Package 'remstimate'

May 13, 2024

Type Package

Title Optimization Frameworks for Tie-Oriented and Actor-Oriented Relational Event Models

Version 2.3.9

Date 2024-05-13

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

Description A comprehensive set of tools designed for optimizing likelihood within a tie-oriented (Butts, C., 2008, <doi:10.1111/j.1467-9531.2008.00203.x>) or an actor-oriented modelling framework (Stadtfeld, C., & Block, P., 2017, <doi:10.15195/v4.a14>) in relational event networks. The package accommodates both frequentist and Bayesian approaches. The frequentist approaches that the package incorporates are the Maximum Likelihood Optimization (MLE) and the Gradient-based Optimization (GDADAMAX). The Bayesian methodologies included in the package are the Bayesian Sampling Importance Resampling (BSIR) and the Hamiltonian Monte Carlo (HMC). The flexibility of choosing between frequentist and Bayesian optimization approaches allows researchers to select the estimation approach which aligns the most with their analytical preferences.

License MIT + file LICENSE

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

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

Depends R (>= 4.0.0)

Imports methods, Rcpp, remify (>= 3.2.4), trust, remstats (>= 3.2.1), mvnfast

Suggests knitr, rmarkdown, tinytest

LinkingTo Rcpp, RcppArmadillo, remify

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 [aut],
Fabio Generoso Vieira [aut],
Marlyne Meijerink-Bosman [ctb],
Diana Karimova [ctb],
Mahdi Shafiee Kamalabad [ctb],
Roger Leenders [ctb],
Joris Mulder [ctb]

Repository CRAN

Date/Publication 2024-05-13 16:10:02 UTC

R topics documented:

`aic` *aic*

Description

A function that returns the AIC (Akaike's Information Criterion) value in a 'remstimate' object.

Usage

```
aic(object, ...)

## S3 method for class 'remstimate'
aic(object, ...)
```

Arguments

`object`	is a `remstimate` object.
`...`	further arguments to be passed to the 'aic' method.

Value

AIC value of a 'remstimate' object.

Methods (by class)

- `aic(remstimate)`: AIC (Akaike's Information Criterion) value of a 'remstimate' object

Examples

```
# ----------------------------------- #
#       tie-oriented model: "MLE"     #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "MLE",
                                  ncores = 1)

# AIC
aic(tie_mle) #
```

`aicc` *aicc*

Description

A function that returns the AICC (Akaike's Information Corrected Criterion) value in a 'remstimate' object.

Usage

```
aicc(object, ...)

## S3 method for class 'remstimate'
aicc(object, ...)
```

Arguments

object	is a remstimate object.
...	further arguments to be passed to the 'aicc' method.

Value

AICC value of a 'remstimate' object.

Methods (by class)

- aicc(remstimate): AICC (Akaike's Information Corrected Criterion) value of a 'remstimate' object

Examples

```
# ----------------------------------- #
#       tie-oriented model: "MLE"     #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "MLE",
                                  ncores = 1)

# AICC
aicc(tie_mle)
```

ao_data *Actor-Oriented Relational Event History*

Description

A randomly generated sequence of relational events with 5 actors and 100 events. The event sequence is generated by following an actor-oriented modeling approach (for more information on the algorithm used for the generation, refer to `help(topic = remulateActor, package = "remulate")` or `?remulate::remulateActor`).

Usage

```
data(ao_data)
```

Format

`ao_data` is a list object containing the following objects:

`edgelist` a `data.frame` with the raw simulated edgelist. The columns of the `data.frame` are:

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

`seed` the seed value used in `remulate::remulateActor()` for generating the event sequence

`true.pars` a list of two vectors named `"rate_model"` and `"choice_model"`, each containing the values of the parameters used in the generation of the event sequence

