Package ‘dejaVu’

October 13, 2022

Type Package

Title Multiple Imputation for Recurrent Events

Version 0.3.0

Maintainer Jonathan Bartlett <j.w.bartlett@bath.ac.uk>

Description Performs reference based multiple imputation of recurrent event data
based on a negative binomial regression model, as described

License GPL (>= 2)

LazyData true

Suggests knitr, testthat,

Depends R (>= 3.1.0)

Imports MASS, stats

VignetteBuilder knitr

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author Nikolas Burkoff [aut],
Paul Metcalfe [aut],
Jonathan Bartlett [aut, cre],
David Ruau [aut]

Repository CRAN

Date/Publication 2021-04-27 12:40:02 UTC

R topics documented:

ConstantRateDrop ........................................ 2
copy_reference ........................................... 3
CreateNewDropoutMechanism ............................. 4
CreateNewImputeMechanism ............................. 5
CreateScenario .......................................... 6
**ConstantRateDrop**

Create a Dropout Mechanism with constant dropout rate

**Description**

Creates an MCAR DropoutMechanism object where subject i dropout is exponentially distributed with rate $R_i$ where $R_i = C \exp(X_i)$ for constant $C$ and $X_i$ a random normal variable with mean 0 and standard deviation $\sigma$.

**Usage**

`ConstantRateDrop(rate, var = 0)`

**Arguments**

- `rate`: $C$ described in the details.
- `var`: $\sigma^2$ described in the details section, by default = 0.
Value

A DropoutMechanism object

See Also

DropoutMechanism.object

Examples

ConstantRateDrop(rate=0.0025)
ConstantRateDrop(rate=0.0025, var=1)

Arguments

proper

If proper=TRUE then proper imputation is performed, in which each imputation is created based on parameters values drawn from the (approximate) posterior distribution of the imputation model. If proper=FALSE, improper imputation is performed. This means all imputed datasets are generated conditional on the maximum likelihood estimates of the parameters.

Value

An ImputeMechanism object

See Also

ImputeMechanism.object
CreateNewDropoutMechanism

Examples

```r
sim <- SimulateComplete(study.time=365, number.subjects=50,
                        event.rates=c(0.01,0.005), dispersions=0.25)
sim.with.MCAR.dropout <- SimulateDropout(sim,
                          drop.mechanism = ConstantRateDrop(rate = 0.0025))
fit <- Simfit(sim.with.MCAR.dropout)
imps <- Impute(fit, copy_reference(), 10)
```

