# Package ‘remulate’

April 16, 2025

**Type** Package

**Title** Simulate Dynamic Networks from Relational Event Models

**Version** 2.1.0

**Date** 2025-03-25

**Maintainer** Rumana Lakdawala <rumanalakdawala@gmail.com>

**Description** Model based simulation of dynamic networks under tie-oriented (Butts, C., 2008, <doi:10.1111/j.1467-9531.2008.00203.x>) and actor-oriented (Stadtfeld, C., & Block, P., 2017, <doi:10.15195/v4.a14>) relational event models. Supports simulation from a variety of relational event model extensions, including temporal variability in effects, heterogeneity through dyadic latent class relational event models (DLC-REM), random effects, blockmodels, and memory decay in relational event models (Lakdawala, R., 2024 <doi:10.48550/arXiv.2403.19329>). The development of this package was supported by a Vidi Grant (452-17-006) awarded by the Netherlands Organization for Scientific Research (NWO) Grant and an ERC Starting Grant (758791).

**License** MIT + file LICENSE

**URL** https://github.com/TilburgNetworkGroup/remulate

**Depends** R (>= 4.0.0),

**Imports** Rcpp, stats

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** tinytest

**RoxygenNote** 7.3.2

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** Rumana Lakdawala [aut, cre] (<https://orcid.org/0000-0002-9992-6035>),
Marlyne Meijerink-Bosman [ctb],
Giuseppe Arena [ctb],
Diana Karimova [ctb],
Mahdi Shafiee Kamalabad [ctb],
Fabio Generoso Vieira [ctb],
Roger Leenders [ctb],
Joris Mulder [ctb]

**Repository** CRAN

**Date/Publication** 2025-04-16 13:40:03 UTC

## Contents

`average` *average*

## Description

This function specifies the input for the average effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

## Usage

```
average(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |
| `attr_actors` | data.frame object with rows specifying values of attr_actors for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

## Details

Dyadic covariate: average attribute value for dyad (i,j) is the average of the attribute values for actors i, j

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `baseline` *baseline*

### Description

This function specified the input for the baseline effect in the `formula` argument for the function `remulateTie` or `remulateActor`.

### Usage

```
baseline(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the baseline in the REM model |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `difference` *difference*

### Description

This function specifies the input for the difference effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
difference(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |
| `attr_actors` | data.frame object with rows specifying values of attr_actors for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Dyadic covariate: (Heterophily) is the tendency to create an event i->j if actors i and j have a high absolute difference in attribute values

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ‘remulate::remulateTie()‘ or 'remulate::remulateActor()' to compute the statistic.

## `dyad` *dyad*

### Description

This function specifies the input for the dyad effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
dyad(param = NULL, variable, attr_dyads, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |

| | |
|---|---|
| `attr_dyads` | data.frame object with rows specifying values of attr_dyads for a pair of actors (dyad). First column must contain sender id, Second column receiver id, Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Dyadic covariate: dyad attribute value is the tendency to create an event i -> j when (i,j) has a high attribute value.

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

## `indegreeReceiver` *indegreeReceiver*

---

### Description

This function specifies the input for the indegreeReceiver effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
indegreeReceiver(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the indegreeReceiver statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the number of past events until time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

## `indegreeSender` *indegreeSender*

---

### Description

This function specifies the input for the indegreeSender effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
indegreeSender(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the indegreeSender statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the number of past events until time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## inertia *inertia*

### Description

This function specifies the input for the inertia effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
inertia(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the inertia statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the out degree of the sender at time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## interact *interact*

### Description

This function specifies the input for the interact effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
interact(param = NULL, indices, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `indices` | is a numeric vector of indices corresponding to the effects specified in `effects` argument of function `remulateTie` or `remulateActor` on which the interaction term needs to be computed. |
| `scaling` | specifies the method for scaling the statistic after the interaction has been computed. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

`isp` *isp*

---

## Description

This function specifies the input for the isp effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

## Usage

```
isp(param = NULL, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the isp statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

## `itp` *itp*

---

### Description

This function specifies the input for the itp effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
itp(param = NULL, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the itp statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## maximum *maximum*

### Description

This function specifies the input for the maximum effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
maximum(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |
| `attr_actors` | data.frame object with rows specifying values of attr_actors for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Dyadic covariate: maximum attribute value for dyad (i,j) is the bigger of the attribute values for actors i , j

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `minimum` *minimum*

### Description

This function specifies the input for the minimum effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
minimum(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |
| `attr_actors` | data.frame object with rows specifying values of attr_actors for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Dyadic covariate: minimum attribute value for dyad (i,j) is the smaller of the attribute values for actors i , j

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ‘remulate::remulateTie()‘ or 'remulate::remulateActor()' to compute the statistic.

## osp *osp*

### Description

This function specifies the input for the osp effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
osp(param = NULL, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the osp statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## ospSender *otp sender*

### Description

This function specified the input for the baseline effect in the `formula` argument for the function `remulateActor`. Not to be used independently.

### Usage

```
ospSender(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the baseline in the REM model |
| `scaling` | the method for scaling the otp sender statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the out degree of the sender at time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateActor()' to compute the statistic.