Examples

```
# (1) load the data into the workspace
data(ao_data)

# (2) process event sequence with \code{remify}
ao_reh <- remify::remify(edgelist = ao_data$edgelist, model = "actor")

# (3) define linear predictor and claculate stastistcs with \code{remstats} package

## linear predictor for the rate model
rate_model <- ~ 1 + remstats::indegreeSender()

## linear predictror for the choice model
choice_model <- ~ remstats::inertia() + remstats::reciprocity()

## calculate statistics
ao_reh_stats <- remstats::remstats(reh = ao_reh, sender_effects = rate_model,
receiver_effects = choice_model)
```

```
# (4) estimate model using method = "MLE" and print out summary

## estimate model
mle_ao <- remstimate::remstimate(reh = ao_reh, stats = ao_reh_stats, method = "MLE")

## print out a summary of the estimation
summary(mle_ao)
```

`bic` *bic*

Description

A function that returns the BIC (Bayesian Information Criterion) value in a 'remstimate' object.

Usage

```
bic(object, ...)

## S3 method for class 'remstimate'
bic(object, ...)
```

Arguments

`object`	is a `remstimate` object.
`...`	further arguments to be passed to the 'bic' method.

Value

BIC value of a 'remstimate' object.

Methods (by class)

- `bic(remstimate)`: BIC (Bayesian Information Criterion) value of a 'remstimate' object

Examples

```
# ----------------------------------- #
#       tie-oriented model: "MLE"     #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")
```

```
# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "MLE",
                                  ncores = 1)

# BIC
bic(tie_mle)
```

`diagnostics` *Compute the diagnostics of a* `remstimate` *object*

Description

A function that returns the diagnostics of a `remstimate` object. The output object of the method `diagnostics` contains the residuals of the model estimated in the `remstimate` object, and the event rates estimated from the model at each tiem point. For tie-oriented modeling frameworks the object contains: a list `residuals` with two objects, `standardized_residuals` containing standardized Schoenfeld's residuals (Schoenfeld, D., 1982, <doi:10.2307/2335876>; Grambsch, P. M., & Therneau, T. M., 1994, <doi:10.2307/2337123>; Winnett, A., & Sasieni, P., 2001, <jstor.org/stable/2673500>), and `smoothing_weights` (a matrix of weights used for the red smooth splines in the plot of the residuals), an array structure `rates` with the event rates estimated from the optimized model parameters, and `.reh.processed` which is a pseudo-hidden object containing a further processed `remify` object that helps speed up the plotting function `plot.remstimate` and that the user is not supposed to modify. As to the actor-oriented modeling frameworks, in the diagnostics output there are two main list objects named after `sender_model` and `receiver_model`. After selecting the model, the structure of diagnostics is the same as for the tie-oriented model. Each model's diagnostics (sender or receiver) is available only if the corresponding model is found in the `remstimate` object.

Usage

```
diagnostics(object, reh, stats, ...)

## S3 method for class 'remstimate'
diagnostics(object, reh, stats, ...)
```

Arguments

object	is a remstimate object.
reh	is a remify object, the same used for the 'remstimate' object.
stats	is a remstats object, the same used for the 'remstimate' object.
...	further arguments to be passed to the 'diagnostics' method.

Value

a object of class "diagnostics" "remstimate" with standardized Schoenfeld's residuals and estimated event rates given the optimized model parameters.

Methods (by class)

- diagnostics(remstimate): diagnostics of a 'remstimate' object

Examples

```
# ----------------------------------- #
#       tie-oriented model: "MLE"     #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "MLE",
                                  ncores = 1)

# diagnostics
tie_diagnostics <- diagnostics(object = tie_mle, reh = tie_reh, stats = tie_reh_stats)
names(tie_diagnostics)
```

`plot.remstimate` *Plot diagnostics of a* `remstimate` *object*

Description

A function that returns a plot of diagnostics given a 'remstimate' object and depending on the 'approach' attribute.