CreateNewDropoutMechanism

*A function which creates a DropOut Mechanism object*

Description

A function which creates a DropOut Mechanism object

Usage

```r
CreateNewDropoutMechanism(
  type, 
  text, 
  cols.needed = vector("character"),
  GetDropTime, 
  parameters = NULL
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The type of mechanism (e.g. &quot;MCAR&quot; or &quot;MNAR&quot;)</td>
</tr>
<tr>
<td>text</td>
<td>A short string describing the mechanism (only used for printing)</td>
</tr>
<tr>
<td>cols.needed</td>
<td>Which columns in the SingleSim$data data frame must be included for this drop out mechanism to work. This option could allow drop out mechanism which depend on covariates to be included.</td>
</tr>
<tr>
<td>GetDropTime</td>
<td>A function with two arguments event.times and data, the corresponding entries from the SingleSim object. This function should return a list of dropout times (if a subject does not dropout its dropout time should be their current censored.time (i.e. the study follow up time))</td>
</tr>
<tr>
<td>parameters</td>
<td>A list of named parameters for the mechanism (only used for printing) or NULL if none</td>
</tr>
</tbody>
</table>

Value

A `DropoutMechanism.object`
CreateNewImputeMechanism

A function which creates an Impute Mechanism object

Description

A function which creates an Impute Mechanism object

Usage

CreateNewImputeMechanism(
  name,
  cols.needed = vector("character"),
  impute,
  parameters = NULL
)

Arguments

name
  The method name (used for printing)
cols.needed
  which columns of the SingleSim data frame are required by the method, typically c("censored.time","observed.events","arm")
impute
  A function which takes a SingleSimFit object and outputs the details for a single imputed data set, specifically a list with two elements: new.censored.times - a vector of times subjects were censored (after taking into account imputation) and newevent.times - a list of vectors where the vectors contain the imputed event times for the subjects (these vectors do not contain the observed event times before subject drop out). If a subject has no imputed events then the vector numeric(0) is returned.
parameters
  A list of named parameters describing the method (used for printing) - or NULL if none

Value

A ImputeMechanism.object

See Also

ImputeMechanism.object

See Also

DropoutMechanism.object
CreateScenario  

Create Scenario object from list of Fit Summaries

Description
Create Scenario object from list of Fit Summaries

Usage
CreateScenario(object, description = "")

Arguments
object Either a list of summary.SingleSimFit or summary.ImputeSimFit objects
description A character string describing the scenario (used for printing)

Value
A Scenario object

See Also
Scenario.object

DejaData.object  

Data frame of covariates for simulating recurrent events

Description
This object allows covariates to be included in the simulation procedure. The object is created using the MakeDejaData function

Arguments
data A data frame containing the subject
arm, character the column name of the treatment arm for each subject
rate, character the column name of the rate to be used when simulating
Id, character the column name of subject Id

Structure
The above components must be included in a DejaData Object
**DropoutMechanism object**

**Description**

An object which defines a specific mechanism which takes a complete SingleSim object and returns a set of drop out times for subjects.

**Arguments**

- **type**
  - The type of mechanism (e.g. "MCAR" or "MNAR")

- **text**
  - A short string describing the mechanism (only used for printing)

- **cols.needed**
  - Which columns in the SingleSim$data data frame must be included for this drop out mechanism to work. This option could allow drop out mechanism which depend on covariates to be included.

- **GetDropTime**
  - A function with two arguments event.times and data, the corresponding entries from the SingleSim object. This function should return a list of dropout times (if a subject does not dropout its dropout time should be their current censored.time (i.e. the study follow up time))

- **parameters**
  - A list of named parameters for the mechanism (only used for printing) or NULL if none

**Details**

It is possible to create user defined mechanisms, however, certain common mechanisms have already been implemented. For example see `ConstantRateDrop` and `LinearRateChangeDrop`.

