Package 'remify'

May 9, 2026

Type Package

Title Processing and Transforming Relational Event History Data

Version 4.0.0

Date 2026-04-30

Maintainer Giuseppe Arena <g.arena@uva.nl>

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://tilburgnetworkgroup.github.io/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, remstats

LinkingTo Rcpp, RcppArmadillo,

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

LazyDataCompression gzip

NeedsCompilation yes

Author Giuseppe Arena [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5204-3326>),
Joris Mulder [aut],
Rumana Lakdawala [ctb],
Marlyne Meijerink-Bosman [ctb],
Diana Karimova [ctb],
Fabio Generoso Vieira [ctb],
Mahdi Shafiee Kamalabad [ctb],
Roger Leenders [ctb]

Repository CRAN

Date/Publication 2026-05-04 05:00:02 UTC

Contents

`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)
```

---

history *History dataset*

Description

A relational event history dataset for testing and examples.

Usage

```
history
```

Format

A data frame that can be used for an analysis with remify, remstats, remstimate

Source

remstats package

---

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 = NULL,
  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 .dict[["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 functiongrDevices::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

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

# `edgelist_reh` is an object of class `remify`
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,
                     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)
```

`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"),
  thin = 1,
  actors = NULL,
  riskset = c("full", "active", "active_saturated", "manual"),
  manual.riskset = NULL,
  extend_riskset_by_type = FALSE,
  event_type = NULL,
  origin = NULL,
  time.units = c("auto", "secs", "mins", "hours", "days", "weeks"),
```

```
  attach_riskset = TRUE,
  riskset_decode = c("labels", "ids", "none"),
  riskset_max_decode = 200000L,
  event_covariates = NULL,
  ncores = 1L,
  omit_dyad = NULL
)
```

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 exact timing must be considered in the model (FALSE). (default value is FALSE). If TRUE, then the column "time" of edgelist is still used to extract the order.
model	either "tie" (default) or "actor" oriented modeling. For "tie", the riskset is at the dyad level. For "actor", the model has two sub-processes: a sender rate model (who sends next?) and a receiver choice model (who does the sender choose?). Actor-oriented modeling requires directed=TRUE. The returned object includes sender_riskset, receiver_riskset, and activeN (see @return).
thin	Integer >= 1. Event-time thinning based on unique time points. Keeps every thin-th unique event time (after time translation) and maps each event time to the next kept time point (i.e., ceiling to the kept grid). This reduces the number of unique time points (and thus memory/computation in later steps).
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.
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. riskset = "active_saturated" extends the active riskset by adding the reverse direction for each observed dyad (if A->B is observed, B->A is also at risk) and includes all event types for each observed

actor pair (type column is ignored). This reflects the assumption that observing any interaction between two actors implies both directions and all types are possible.

`manual.riskset` [*optional*] When `riskset = "manual"`, this argument of class `data.frame` specifies which dyadic riskset to consider through the entire sequence. If observed dyads from the `edgelist` are missing, they will be automatically be added.

`extend_riskset_by_type`

logical. `FALSE` (default). When event types are present (via `event_type`), controls whether the risk set is expanded over types. If `TRUE` (default when types are present), each actor pair is duplicated for each event type, so the risk set has size $D = N(N-1) \times C$ (directed) or $D = N(N-1)/2 \times C$ (undirected), and the `type` column appears in the decoded risk set. If `FALSE`, event type is treated as a mark on events only and does not expand the risk set: $D = N(N-1)$ (directed) or $D = N(N-1)/2$ (undirected), and no `type` column appears in the decoded risk set. This argument is ignored when no event types are present.

`event_type` Optional. Either `NULL` (default) or a single character string giving the name of the column in `edgelist` that contains event types (marks).

If `event_type` is `NULL`, `remify()` uses `edgelist$type` if it exists; otherwise events are treated as untyped.

If `event_type` is a column name, that column is used as the event-type mark. If a column named `type` already exists and `event_type != "type"`, the existing `edgelist$type` is overridden (with a warning).

When event types are present (via `edgelist$type` or `event_type`), the dyadic risk set is extended over types, i.e., each dyad is duplicated for each event type (dyad $\times$ type).

`origin` [*optional*] starting time point of the observation period (default is `NULL`). If it is supplied, it must have the same class of the 'time' column in the input `edgelist`. If unsupplied, the origin is set to the average waiting time in the sequence subtracted from the time of the first event.

`time.units` Character string specifying the time unit for converting time values when 'edgelist$time' is of class Date or POSIXct; ignored for numeric or integer time. Default is "secs".

`attach_riskset` Logical. If `TRUE`, attaches a list `riskset_info` to the returned `remify` object. The list contains the effective risk set representation used for estimation (e.g., `riskset_idx`, `dyadIDactive`, dictionaries, and basic risk set metadata). This is intended to make the returned object self-describing and easier to inspect/debug.

`riskset_decode` Character. Controls how (and whether) the included risk set dyads are decoded and attached in `riskset_info$included`.

`"labels"` Attach a decoded dyad table including actor (and type) labels (e.g., `actor1`, `actor2`, and optional `type`).

`"ids"` Attach a decoded dyad table with integer IDs only (e.g., `actor1ID`, `actor2ID`, optional `typeID`, and `dyadID`).

`"none"` Do not attach a decoded dyad table.

`riskset_max_decode`

Integer. Maximum number of included dyads (i.e., `length(riskset_idx) / D_active`) for which `riskset_decode="labels"` is allowed. If the included

risk set exceeds this threshold, decoding to labels is skipped (typically falling back to "ids" with a warning) to avoid large memory usage.

event_covariates
: Optional character vector of column names in edgelist to retain as additional event-level variables in the returned reh object.

  These columns are stored as reh$event_covariates together with the corresponding time, actor1, and actor2 columns (and an internal .event_id). This is useful when downstream functions (e.g., in **remstats**) need access to event-level marks/covariates that are not part of the core reh$edgelist produced by remify().

  Note: event_covariates does not affect risk set construction or type handling in remify(); it only preserves additional columns for later use. Currently there is no further support yet when event_covariates have been added.

ncores
: [*optional*] number of cores used in the parallelization of the processing functions. (default is 1).

omit_dyad
: Deprecated. Set to NULL.

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

A remify S3 object (list) with the following elements:

- M number of events (or unique time points if simultaneous events exist).
- N number of actors.
- C number of event types (1 if untyped).
- D number of dyads in the riskset.
- intereventTime vector of inter-event waiting times (NULL if ordinal=TRUE).
- edgelist processed input edgelist as data.frame.
- edgelist_id per-event integer ID summary.
- meta list of metadata (model, directed, ordinal, riskset, dictionary, etc.).
- ids list of per-event integer IDs (actor1, actor2, dyad, type).
- index list of decoded riskset tables (dyad_map or dyad_map_active for tie model; sender_map for actor model).
- activeD number of active dyads (tie model, riskset="active" or "manual" only).
- riskset_info decoded riskset metadata (tie model only, when attach_riskset=TRUE).

For **actor-oriented models** (model="actor"), the following additional elements are returned:

- sender_riskset integer vector of actor IDs allowed to send (all actors for "full"; observed senders for "active"; senders in manual.riskset for "manual"/"active_saturated").

- `receiver_riskset` named list (actor names) of integer vectors of allowed receiver IDs per sender.
- `activeN` number of active senders.
- `index\$sender_map` data.frame with columns `senderID` and `actorName` for active senders.

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$edgelist,
       directed = TRUE,
       ordinal = FALSE,
       model = "tie",
       origin = randomREH$origin)

# summary
summary(tie_randomREH)

# 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)

# visualize
plot(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