Usage

```
## S3 method for class 'remstimate'
plot(
  x,
  reh,
  diagnostics = NULL,
  which = c(1:4),
  effects = NULL,
  sender_effects = NULL,
  receiver_effects = NULL,
  ...
)
```

Arguments

`x`	is a `remstimate` object.
`reh`	a `remify` object, the same used for the `remstimate` object.
`diagnostics`	is a `'diagnostics' 'remstimate'` object.
`which`	one or more numbers between 1 and 2. Plots described in order: (1) two plots: a Q-Q plot of the waiting times where theoretical quantiles (Exponential distribution with rate 1) are plotted against observed quantiles (these are calculated as the multiplication at each time point between the sum of the event rates and the corresponding waiting time, which should be distributed as an exponential with rate 1). Next to the q-q plot, a density plot of the rescaled waiting times (in red) vs. the theoretical distribution (exponential distribution with rate 1, in black). The observed density is truncated at the 99th percentile of the waiting times, (2) standardized Schoenfeld's residuals (per each variable in the model, excluding the baseline) with smoothed weighted spline (line in red). The Schoenfeld's residuals help understand the potential presence of time dependence of the effects of statistics specified in the model, (3) distributions of posterior draws with histograms (only for BSIR and HMC method), (4) trace plots of posterior draws after thinning (only for HMC method).
`effects`	[*optional*] for tie-oriented modeling (model = "tie"), the names of the statistics which the user wants to plot the diagnostics for (default value is set to all the statistics available inside the object 'diagnostics'). The user can specify this argument for the standardized Schoenfeld's residuals (`which = 2`), histograms of posterior distributions (`which = 3`) and trace plots (`which = 4`). Default value is `NULL`, selecting all the effects available in the 'remstimate' object.

`sender_effects` [*optional*] for actor-oriented modeling (model = "actor"), the names of the statistics as to the sender model which the user wants to plot the diagnostics for (default value is set to all the statistics available inside the object 'diagnostics'). The user can specify this argument for the standardized Schoenfeld's residuals (`which = 2`), histograms of posterior distributions (`which = 3`) and trace plots (`which = 4`). If the user wants to plot only the diagnostics of one or more effects of the sender model and at the same time wants to exclude the plots of the receiver model, then set argument `receiver_effects = NA` and specify the vector of effects to `sender_effects` (or leave it `sender_effects = NULL` for selecting all effects of the sender model). Default value is `NULL`, selecting all the effects available for the sender model in the 'remstimate' object.

`receiver_effects` [*optional*] for actor-oriented modeling (model = "actor"), the names of the statistics as to the receiver model which the user wants to plot the diagnostics for (default value is set to all the statistics available inside the object 'diagnostics'). The user can specify this argument for the standardized Schoenfeld's residuals (`which = 2`), histograms of posterior distributions (`which = 3`) and trace plots (`which = 4`). If the user wants to plot only the diagnostics of one or more effects of the receiver model and at the same time wants to exclude the plots of the sender model, then set argument `sender_effects = NA` and specify the vector of effects to `receiver_effects` (or leave it `receiver_effects = NULL` for selecting all effects of the receiver model). Default value is `NULL`, selecting all the effects available for the receiver model in the 'remstimate' object (`x`).

`...` further arguments to be passed to the 'plot' method, for instance, the remstats object with statistics ('stats') when the object 'diagnostics' is not supplied.

Value

no return value. The function plots the diagnostics of a 'remstimate' object.

Examples

```
# ----------------------------------- #
#       tie-oriented model: "MLE"     #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
```

```
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "MLE",
                                  ncores = 1)

# diagnostics
tie_diagnostics <- diagnostics(object = tie_mle, reh = tie_reh, stats = tie_reh_stats)

# plot
plot(x = tie_mle, reh  = tie_reh, diagnostics = tie_diagnostics)
```

`print.remstimate` *Print out a quick overview of a* `remstimate` *object*

Description

A function that prints out the estimates returned by a 'remstimate' object.

Usage

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

Arguments

`x`	is a `remstimate` object.
`...`	further arguments to be passed to the print method.

Value

no return value. Prints out the main characteristics of a 'remstimate' object.