Only the GetDropTime and cols.needed entries are required for calculation, the other entries are used for printing the object.

**print.DropoutMechanism** methods is defined.

**Structure**

The following components must be included in a DropoutMechanism Object

- **expandEventCount**
  - *Expand event counts into a list of event times*

**Description**

This function exists to allow clinical trial data which typically gives event counts over time to be plugged into this software, which relies on actual event counts.
Usage

```
expandEventCount(count, time)
```

Arguments

- `count` a vector of event counts. All entries must be non-negative.
- `time` a matching (strictly positive) vector of followup times.

Details

This function always produces a warning: anyone relying on this function to actually analyze data should take great care.

Value

- a list of vectors of event times

Examples

```
expandEventCount(count=c(0, 20), time=c(10, 20))
```

---

`extract_results`  
Extract the results of running a scenario

Description

This function is a wrapper around `CreateScenario` See the user guide vignette for an example of using this function

Usage

```
extract_results(answer, name, description)
```

Arguments

- `answer` A named list of lists
- `name` The name of the lists of answer which should be extracted and put together into a `sc`
- `description` The description parameter to be passed into the `CreateScenario` function

Value

- A `Scenario` object

See Also

- `CreateScenario`
GetImputedDataSet Output a single imputed data set

Description
Output a single imputed data set

Usage
GetImputedDataSet(imputeSim, index)

Arguments
- imputeSim: A ImputeSim object which contains multiple imputed data sets
- index: numeric, which of the multiple imputed data sets to output

Value
A SingleSim object with status="imputed"

See Also
ImputeSim.object

Examples
```r
sim <- SimulateComplete(study.time=365,number.subjects=50,
event.rates=c(0.01,0.005),dispersions=0.25)
sim.with.MCAR.dropout <- SimulateDropout(sim,
  drop.mechanism = ConstantRateDrop(rate = 0.0025))
fit <- Simfit(sim.with.MCAR.dropout)
imps <- Impute(fit, copy_reference(), 10)
imp1 <- GetImputedDataSet(imps, 1)
```

ImportSim Import an existing data frame for use with the package

Description
Import an existing data frame for use with the package
Usage

ImportSim(
  dejaData,
  event.times,
  status,
  study.time,
  censored.time = NULL,
  actual.events = NULL,
  allow.beyond.study = FALSE
)

Arguments

dejaData  a DejaData object contain the subject covariates and treatment arm
event.times  A list of vectors, containing the observed event times of each subject. If no
events are observed then numeric(0) should be used. See example in this help
file for more details
status  The status of the data set imported, either "complete" (if all subjects complete
their follow up period) or "dropout" (if not)
study.time  The total follow up time according to study protocol
censored.time  If status is "dropout", this is a vector of the times at which each subject is cen-
sored
actual.events  If status is "dropout" and the total number of events (i.e. not just the number
observed) is known (e.g. if a different simulation procedure was used) a vector
of total number of events should be included. If the number is not known or
status is "complete" then this should be set to NULL
allow.beyond.study  Whether or not to allow imported data with events after the nominal end of study.

Value

A SingleSim object

Examples

covar.df <- data.frame(Id=1:6,
  arm=c(rep(0,3),rep(1,3)),
  Z=c(0,1,1,0,1,0))

dejaData <- MakeDejaData(covar.df,arm="arm",Id="Id")

event.times <- list(c(25,100,121,200,225),
  c(100,110),c(55),numeric(0),
  150,45)

complete.dataset <- ImportSim(dejaData, event.times,
Impute

status="complete",
study.time=365)
censored.time <- c(365,178,100,245,200,100)
dropout.dataset <- ImportSim(dejaData, event.times,
status="dropout",
study.time=365,
censored.time=censored.time)

---

Impute

*Produce imputed data sets*

**Description**

Given a SingleSimFit object (with impute.parameters not NULL) and an imputation mechanism, create a collection of imputed data sets

**Usage**

Impute(fit, impute.mechanism, N)

**Arguments**

- **fit**
  A SingleSimFit object
- **impute.mechanism**
  An ImputeMechanism object
- **N**
  The number of data sets to impute

**Value**

An ImputeSim object

**Examples**

