE,
       model = "actor",
       actors = randomREHsmall$actors,
       origin = randomREHsmall$origin)

# summary
summary(actor_randomREH)

# dimensions of the processed network
dim(actor_randomREH)

# ------------------------------------ #
# for more information about remify()  #
# check: vignette(package="remify")    #
# ------------------------------------ #
```

`summary.remify` *summary.remify*

Description

A function that returns a easy-to-read summary of the main characteristics as to the processed relational event sequence.

Usage

```
## S3 method for class 'remify'
summary(object, ...)
```

Arguments

`object`	a `remify` object.
`...`	other arguments.

Value

prints out the main characteristics of the processed relational event sequence.

Examples

```
# processing the random network 'randomREHsmall'
library(remify)
data(randomREHsmall)
reh <- remify(edgelist = randomREHsmall$edgelist,
              model = "tie")

# printing a summary of the processed 'remify' object
summary(reh)
```

Index