---

## `otp` *otp*

---

### Description

This function specifies the input for the otp effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
otp(param = NULL, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the otp statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

---

## otpSender — *osp sender This function specifies the input for the osp sender effect in the* `s_formula` *argument for the function* `remulateActor`*. Not to be used independently.*

---

### Description

osp sender

This function specifies the input for the osp sender effect in the `s_formula` argument for the function `remulateActor`. Not to be used independently.

### Usage

```
otpSender(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the baseline in the REM model |
| `scaling` | the method for scaling the osp sender statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the out degree of the sender at time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it’s first argument must be ’t’, corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ’remulate::remulateActor()’ to compute the statistic.

## `outdegreeReceiver` *outdegreeReceiver*

### Description

This function specifies the input for the outdegreeReceiver effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
outdegreeReceiver(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the outdegreeReceiver statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the number of past events until time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ‘remulate::remulateTie()‘ or 'remulate::remulateActor()' to compute the statistic.

## `outdegreeSender` *outdegreeSender*

### Description

This function specifies the input for the outdegreeSender effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
outdegreeSender(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the outdegreeSender statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the number of past events until time t. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

`psABAY` *psABAY*

---

### Description

This function specifies the input for the psABAY effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
psABAY(param = NULL, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the psABAY statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

---

`psABBA` *psABBA*

---

## Description

This function specifies the input for the psABBA effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

## Usage

```
psABBA(param = NULL, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the psABBA statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it’s first argument must be ’t’, corresponding to the time. The function may have additional arguments.

## Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

## psABBY *psABBY*

### Description

This function specifies the input for the psABBY effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
psABBY(param = NULL, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the psABBY statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## psABXA *psABXA*

### Description

This function specifies the input for the psABXA effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
psABXA(param = NULL, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the psABXA statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

`psABXB` *psABXB*

---

## Description

This function specifies the input for the psABXB effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

## Usage

```
psABXB(param = NULL, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the psABXB statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

---

## psABXY *psABXY*

---

### Description

This function specifies the input for the psABXY effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
psABXY(param = NULL, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the psABXY statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it’s first argument must be ’t’, corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

## receive *receive*

### Description

This function specifies the input for the receive effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
receive(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |
| `attr_actors` | data.frame object with rows specifying values of attr_actors for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Receiver covariate: The tendency to create an event i->j when j has a high attribute value.

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `recencyContinue` *recencyContinue*

### Description

This function specifies the input for the recencyContinue effect in the `formula` argument for the function `remulateTie`. Not to be used independently

### Usage

```
recencyContinue(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `recencyReceiveReceiver` *recencyReceiveReceiver*

### Description

This function specifies the input for the recencyReceiveReceiver effect in the `formula` argument for the function `remulateTie`. Not to be used independently

### Usage

```
recencyReceiveReceiver(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

## `recencyReceiveSender` *recencyReceiveSender*

---

## Description

This function specifies the input for the recencyReceiveSender effect in the `formula` argument for the function `remulateTie`. Not to be used independently

## Usage

```
recencyReceiveSender(param = NULL)
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `recencySendReceiver` *recencySendReceiver*

### Description

This function specifies the input for the recencySendReceiver effect in the `formula` argument for the function `remulateTie`. Not to be used independently

### Usage

```
recencySendReceiver(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by ‘remulate::remulateTie()‘ or 'remulate::remulateActor()' to compute the statistic.

## `recencySendSender` *recencySendSender*

### Description

This function specifies the input for the recencySendSender effect in the `formula` argument for the function `remulateTie`. Not to be used independently

### Usage

```
recencySendSender(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

`reciprocity` *reciprocity*

---

## Description

This function specifies the input for the reciprocity effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

## Usage

```
reciprocity(param = NULL, scaling = c("none", "std", "prop"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the reciprocity statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the in degree of the sender at time t. |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## remulateActor — *Simulate Relational Event Data - Actor-Oriented model*

### Description

A function to simulate relational event data by sampling from an actor-oriented relational event model.

### Usage