Examples

```
# ----------------------------------- #
#       method 'print' for the        #
#       tie-oriented model: "BSIR"    #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
```

```
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "BSIR",
                                  nsim = 100,
                                  ncores = 1)

# print
tie_mle

# ----------------------------------- #
#      method 'print' for the         #
#      actor-oriented model: "BSIR"   #
# ----------------------------------- #

# loading data
data(ao_data)

# processing event sequence with remify
ao_reh <- remify::remify(edgelist = ao_data$edgelist, model = "actor")

# specifying linear predictor (for sender rate and receiver choice model)
rate_model <- ~ 1 + remstats::indegreeSender()
choice_model <- ~ remstats::inertia() + remstats::reciprocity()

# calculating statistics
ao_reh_stats <- remstats::remstats(reh = ao_reh,
                                   sender_effects = rate_model,
                                   receiver_effects = choice_model)

# running estimation
ao_mle <- remstimate::remstimate(reh = ao_reh,
                                 stats = ao_reh_stats,
                                 method = "BSIR",
                                 nsim = 100,
                                 ncores = 1)

# print
ao_mle

# ----------------------------------- #
```

```
#   for more examples check vignettes  #
# ---------------------------------- #
```

---

remstimate *remstimate - optimization of tie-oriented and actor-oriented likelihood*

Description

A function for the optimization of tie-oriented and actor-oriented likelihood. There are four optimization algorithms: two Frequentists, Maximum Likelihood Estimation (MLE) and Adaptive Gradient Descent (GDADAMAX), and two Bayesian, Bayesian Sampling Importance Resampling (BSIR) and Hamiltonian Monte Carlo (HMC).

Usage

```
remstimate(
  reh,
  stats,
  method = c("MLE", "GDADAMAX", "BSIR", "HMC"),
  ncores = attr(reh, "ncores"),
  prior = NULL,
  nsim = 1000L,
  nchains = 1L,
  burnin = 500L,
  thin = 10L,
  init = NULL,
  epochs = 1000L,
  L = 50L,
  epsilon = ifelse(method == "GDADAMAX", 0.001, 0.002),
  seed = NULL,
  WAIC = FALSE,
  silent = TRUE,
  ...
)
```

Arguments

reh	a remify object of the processed relational event history. Output object of the function remify::remify().
stats	a remstats object: when ‘attr(reh,"model")‘ is ‘"tie"‘, stats is an array of statistics with dimensions [M x D x P]: where M is the number of events, D is the number of possible dyads (full riskset), P is the number of statistics; if ‘attr(reh,"model")‘ is ‘"actor"‘, stats is a list that can contain up to two arrays named "sender_stats" and "receiver_stats" with dimensions [M x N x P], where N are the actors (senders in the array "sender_stats", receivers in the array "receiver_stats"). Furthermore, it is possible to only estimate

the sender rate model or only the receiver choice model, by using the correct naming of the arrays.

`method` the optimization method to estimate model parameters. Methods available are: Maximum Likelihood Estimation (`"MLE"`, and also the default method), Adaptive Gradient Descent (`"GDADAMAX"`) based on the work of Diederik P. Kingma and Jimmy Ba, 2014 (<doi:10.48550/arXiv.1412.6980>), Bayesian Sampling Importance Resampling (`"BSIR"`), Hamiltonian Monte Carlo (`"HMC"`). (default method is `"MLE"`).

`ncores` [*optional*] number of threads for the parallelization. (default value is `1`, which means no parallelization).

`prior` [*optional*] prior distribution when `method` is `"BSIR"`. Default value is `NULL`, which means that no prior is assumed. For the tie-oriented modeling, the argument `prior` is the name of the function in the format `name_package::name_density_function`. The parameters of the prior distribution can be supplied as inputs to the remstimate function (e.g., `remstimate::remstimate(reh=reh,stats=stats,method="BSIR",ncores=5,pr`). For actor-oriented modeling the argument `prior` is a named list of two objects `"sender_model"`, which calls the prior function for the sender rate model, and, `"receiver_model"`, which calls the prior function for the receiver choice model. For the specification of the prior parameters, the user must define an optional argument called `prior_args`, which is also a named list (with names `"sender_model"` and `"receiver_model"`): each list is a list of objects named after the prior arguments and with value of the prior argument (e.g., `prior_args$sender_model = list(mu = rep(1.5,3), sigma = diag(3)*0.5, log = TRUE)`). Finally, both in tie-oriented and actor-oriented modeling prior functions must have an argument that returns the value of the density on a logarithmic scale (i.e., `log=TRUE`). `log=TRUE` is already set up internally by `remstimate()`.