```
sim <- SimulateComplete(study.time=365,number.subjects=50,
  event.rates=c(0.01,0.005),dispersions=0.25)
sim.with.MCAR.dropout <- SimulateDropout(sim,
drop.mechanism = ConstantRateDrop(rate = 0.0025))
fit <- Simfit(sim.with.MCAR.dropout)
imps <- Impute(fit, copy_reference(), 10)
```
ImputeMechanism.object

ImputeMechanism object

Description

An object which defines a mechanism for taking a SingleSimFit object and imputing missing data to create a ImputeSim

Arguments

name The method name (used for printing)
cols.needed which columns of the SingleSim data frame are required by the method, typically c("censored.time","observed.events","arm")
impute A function which takes a SingleSimFit object and outputs the details for a single imputed data set, specifically a list with two elements: new.censored.times - a vector of times subjects were censored (after taking into account imputation) and newevent.times - a list of vectors where the vectors contain the imputed event times for the subjects (these vectors do not contain the observed event times before subject drop out). If a subject has no imputed events then the vector numeric(0) is returned.
parameters A list of named parameters describing the method (used for printing) - or NULL if none

Details

It is possible to create user defined mechanisms, however, common mechanisms have already been implemented. For example see weighted_j2r

A print.ImputeMechanism method is defined.

Structure

The following components must be included in an ImputeMechanism Object

Examples

j2r <- weighted_j2r(trt.weight=0)
**ImputeSim.object**

**Description**

This object contains a collection of imputed data sets derived from a SingleSimFit object and ImputeMechanism

**Arguments**

- `singleSim`: The SingleSim object from which the imputed data sets have been derived
- `impute.mechanism`: The ImputeMechanism object used to perform the imputation
- `imputed.values`: A matrix with 1 column per imputed data set and two rows: `newevent.times` - a list of vectors containing the imputed event times (not including the events which were observed) and `new.censored.times` - a vector containing the times at which subjects (with imputed data) are now censored
- `dropout`: A vector containing the number of subjects who have dropped out in each arm, for whom data is to be imputed

Use `GetImputedDataSet` to extract a single imputed data set and use `Simfit` to fit a model to the set of data sets

**See Also**

- `GetImputedDataSet`

**ImputeSimFit.object**

**Description**

An object which contains both a set of imputed data sets (ImputeSim object) and a set of models fitted to them

**Arguments**

- `imputeSim`: The ImputeSim object for which models have been fitted
- `summaries`: A list of summary.SingleSimFit objects containing the model fits for each of the imputed data sets

**Details**

Calling `summary.ImputeSimFit` will apply Rubin's formula to calculate estimates for the treatment effect and standard error

Functions `summary.ImputeSimFit` and `as.data.frame.ImputeSimFit` have been implemented
LinearRateChangeDrop

Create a Dropout Mechanism with drop out rate which changes by a fixed constant after every event

Description

Creates an MAR DropoutMechanism object where subject i has piecewise exponential dropout rate where the rate changes by a constant amount after each event, specifically after j events the subject has rate \( R_{ij} = C_j \cdot \exp(X_{ij}) \) where \( C_j = C + j \cdot D \) for constants C, D and \( X_{ij} \) is a standard normal variable with mean 0 and standard deviation sigma

Usage

LinearRateChangeDrop(starting.rate, rate.change, var = 0)

Arguments

- starting.rate \( C \), see description section.
- rate.change \( D \), see description section. Note if \( D < 0 \), \( C_j \) could be negative for large j, this is not possible and the rate remains constant if the next change would set \( C_j \leq 0 \)
- var \( \sigma^2 \), see description section

Value

A DropoutMechanism object

See Also

DropoutMechanism.object

Examples

LinearRateChangeDrop(starting.rate=0.0025, rate.change=0.0005)
LinearRateChangeDrop(starting.rate=0.0025, rate.change=-0.00001, var=1)
MakeDejaData

Create a DejaData object

Description

This object is can be used to create a SingleSim object with subject specific rates

Usage

MakeDejaData(data, arm, Id, rate = NULL)

Arguments

data A data frame containing the subject
arm, character the column name of the treatment arm for each subject
Id, character the column name of subject Id
rate, character the column name of the rate to be used when simulating (or NULL, if using DejaData to import a data set, see ImportSim)

Value

A DejaData object

Examples

set.seed(232)

my.df <- data.frame(Id=1:100,
               arm=c(rep(0,50),rep(1,50)),
               covar=rbinom(n=100,size=1,prob=0.5))

my.df$rate <- 0.0025 + my.df$covar*0.002 + (1-my.df$arm)*0.002

my.dejaData <- MakeDejaData(my.df,arm="arm",rate="rate",Id="Id")

top numberSubjects

S3 generic to output the number of subjects in a given object

Description

S3 generic to output the number of subjects in a given object
Usage

   numberSubjects(x)

Arguments

   x  The object

Value

   The number of subjects

Examples

   sim <- SimulateComplete(study.time=365,number.subjects=50,
                          event.rates=c(0.01,0.005),dispersions=0.25)
   numberSubjects(sim)

Scenario.object  Scenario object

Description

   This class contains a collection of model fit summaries and summarizing this object will calculate
   overall summary statistics such as power/type I error

Arguments

   description  A string containing a description of the scenario
   summaries    A list of either summary.ImputeSimFit or summary.SingleSimFit objects

Details

   Functions as.data.frame.Scenario and summary.Scenario have been implemented

See Also

   CreateScenario
simData

Simulated recurrent event data.

Description

A simulated dataset containing a randomised treatment group, follow-up time, and number of events, for 500 patients. The planned follow-up period for the study was 1 year, but some patients dropped out early and so their follow-up ended prematurely (i.e. before 1 year).

Usage

simData

Format

A data frame with 500 rows and 3 variables:

- z: a binary variable indicating randomised treatment group
- y: number of events observed during patient's follow-up
- fupTime: the time in years the patient was followed up for...

Source

Simulated data

Simfit

S3 generic for fitting models

Description

S3 generic for fitting models

Usage

Simfit(x, family = "negbin", equal.dispersion = TRUE, covar = NULL, ...)

Arguments

- x: The S3 object
- family: Either "negbin" for fitting a negative binomial model (using MASS::glm.nb), "poisson" for fitting a poisson model (glm) or "quasipoisson" for fitting a quasipoisson model glm
- equal.dispersion: logical, should the arms have the same dispersion parameter when fitting negative binomial models
SimulateComplete

Simulate a complete data set

Description

Simulate a complete data set of a recurrent event clinical trial without dropouts using a negative binomial model with given rates and dispersion parameters.

Usage