```
remulateActor(
  rateEffects,
  choiceEffects,
  actors,
  endTime,
  events = NULL,
  startTime = 0,
  initial = 0,
  riskset = NULL,
  memory = c("full", "window", "window_m", "decay"),
  memoryParam = NULL
)
```

### Arguments

| | |
|---|---|
| `rateEffects` | A `formula` object specifying the statistics used to simulate the network under the actor rate sub-model. |
| `choiceEffects` | A `formula` object specifying the statistics used to simulate the network under the receiver choice sub-model. |
| `actors` | A numeric or character vector representing the actor names. |
| `endTime` | A numeric value specifying the end time up to which the network should be simulated. |
| `events` | [Optional] An integer specifying the maximum number of events to simulate. |
| `startTime` | [Optional] A numeric value (default = 0) indicating the time at which the simulation should start. |
| `initial` | [Optional] A numeric or `data.frame` object (default = 0) specifying how to initialize the network. - If an integer is provided, it represents the number of random events to sample before beginning data generation. - If a `data.frame` is provided with columns (time, sender, receiver), it serves as an edgelist of initial events, after which subsequent events are predicted. |
| `riskset` | [Optional] A `matrix` with columns (sender, receiver) defining a custom risk set. |
| `memory` | [Optional] A string (default = "full") specifying the memory type used for computing statistics. - ‘"full"‘: Uses the entire event history. - ‘"window"‘: Considers only events occurring within a specified time window. - ‘"window_m"‘: Considers only a specified number of most recent events. - ‘"decay"‘: Applies an exponential decay, where older events contribute less based on elapsed time. |

| | |
|---|---|
| `memoryParam` | [Optional] A numeric value (> 0) defining the memory parameter based on the selected memory type: - ‘"window"‘: Length of the time window. - ‘"window_m"‘: Number of past events to consider. - ‘"decay"‘: Half-life (i.e., time until an event’s weight is reduced to half). |

## Details

#’ If time is irrelevant and only a specific number of events are desired, set time to Inf. If both time and events are supplied then the function stops simulating whenever the first stop condition is met

A list of available statistics for actor rate model. See remulateActorEffects for details on effects:

- `baseline()`
- `indegreeSender()`
- `outdegreeSender()`
- `totaldegreeSender()`
- `ospSender()`
- `otpSender()`
- `send()`
- `interact()`

A list of available statistics for receiver choice model. See remulateActorEffects for details on effects: :

- `inertia()`
- `reciprocity()`
- `indegreeReceiver()`
- `outdegreeReceiver()`
- `totaldegreeReceiver()`
- `otp()`
- `itp()`
- `osp()`
- `isp()`
- `psABBA()`
- `psABBY()`
- `psABXA()`
- `psABXB()`
- `psABXY()`
- `psABAY()`
- `recencyContinue()`
- `recencySendReceiver()`
- `recencyReceiveReceiver()`
- `rrankReceive()`

- `receive()`
- `dyad()`
- `same()`
- `average()`
- `difference()`
- `minimum()`
- `maximum()`
- `interact()`

## Value

An object of class `"remulateActor"`. A data.frame containing the simulated event sequence with columns (time, sender, receiver). The `"remulateActor"` object has the following attributes::

**evls** A `matrix` containing the event list with columns (dyad, time), where `dyad` represents the index of the (sender, receiver) pair in the risk set.

**rateStatistics** An array of dimensions `M` x `N` x `P`, where: - `M` is the number of events, - `N` is the number of actors, - `P` is the number of sender rate statistics.

**choiceStatistics** An array of dimensions `M` x `D` x `Q`, where: - `M` is the number of events, - `D` is the number of dyads in the risk set, - `Q` is the number of receiver choice statistics.

**rateParams** A named list of rate model parameters corresponding to the specified rate statistics.

**choiceParams** A named list of choice model parameters corresponding to the specified choice statistics.

**riskset** A `matrix` with columns (sender, receiver) representing the risk set used in the simulation.

**actors** A `data.frame` mapping the actor names provided by the user to the integer IDs used in internal computations. #'

**initial** A A numeric or `data.frame` object representing the network initialization, which can be a number of random initialization events or a `data.frame` specifying pre-existing ties.

**rateEffects** A `formula` object specifying the effects included in the rate sub-model.

**choiceEffects** A `formula` object specifying the effects included in the choice sub-model.

**density** A numeric value indicating the density of the generated network, defined as the number of observed ties divided by `N*(N-1)`, where `N` is the number of actors.

## References

Lakdawala, R., Mulder, J., & Leenders, R. (2025). *Simulating Relational Event Histories: Why and How*. arXiv:2403.19329.

## Examples

```
# To generate events up to time '50' in a network of 25 actors with
# 200 random initial events

# Exogenous attributes data.frame
cov <- data.frame(
  id   = 1:25,
  time = rep(0, 25),
  sex  = sample(c(0,1), 25, replace = TRUE, prob = c(0.4, 0.6)),
  age  = sample(20:30, 25, replace = TRUE)
)

# Effects specification
rateform <- ~ remulate::baseline(-6) +
             remulate::indegreeSender(0.01) +
             remulate::send(0.02, variable = "age", attr_actors = cov) +
             remulate::interact(0.01, indices = c(2, 3))

choiceform <- ~ remulate::inertia(0.01) +
              remulate::reciprocity(-0.03) +
              remulate::interact(0.01, indices = c(2, 1))

# Calling remulateActor
remulate::remulateActor(
  rateform,
  choiceform,
  actors  = 1:25,
  endTime = 100,
  initial = 200,
  events  = 500,
)

# To predict events, given an edgelist of initial events
initialREH <- data.frame(
  time = seq(0.5, 100, 0.5),
  sender = sample(1:25, 200, TRUE),
  receiver = sample(1:25, 200, TRUE)
)

remulate::remulateActor(
  rateform,
  choiceform,
  actors  = 1:25,
  endTime = 200,
  initial = initialREH,
  events  = 500
)
```

## `remulateActorEffects` *Remulate Actor Effects*

### Description

This page lists the effects that are available in the remulate package for the actor-oriented relational event model.

## Usage