`nsim` [*optional*] when `method` is `"HMC"`, `nsim` is the number of simulations (iterations) in each chain, when `method` is `"BSIR"`, then `nsim` is the number of samples from the proposal distribution. Default value is `1000`.

`nchains` [*optional*] number of chains to generate in the case of `method = "HMC"`. Default value is `1`.

`burnin` [*optional*] number of initial iterations to be added as burnin for `method = "HMC"`. Default value is `500`.

`thin` [*optional*] number of steps to skip in the posterior draws for `method = "HMC"`. Default value is `10`. If `nsim<100`, thin is set to `1`.

`init` [*optional*] vector of initial values if tie-oriented model, or a named list of two vectors ('sender_model' and 'receiver_model') if both models of the actor-oriented framework are specified. `init` can also be a list of only one vector (named 'sender_model' or 'receiver_model'), if the interest is to estimate one specific model of the actor-oriented framework. `init` is used for the methods `"GDADAMAX"` and `"HMC"`. If `init` is `NULL`, then it will be assigned internally.

`epochs` [*optional*] It is the number of iteration used in the method `"GDADAMAX"`. Default value is `1000`.

`L` [*optional*] number of leap-frog steps to use in the method `"HMC"`. Default value is `50`.

epsilon	[*optional*] It is a parameter used in two methods: if method is "GDADAMAX", it represents the inter-iteration difference of the loss function and it is used as stop-rule within the algorithm (default value is 0.001), if method is "HMC" (default value is 0.002), it is a parameter used in the leap-frog algorithm and it is proportional to the step size.
seed	[*optional*] seed value for reproducibility. If NULL, seed will be assigned by the machine and saved in the output object.
WAIC	[*optional*] logical value. The Watanabe Akaike's Information Criterion (WAIC) will be calculated is WAIC = TRUE. The default number of simulations used to calculate the WAIC is 500. In order to specify a different number of simulations, the user must supply an additional argument nsimWAIC to the function.
silent	[*optional*-not-yet-implemented] a TRUE/FALSE value. If FALSE, progress of optimization status will be printed out.
...	additional parameters. They can be parameters of other functions defined as input in some of the arguments above. (e.g., arguments of the prior distribution)

Value

'remstimate' S3 object.

Examples

```
# ----------------------------------- #
#       tie-oriented model: "MLE"     #
# ----------------------------------- #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "MLE",
                                  ncores = 1)
# summary
summary(tie_mle)
```

```
# ------------------------------------ #
#      actor-oriented model: "MLE"     #
# ------------------------------------ #

# loading data
data(ao_data)

# processing event sequence with remify
ao_reh <- remify::remify(edgelist = ao_data$edgelist, model = "actor")

# specifying linear predictor (for sender rate and receiver choice model)
rate_model <- ~ 1 + remstats::indegreeSender()
choice_model <- ~ remstats::inertia() + remstats::reciprocity()

# calculating statistics
ao_reh_stats <- remstats::remstats(reh = ao_reh,
                                   sender_effects = rate_model,
                                   receiver_effects = choice_model)

# running estimation
ao_mle <- remstimate::remstimate(reh = ao_reh,
                                 stats = ao_reh_stats,
                                 method = "MLE",
                                 ncores = 1)
# summary
summary(ao_mle)

# ------------------------------------ #
#   for more examples check vignettes  #
# ------------------------------------ #
```

`summary.remstimate` *Generate the summary of a* `remstimate` *object*

Description

A function that returns the summary of a `remstimate` object.