```r
SimulateComplete(
  study.time,
  dejaData = NULL,
  number.subjects = NULL,
  event.rates = NULL,
  dispersions
)
```

Arguments

- `study.time`: The study follow up period.
- `dejaData`: If not NULL this should contain a DejaData object. If this is used then number.subjects and event.rates arguments are ignored.
SimulateDropout

number.subjects

The number of subjects, if a vector c(a,b) then a subjects on the control arm and b subjects on the active arm. If number.subjects is a single number then both arms have the given number of subjects.

event.rates

The rate parameter(s) for the negative binomial model (if single parameter then it is used for both arms)

dispersions

The dispersion parameter(s) for the negative binomial model (if single parameter then it is used for both arms)

Details

Each subject’s events are described by a Poisson process with a subject specific rate given by \( \lambda / study.time \) where \( study.time \) is the study follow up period and \( \lambda \) has a gamma distribution with shape=1/dispersion and scale=dispersion*event.rate*study.time

Different dispersions, event.rates and number of subjects can be specified for both arms of the trial

Value

A SingleSim object with status='complete'

See Also

SingleSim.object

Examples

```r
sim <- SimulateComplete(study.time=365, number.subjects=50, 
                       event.rates=c(0.01,0.005), dispersions=0.25)
sim2 <- SimulateComplete(study.time=365, number.subjects=c(50,75), 
                         event.rates=c(0.01,0.005), dispersions=c(0,0.25))
```

SimulateDropout

Simulate subject dropout

Description

This function takes a complete recurrent event data set and dropout mechanism and creates a data set set with dropout

Usage

SimulateDropout(simComplete, drop.mechanism)

Arguments

simComplete A SingleSim object (with status="complete")
drop.mechanism A DropoutMechanism object
SingleSim.object

Value

A SingleSim object with status='dropout'

Examples

```r
sim <- SimulateComplete(study.time=365,number.subjects=50,
event.rates=c(0.01,0.005),dispersions=0.25)
sim.with.MCAR.dropout <- SimulateDropout(sim,
drop.mechanism = ConstantRateDrop(rate = 0.0025))
sim.with.MAR.dropout <- SimulateDropout(sim,
drop.mechanism = LinearRateChangeDrop(
    starting.rate = 0.0025,
    rate.change = 0.0005))
```

Description

A class containing the data for a single simulation. Depending on the value of status, this may be a complete data set, a set including subject dropouts or a data set after multiple imputation. print.SingleSim and summary.SingleSim methods are defined.

Arguments

data The data frame, one row per subject containing (at least) the following columns Id, arm, censored.time, observed.events and actual.events

event.times A list of event times. event.times[[1]] is a list of event times for subject with Id 1 The length of event.times[[1]] = the number of observed events of subject with Id 1

status Either "complete", "dropout" or "imputed" denoting the status of the data set.

subject.rates A vector of the specific rates used for the Poisson process for subjects when generating the data

dropout.mechanism If status is not "complete" then this contains the DropoutMechanism object used to perform the subject dropout. See DropoutMechanism.object.

impute.mechanism If the status is "imputed" then this contains the ImputeMechanism object used to perform the imputation. See ImputeMechanism.object

study.time The study follow up period (see SimulateComplete)

event.rates The control/active event rates (see SimulateComplete), if data set was generated without using these (e.g. the dejaData argument was used) then this is set to NULL

dispersions The control/active dispersion rates (see SimulateComplete)
**Structure**

The above components must be included in a SingleSim Object

---

**SingleSimFit.object**  
**SingleSimFit object**

---

**Description**

A SingleSimFit object is returned from calling `Simfit` with a SingleSim object. It can be used to both impute data sets or can be summarized.

**Arguments**

- `singleSim` The SingleSim object to which a model has been fitted
- `model` The model which has been fitted
- `genCoeff.function` A function which returns a list of parameters from the model fit(s) which can be used when performing the gamma imputation. It takes one argument, `use.uncertainty` (by default is TRUE) which if TRUE stochastically incorporates uncertainty into the parameter estimates in preparation for use with imputation. If a Poisson/quasi-Poisson model was fitted to the SingleSimFit object then this will be NULL.
- `equal` dispersion whether equal dispersions were used when fitting model(s) to the data

**Details**

A `summary.SingleSimFit` method has been implemented.

---

**subjectsPerArm**  
*S3 generic to output the number of subjects in each arm for a given object*

---

**Description**

S3 generic to output the number of subjects in each arm for a given object

**Usage**

`subjectsPerArm(x)`

**Arguments**

- `x` The object
Value

A vector of the number of subjects in each arm

Examples