```
remulateActorEffects(rateEffects = TRUE, choiceEffects = TRUE)
```

## Arguments

| | |
|---|---|
| `rateEffects` | Logical. If TRUE, includes rate effects (i.e sender-rate effects) of the actor-oriented relational event model. If FALSE, rate effects are excluded. |
| `choiceEffects` | Logical. If TRUE, includes choice effects (i.e receiver-choice effects) of the actor-oriented relational event model. If FALSE, choice effects are excluded. |

## Details

The attr_actors object contains at least three columns (actor,time,attribute). It should be constructed as follows: Each row refers to the attribute value of actor i at timepoint t. The first column contains the actor names (corresponding to the vector of names in the `actors` argument of `remulateActor`). The second column contains the time when attributes change (set to zero if the attributes do not vary over time). Subsequent columns contain the attributes that are called in the specifications of exogenous statistics. The same `attr_actors` object can be used with multiple exogenous statistics.

## Value

Returns a character vector of available effects for the `rateEffects` or `choiceEffects` argument for the function `remulateActor`.

## remulateActor Rate Effects

**Endogenous effects:**

`baseline` Baseline tendency for actors to create events. The statistic equals to 1 for all actors in the riskset. The parameter for baseline controls the average number of events per unit time.

`indegreeSender` In degree effect of the sender is the tendency for actor i to create an event when i has received more events in the past. The statistic at timepoint t for dyad (i,j) is equal to the number of events received by actor i before timepoint t. Note: if `scaling` is "prop" for indegreeSender, the statistic for dyad (i,j) at time t is divided by the total degree of the sender i at time t.

`outdegreeSender` Out degree effect of sender is the tendency for actor i to create an event when i has sent more events in the past. Note: if `scaling` is "prop" for outdegreeSender, the statistic for dyad (i,j) at time t is divided by the total degree of the sender i at time t.

`totaldegreeSender` Total degree effect of sender is the tendency for actor i to create an event when i has sent and received more events in the past.

`ospSender` Outgoing Shared Partners actor effect is the tendency for actor i to create an event if actor i is the source in a transitive structure (i->h<-j<-i).

`otpSender` Outgoing Two Path actor effect is the tendency for sender i to create an event if actor i is the source in a transitive structure (i->h->j<-i).

**Exogenous effects:**

`send` The tendency for actor i to create an event when i has a high attribute value.

**remulateActor Choice Effects**

**Endogenous effects (Dyad statistics):**

`inertia` Inertia is the tendency to create an event i->j if the event i->j occurred in the past. The statistic at timepoint t for dyad (i,j) is equal to the number of (i,j) events before timepoint t. Note: if `scaling` is "prop" for inertia, the statistic for dyad (i,j) at time t is divided by the out degree of the sender i at time t.

`reciprocity` Reciprocity is the tendency to create an event i->j if j->i occurred in the past.The statistic at timepoint t for dyad (i,j) is equal to the number of (j,i) events before timepoint t. Note: if `scaling` is "prop" for inertia, the statistic for dyad (i,j) at time t is divided by the in degree of the sender i at time t.

`tie` Tie effect is the tendency to create an event i->j if the event i->j occurred at least once in the past. The statistic at timepoint t for dyad (i,j) is equal to 1 if a an event i->j occurred before timepoint t

**Endogenous effects (Triadic statistics):**

`otp` Outgoing Two Path effect is the tendency to create an event i->j if they have past outgoing two-paths between them (i->h->j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (i,h), (h,j) events, summed over all h.

`itp` Incoming Two Path effect is the tendency to create an event i->j if they have past incoming two-paths between them (i<-h<-j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (j,h), (h,i) events, summed over all h.

`osp` Outgoing Shared Partners effect is the tendency to create an event i->j if they have past outgoing shared partners between them (i->h<-j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (i,h), (j,h) events, summed over all h.

`isp` Incoming Shared Partners effect is the tendency to create an event i->j if they have past incoming shared partners between them (i<-h->j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (h,i), (h,j) events, summed over all h.

**Endogenous effects (Node statistics):**

`indegreeReceiver` In degree effect of receiver is the tendency to create an event i->j if j has received more events in the past. The statistic at timepoint t for dyad (i,j) is equal to the number of events received by actor j before timepoint t. Note: if `scaling` is "prop" for indegreeReceiver, the statistic for dyad (i,j) at time t is divided by the total degree of the receiver j at time t.

`outdegreeReceiver` Out degree effect of receiver is the tendency to create an event i->j if j has sent more events in the past. Note: if `scaling` is "prop" for outdegreeReceiver, the statistic for dyad (i,j) at time t is divided by the total degree of the receiver j at time t.

`totaldegreeReceiver` Total degree effect of receiver is the tendency to create an event i->j if j has sent and received more events in the past.

**Exogenous effects:**

`dyad` Dyadic attribute value is tendency to create an event i -> j when (i,j) has a high attribute value.

`receive` Receiver attribute value is the tendency to create an event i->j when j has a high attribute value.

`same` Same attribute value (Homophily) is the tendency to create an event i->j if actors i and j have the same attribute values

`Difference` difference attribute value (Heterophily) is the tendency to create an event i->j if actors i and j have a high absolute difference in attribute values

## Examples