Usage

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

Arguments

`object`	is a `remstimate` object.
`...`	further arguments to be passed to the 'summary' method.

Value

no return value. Prints out the summary of a 'remstimate' object. The output can be save in a list, which contains the information printed out by the summary method.

Examples

```
# ------------------------------------ #
#       method 'summary' for the       #
#       tie-oriented model: "BSIR"     #
# ------------------------------------ #

# loading data
data(tie_data)

# processing event sequence with remify
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# specifying linear predictor
tie_model <- ~ 1 +
               remstats::indegreeSender()+
               remstats::inertia()+
               remstats::reciprocity()

# calculating statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh,
                                    tie_effects = tie_model)

# running estimation
tie_mle <- remstimate::remstimate(reh = tie_reh,
                                  stats = tie_reh_stats,
                                  method = "BSIR",
                                  nsim = 100,
                                  ncores = 1)

# summary
summary(tie_mle)

# ------------------------------------ #
#      method 'summary' for the        #
#      actor-oriented model: "BSIR"    #
# ------------------------------------ #

# loading data
data(ao_data)

# processing event sequence with remify
ao_reh <- remify::remify(edgelist = ao_data$edgelist, model = "actor")

# specifying linear predictor (for sender rate and receiver choice model)
rate_model <- ~ 1 + remstats::indegreeSender()
choice_model <- ~ remstats::inertia() + remstats::reciprocity()
```

```
# calculating statistics
ao_reh_stats <- remstats::remstats(reh = ao_reh,
                                   sender_effects = rate_model,
                                   receiver_effects = choice_model)

# running estimation
ao_mle <- remstimate::remstimate(reh = ao_reh,
                                 stats = ao_reh_stats,
                                 method = "BSIR",
                                 nsim = 100,
                                 ncores = 1)

# summary
summary(ao_mle)

# ------------------------------------ #
#   for more examples check vignettes  #
# ------------------------------------ #
```

`tie_data` *Tie-Oriented Relational Event History*

Description

A randomly generated sequence of relational events with 5 actors and 100 events. The event sequence is generated by following a tie-oriented modeling approach (for more information run on console `help(topic = remulateTie, package = "remulate")` or `?remulate::remulateTie`).

Usage

```
data(tie_data)
```

Format

`tie_data` is a list object containing the following objects:

`edgelist` a `data.frame` with the raw simulated edgelist. The columns of the `data.frame` are:

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

`seed` the seed value used in `remulate::remulateTie()` for generating the event sequence

`true.pars` a vector containing the values of the parameters used in the generation of the event sequence

Examples

```
# (1) load the data into the workspace
data(tie_data)

# (2) process event sequence with \code{remify}
tie_reh <- remify::remify(edgelist = tie_data$edgelist, model = "tie")

# (3) define linear predictor and claculate stastistcs with \code{remstats} package

## linear predictor
tie_model <- ~ 1 + remstats::indegreeSender() + remstats::inertia() + remstats::reciprocity()

## calculate statistics
tie_reh_stats <- remstats::remstats(reh = tie_reh, tie_effects = tie_model)

# (4) estimate model using method = "MLE" and print out summary

## estimate model
mle_tie <- remstimate::remstimate(reh = tie_reh, stats = tie_reh_stats, method = "MLE")

## print out a summary of the estimation
summary(mle_tie)
```

`waic` *waic*

Description

A function that returns the WAIC (Watanabe-Akaike's Information Criterion) value in a 'remstimate' object.

Usage

```
waic(object, ...)

## S3 method for class 'remstimate'
waic(object, ...)
```

Arguments

`object`	is a `remstimate` object.
`...`	further arguments to be passed to the 'waic' method.

Value

WAIC value of a 'remstimate' object.

Methods (by class)

- `waic(remstimate)`: WAIC (Watanabe-Akaike's Information Criterion) value of a 'remstimate' object

Examples

```
# No examples available at the moment
```

Index