```
#To specify an exogenous effect (example: same)

cov <- data.frame(
  actor = 1:10,
  time = rep(0, 10),
  gender = sample(c(0, 1), replace = TRUE, 10),
  age = sample(20:30, 10, replace = TRUE)
)

effects <- ~ same(0.2, variable = "gender", attr_actors = cov)

#Rate Effects:

#If parameter is constant

rateEffects <- ~ outdegreeSender(0.3) +
  send(0.1, variable = "age", attr_actors = cov)

#If parameter varies with time

rateEffects <- ~ outdegreeSender(param = function(t) exp(-t)) +
  send(0.1, variable = "age", attr_actors = cov)

#Choice Effects:

#If parameter is constant

choiceEffects <- ~ inertia(0.4) +
  reciprocity(-0.1) +
  same(0.2, variable = "gender", attr_actors = cov) +
  receive(0.1, variable = "age", attr_actors = cov)

#If parameter varies with time

choiceEffects <- ~ inertia(param = function(t) exp(-t)) +
  reciprocity(-0.1) +
  same(0.2, variable = "gender", attr_actors = cov) +
  receive(0.1, variable = "age", attr_actors = cov)
```

---

## `remulateTie` *Simulate Relational Event Data - Tie-Oriented model*

---

## Description

A function to simulate relational event data by sampling from a tie-oriented relational event model.

## Usage

```
remulateTie(
  effects,
  actors,
  endTime,
  events = NULL,
  startTime = 0,
  initial = 0,
  riskset = NULL,
  memory = c("full", "window", "window_m", "decay"),
  memoryParam = NULL
)
```

## Arguments

| | |
|---|---|
| `effects` | A `formula` object specifying the statistics used to simulate the network or an object of class "`remstimate`" containing a fitted object. |
| `actors` | A numeric or character vector representing the actor names. |
| `endTime` | A numeric value specifying the end time up to which the network should be simulated. |
| `events` | [Optional] An integer specifying the maximum number of events to simulate. |
| `startTime` | [Optional] A numeric value (default = 0) indicating the time at which the simulation should start. |
| `initial` | [Optional] A numeric or `data.frame` object (default = 0) specifying how to initialize the network. If an integer is provided, it represents the number of random events to sample before beginning data generation. If a `data.frame` is provided with columns (time, sender, receiver), it serves as an edgelist of initial events, after which subsequent events are predicted. |
| `riskset` | [Optional] A `matrix` with columns (sender, receiver) defining a custom risk set. |
| `memory` | [Optional] A string (default = "full") specifying the memory type used for computing statistics. ‘"full"‘ uses the entire event history. ‘"window"‘ considers only events occurring within a specified time window. ‘"window_m"‘ considers only a specified number of most recent events. ‘"decay"‘ applies an exponential decay, where older events contribute less based on elapsed time. |
| `memoryParam` | [Optional] A numeric value (> 0) defining the memory parameter based on the selected memory type. ‘"window"‘ defines the length of the time window. ‘"window_m"‘ specifies the number of past events to consider. ‘"decay"‘ represents the half-life (i.e., time until an event’s weight is reduced to half). |

## Details

If time is irrelevant and only a specific number of events are desired, set time to Inf. If both time and events are supplied then the function stops simulating whenever the first stop condition is met

If effects is a `'remstimate'` object, then the params are extracted automatically. Furthermore if exogenous statistics are used, the data.frames corresponding to the 'attr_actors' argument of the `'remstats'` formula must be loaded in the environment.

A list of available statistics. See remulateTieEffects for details:

- `baseline(param)`
- `send()`
- `receive()`
- `dyad()`
- `same()`
- `difference()`
- `average()`
- `minimum()`
- `maximum()`
- `inertia()`
- `reciprocity()`
- `tie()`
- `indegreeSender()`
- `indegreeReceiver()`
- `outdegreeSender()`
- `outdegreeReceiver()`
- `totaldegreeSender()`
- `totaldegreeReceiver()`
- `otp()`
- `itp()`
- `osp()`
- `isp()`
- `psABBA()`
- `psABBY()`
- `psABXA()`
- `psABXB()`
- `psABXY()`
- `psABAY()`
- `recencyContinue()`
- `recencySendSender()`
- `recencySendReceiver()`
- `recencyReceiveSender()`
- `recencyReceiveReceiver()`
- `rrankSend()`
- `rrankReceive()`
- `interact()`

## Value

An object of class "remulateTie". A data.frame containing the simulated event sequence with columns (time, sender, receiver). The "remulateTie" object has the following attributes:

**statistics** An array with dimensions M x D x P, where M is the number of events, D is the number of dyads in the risk set, and P is the number of computed statistics.

**evls** A matrix containing the event list with columns (dyad, time), where dyad represents the index of the (sender, receiver) pair in the risk set.

**actors** A data.frame mapping the actor names provided by the user to the integer IDs used in internal computations.

**riskset** A data.frame with columns (sender, receiver) containing the risk set used for dyad indices in the computed statistics and event list.

**initial** A A numeric or data.frame object representing the network initialization, which can be a number of random initialization events or a data.frame specifying pre-existing ties.

**effects** A formula object specifying the effects included in the model.

**density** A numeric value indicating the density of the generated network, defined as the number of observed ties divided by N*(N-1), where N is the number of actors.

## References

Lakdawala, R., Mulder, J., & Leenders, R. (2025). *Simulating Relational Event Histories: Why and How*. arXiv:2403.19329.

## Examples

```
# To generate events up to time '50' in a network of 25 actors with
# 200 random initial events
# Exogenous attributes data.frame

cov <- data.frame(
  id = 1:25,
  time = rep(0, 25),
  sex = sample(c(0, 1), 25, replace = TRUE, prob = c(0.4, 0.6)),
  age = sample(20:30, 25, replace = TRUE)
)

#Effects specification
effects <- ~ remulate::baseline(-5) +
            remulate::inertia(0.01) +
            remulate::reciprocity(-0.04) +
            remulate::itp(0.01, scaling = "std") +
            remulate::same(0.02, variable = "sex", attr_actors = cov) +
            remulate::interact(0.01, indices = c(2, 5))
# Calling remulateTie
remulate::remulateTie(
  effects,
  actors  = 1:25,
  endTime = 50,
  events  = 500,
  initial = 200
)

# To predict events, given an edgelist of initial events
initialREH <- data.frame(
  time = seq(0.5, 100, 0.5),
  sender = sample(1:25, 200, TRUE),
  receiver = sample(1:25, 200, TRUE)
)

remulate::remulateTie(
  effects,
  actors  = 1:25,
  endTime = 150,
  events  = 500,
  initial = initialREH
)

# Custom risk set
rs <- as.matrix(expand.grid(1:25, 1:25))
rs <- rs[rs[, 1] != rs[, 2], ]

custom_rs <- rs[sample(1:90, 50), ]

remulate::remulateTie(
  effects,
  actors  = 1:25,
  endTime = 150,
  events  = 500,
  riskset = custom_rs
)
```

---

## remulateTieEffects *Remulate Tie Effects*

---

### Description

This page lists the effects that are available in the remulate package for the tie oriented relational event model.

### Usage

```
remulateTieEffects(endogenous = NULL)
```

### Arguments

| | |
|---|---|
| endogenous | Logical or NULL. If 'TRUE', returns only endogenous effects. If 'FALSE', returns only exogenous effects. If 'NULL' (default), returns both endogenous and exogenous effects. |

## Details

The attr_actors object for exogenous effects based on actor covariates (`send`, `receive`, `same`, `difference`, `average`, `max`, `min`) contains at least three columns (actor,time,attribute). It should be constructed as follows: Each row refers to the attribute value of actor i at timepoint t. The first column contains the actor names (corresponding to the vector of names in the `actors` argument of `remulateTie`). The second column contains the time when attr_actors change (set to zero if the attr_actors do not vary over time). At least one of the subsequent columns must contain values for the attr_actors with column name corresponding to variable name specified in the effect specification.

The attribute object for exogenous effect `dyad` contains at least three columns (sender_id,receiver_id,attribute). It should be constructed as follows: First column must contain sender id, second column receiver id, at least one of the subsequent columns must contain values for the attr_actors with column name corresponding to variable name specified in the effect specification.

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

The indices aregument in the interact effect corresponds to the position of the specified effects in the `effects` argument of `remulateTie` for which the interaction needs to be computed. The individual constitutive effects for an interaction must be specified before the interact term in the `effects` argument. To omit the individual constitutive effects in the generation, specify the `param` arugment to zero.

## Value

Returns a character vector of available effects for the `effects` argument for the function `remulateTie`.

## Remulate Effects

`baseline` Baseline tendency for dyads to create events. The statistic equals to 1 for all dyads (i,j) in the riskset. The parameter for baseline controls the average number of events per unit time.

**Endogenous effects (Dyad statistics):**

`inertia` Inertia is the tendency to create an event i->j if the event i->j occurred in the past. The statistic at timepoint t for dyad (i,j) is equal to the number of (i,j) events before timepoint t. Note: if `scaling` is "prop" for inertia, the statistic for dyad (i,j) at time t is divided by the out degree of the sender i at time t.

`reciprocity` Reciprocity is the tendency to create an event i->j if j->i occurred in the past.The statistic at timepoint t for dyad (i,j) is equal to the number of (j,i) events before timepoint t. Note: if `scaling` is "prop" for reciprocity, the statistic for dyad (i,j) at time t is divided by the in degree of the sender i at time t.

`tie` Tie effect is the tendency to create an event i->j if the event i->j occurred at least once in the past. The statistic at timepoint t for dyad (i,j) is equal to 1 if a an event i->j occurred before timepoint t

**Endogenous effects (Triadic statistics):**

`otp` Outgoing Two Path effect is the tendency to create an event i->j if they have past outgoing two-paths between them (i->h->j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (i,h), (h,j) events, summed over all h.

`itp` Incoming Two Path effect is the tendency to create an event i->j if they have past incoming two-paths between them (i<-h<-j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (j,h), (h,i) events, summed over all h.

`osp` Outgoing Shared Partners effect is the tendency to create an event i->j if they have past outgoing shared partners between them (i->h<-j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (i,h), (j,h) events, summed over all h.

`isp` Incoming Shared Partners effect is the tendency to create an event i->j if they have past incoming shared partners between them (i<-h->j). The statistic for dyad (i,j) at timepoint t is equal to the minimum of past (h,i), (h,j) events, summed over all h.

**Endogenous effects (Node statistics):**

`indegreeSender` In degree effect of the sender is the tendency to create an event i->j if i has received more events in the past. The statistic at timepoint t for dyad (i,j) is equal to the number of events received by actor i before timepoint t. Note: if `scaling` is "prop" for indegreeSender, the statistic for dyad (i,j) at time t is divided by the number of past events until time t.

`indegreeReceiver` In degree effect of receiver is the tendency to create an event i->j if j has received more events in the past. The statistic at timepoint t for dyad (i,j) is equal to the number of events received by actor j before timepoint t. Note: if `scaling` is "prop" for indegreeReceiver, the statistic for dyad (i,j) at time t is divided by the number of past events until time t.

`outdegreeSender` Out degree effect of sender is the tendency to create an event i->j if i has sent more events in the past. Note: if `scaling` is "prop" for outdegreeSender, the statistic for dyad (i,j) at time t is divided by by the number of past events until time t.

`outdegreeReceiver` Out degree effect of receiver is the tendency to create an event i->j if j has sent more events in the past. Note: if `scaling` is "prop" for outdegreeReceiver, the statistic for dyad (i,j) at time t is divided by the number of past events until time t.

`totaldegreeSender` Total degree effect of sender is the tendency to create an event i->j if i has sent and received more events in the past.

`totaldegreeReceiver` Total degree effect of receiver is the tendency to create an event i->j if j has sent and received more events in the past.

**Endogenous effects (Participating Shifts):**

`psABBA` AB-BA Pacticipating shift (turn receiving) is the tendency to create an event j->i at timepoint t if event i->j occurred at timepoint t-1. The psABBA statistic is equal to one for the dyad (j.i) that will create the participation shift at timepoint t.

`psABBY` AB-BY Participating shift (turn receiving) is the tendency to create an event j->h at timepoint t if event i->j occurred at timepoint t-1. The psABBY statistic is equal to one for the dyads (j,h) for all h not equal to i, that will create the participation shift at timepoint t.

`PSABAY` AB-AY Participating shifts (turn continuing) is the tendency to create an event i->h at timepoint t if event i->j occurred at timepoint t-1. The PSABAY statistic is equal to one for the dyads (i,h) for all h not equal to j, that will create the participation shift at timepoint t.

`psABXA` AB-XA Participating shifts (turn usurping) is the tendency to create an event h->i at timepoint t if event i->j occurred at timepoint t-1. The psABXA statistic is equal to one for the dyads (h,i) for all h not equal to j, that will create the participation shift at timepoint t.

`psABXB` AB-XB Participating shifts (turn usurping) is the tendency to create an event h->j at timepoint t if event i->j occurred at timepoint t-1. The psABXB statistic is equal to one for the dyads (h,j) for all h not equal to i, that will create the participation shift at timepoint t.

`psABXY` AB-XY Participating shifts (turn usurping) is the tendency to create an event h->k at timepoint t if event i->j occurred at timepoint t-1. The psABXB statistic is equal to one for the dyads (h,k) for all h not equal to i and k not equal to j, that will create the participation shift at timepoint t.

**Endogenous effects (Recency statistics):**

`recencyContinue` The recencyContinue effect refers to a recency statistic similar to what is described in Vu et al. (2017) and Mulder and Leenders (2019). For each timepoint t, for directed dyad (i,j) the statistic is equal to 1/(the time that has past since the dyad was last active + 1).

`recencySendSender` The recencySendSender effect refers to a recency statistic similar to what is described in Vu et al. (2017) and Mulder and Leenders (2019). For each timepoint t, for directed dyad (i,j) the statistic is equal to 1/(the time that has past since sender i was last active as sender + 1).

`recencySendReceiver` The recencySendReceiver effect refers to a recency statistic similar to what is described in Vu et al. (2017) and Mulder and Leenders (2019). For each timepoint t, for directed dyad (i,j) the statistic is equal to 1/(the time that has past since receiver j was last active as sender + 1).

`recencyReceiveSender` The recencyReceiveSender effect refers to a recency statistic similar to what is described in Vu et al. (2017) and Mulder and Leenders (2019). For each timepoint t, for directed dyad (i,j) the statistic is equal to 1/(the time that has past since sender i was last active as receiver + 1).

`recencyReceiveReceiver` The recencyReceiveReceiver effect refers to a recency statistic similar to what is described in Vu et al. (2017) and Mulder and Leenders (2019). For each timepoint t, for directed dyad (i,j) the statistic is equal to 1/(the time that has past since receiver j was last active as receiver + 1).

**Exogenous effects (Node attr_actors):**

`send` Sender covariate: The tendency to create an event i->j when i has a high attribute value.

`receive` Receiver covariate: The tendency to create an event i->j when j has a high attribute value.

**Exogenous effects (Tie Attribute):**

`dyad` dyad attribute value is the tendency to create an event i -> j when (i,j) has a high attribute value.

`same` (Homophily) is the tendency to create an event i->j if actors i and j have the same attribute values

`difference` (Heterophily) is the tendency to create an event i->j if actors i and j have a high absolute difference in attribute values

`average` average attribute value for dyad (i,j) is the average of the attribute values for actors i, j

`minimum` minimum attribute value for dyad (i,j) is the smaller of the attribute values for actors i , j

`maximum` maximum attribute value for dyad (i,j) is the bigger of the attribute values for actors i , j

## Examples

```
#To specify an endogenous effect (example: inertia)

effects <- ~ inertia(0.1, scaling = "std")

#To specify an exogenous effect (example: same)

cov <- data.frame(
  actor = 1:10,
  time = rep(0, 10),
  gender = sample(c(0, 1), replace = TRUE, 10),
  age = sample(20:30, 10, replace = TRUE)
)

effects <- ~ same(0.2, variable = "gender", attr_actors = cov)

#To specify an exogenous dyadic effect (example: dyad)

cov <- expand.grid(1:10, 1:10)
cov <- cov[cov[, 1] != cov[, 2], ]
cov$grade <- runif(90, 1, 10)

effects <- ~ dyad(0.2, variable = "grade", attr_actors = cov)

#If parameter is constant

effects <- ~ inertia(0.3) +
  same(0.2, variable = "gender", attr_actors = cov) +
  reciprocity(-0.1) +
  itp(0.01)

#If parameter varies with time

effects <- ~ inertia(param = function(t) exp(-t)) +
  same(0.2, variable = "gender", attr_actors = cov) +
  reciprocity(-0.1) +
  itp(0.01)

 #If parameter varies across dyads or actors
 rs <- expand.grid(1:10,1:10)
 rs <- rs[rs[,1] != rs[, 2],]

 param_df <- as.data.frame(rs)
 param_df$beta = runif(nrow(rs),-0.1,0.1)

 effects <- ~ remulate::baseline(-3)+
    remulate::inertia(param_df) +
    remulate::reciprocity(0.1)


#To specify an interaction (example: between inertia and same constitutive effects)
```

```
effects <- ~ inertia(0.3) +
  same(0.2, variable = "gender", attr_actors = cov) +
  reciprocity(-0.1) +
  itp(0.01) +
  interact(0.1, indices = c(1, 2))
```

## `rrankReceive` *rrankReceive*

### Description

This function specifies the input for the rrankReceive effect in the `formula` argument for the function `remulateTie`. Not to be used independently

### Usage

```
rrankReceive(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `rrankSend` *rrankSend*

### Description

This function specifies the input for the rrankSend effect in the `formula` argument for the function `remulateTie`. Not to be used independently

### Usage

```
rrankSend(param = NULL)
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `same` *same*

### Description

This function specifies the input for the same effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
same(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in attr_actors data.frame |
| `attr_actors` | data.frame object with rows specifying values of attr_actors for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the attributes with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Dyadic covariate: (Homophily) is the tendency to create an event i->j if actors i and j have the same attribute values

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

---

## `send` *send*

---

### Description

This function specifies the input for the send effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
send(param = NULL, variable, attr_actors, scaling = c("none", "std"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `variable` | character vector specifies the name of the column with covariate value in data dataframe |
| `attr_actors` | data.frame object with rows specifying values of data for an actor. First column must contain actor id, Second column time when covariate value changes (default zero if no change), Third column contains values for the data with column name corresponding to variable name |
| `scaling` | specifies the method for scaling the statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time |

### Details

Sender covariate: The tendency to create an event i->j when i has a high attribute value.

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

---

`tie` *tie*

---

## Description

This function specifies the input for the tie effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

## Usage

```
tie(param = NULL, scaling = c("none", "std"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the tie statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point, and `"prop"` denotes proportional scaling in which raw counts are divided by the out degree of the sender at time t. |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it’s first argument must be ’t’, corresponding to the time. The function may have additional arguments.

## Value

List with all information required by ‘remulate::remulateTie()‘ or ’remulate::remulateActor()’ to compute the statistic.

## `totaldegreeReceiver` *totaldegreeReceiver*

### Description

This function specifies the input for the totaldegreeReceiver effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
totaldegreeReceiver(param = NULL, scaling = c("none", "std", "prop"))
```

### Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the totaldegreeReceiver statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

### Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

### Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

## `totaldegreeSender` *totaldegreeSender*

### Description

This function specifies the input for the totaldegreeSender effect in the `formula` argument for the function `remulateTie` or `remulateActor`. Not to be used independently

### Usage

```
totaldegreeSender(param = NULL, scaling = c("none", "std", "prop"))
```

## Arguments

| | |
|---|---|
| `param` | numeric value, data.frame or function with time parameter. Specifies the value of the effect for the statistic in the REM model |
| `scaling` | the method for scaling the totaldegreeSender statistic. `"none"` [default] gives raw value of the statistic at time t, `"std"` the statistic is standardized per time point. |

## Details

if param is a data frame, it must have three columns: sender, receiver, and value (numeric), representing the parameter value for thay dyadic pair. The data.frame must contain all pairs of actors or dyads corresponding to the riskset.

if param is a function, it's first argument must be 't', corresponding to the time. The function may have additional arguments.

## Value

List with all information required by 'remulate::remulateTie()' or 'remulate::remulateActor()' to compute the statistic.

# Index