```r
sim <- SimulateComplete(study.time=365, number.subjects=50, event.rates=c(0.01,0.005), dispersions=0.25)
subjectsPerArm(sim)
```

Description

The summary of a `ImputeSimFit` object. Rubin’s formula is used to combine the test statistics into a single summary

Arguments

treatment.effect
  The mean of the estimated treatment.effect from the imputed data
se
  The standard error of the (log) treatment effect calculated using Rubin’s formula
df
  The number of degrees of freedom used to calculate the p-value
adjusted.df
  The number of degrees of freedom used to calculate the adjusted p-value (this should be used if the complete data number of degrees of freedom is small)
dispersion
  The mean of the estimated dispersion parameter
pval
  The p-value for the test log(treatment.effect)=0 using Rubin’s formula
adjusted.pval
  The p-value for the test log(treatment.effect)=0 using Rubin’s formula and the adjusted number of degrees of freedom
dropout
  The number of subjects who drop out (per arm) for this imputed data set
number.subjects
  The number of subjects (per arm) for this imputed data set

Details

A `print.summary.ImputeSimFit` object has been implemented
**description**

This object contains the overall summary statistics for a specific scenario. It is envisioned that multiple scenarios are run and a set of `summary.Scenario` objects are created and these can then be used for plotting.

**arguments**

- **treatment.effect**
  The \(\exp(\text{mean}(\log(\text{individual treatment effects})))\),

- **se**
  The mean standard error of the (log) treatment effect

- **power**
  The proportion of simulations for which the p-value is < \(\alpha\)

- **alpha**
  The significance level used when calculating power, by default 0.05 use `summary(object, alpha=x)` to use a different p value

- **use.adjusted.pval**
  logical, default FALSE should the p values calculated using Rubin’s formula with the adjusted number of degrees of freedom be used. Use `summary(object, use.adjusted.pval=TRUE)` to use the adjusted p values

- **description**
  A string containing a description of the scenario

- **dropout**
  A list of summary statistics regarding number of subject dropouts

**details**

A `print.summary.Scenario` function has been implemented.

**summary.SingleSim.object**

**description**

The object returned when calling the summary function on a `SingleSim` object.
summary.SingleSimFit

Arguments

status  The status of the SingleSim object
study.time  The study.time from the SingleSim object
number.subjects  The number of subjects on each arm
number.dropouts  The number of subjects who dropout on each arm
total.events  The total number of events for each arm
time.at.risk  The total time at risk for each arm
empirical.rates  
  total.events/time.at.risk
  The print.summary.SingleSim method has been implemented

summary.SingleSimFit

Description

The summary object for a SingleSimFit object

Arguments

model.summary  The model summary from the fit
treatment.effect  The estimate of treatment effect from the model fit
CI.limit  The confidence interval limit (by default 0.95), call summary(object, CI.limit=x) to use CI of x instead.
CI  The confidence interval of the treatment effect
se  Estimate for the standard error of (log) treatment effect
dispersion  Estimate for the dispersion parameter or numeric(0) if Poisson/quasi-Poisson model used
rate.estimate  Estimate of the event rates from the model a vector c(control arm, treatment arm)
pval  The p value directly from the model fit (this is for the single model fit only, i.e. not using Rubin's formula)
datastatus  The status of SingleSim object to which the fit was applied
df  The number of degrees of freedom of the model
dropout  The number of dropouts of each arm
number.subjects  The number of subjects in each arm
weighted_j2r

Details

A print.summary.SingleSimFit method has been implemented

See Also

SingleSimFit.object

Examples

```r
sim <- SimulateComplete(study.time=365,number.subjects=50,
event.rates=c(0.01,0.005),dispersions=0.25)
fit <- Simfit(sim)
summary(fit)
```

weighted_j2r

Create a weighted_j2r ImputeMechanism object

Description

Missing counts for a subject in the active treatment arm will be imputed according to a point (determined by \( trt.weight \)) between the means of the placebo and treatment arms, conditioned on the number of events. Missing counts for subjects in the placebo arm will be imputed according to the mean of the placebo arm, conditioned on the subject’s observed number of events.

Usage

```r
weighted_j2r(trt.weight, delta = c(1, 1), proper = TRUE)
```

Arguments

- `trt.weight`: See details
- `delta`: If \( trt.weight=1 \) then \( \delta \) is a vector of length 2 (control.delta,treatment.delta) and the mean number of expected events for the imputed missing data is multiplied by the appropriate \( \delta \)
- `proper`: If \( proper=TRUE \) then proper imputation is performed, in which each imputation is created based on parameters values drawn from the (approximate) posterior distribution of the imputation model. If \( proper=FALSE \), improper imputation is performed. This means all imputed datasets are generated conditional on the maximum likelihood estimates of the parameters.

Details

If \( trt.weight = 0 \) then imputation using this mechanism will follow the jump to reference (j2r) model whereby missing counts for subjects in both arms will be imputed according to the mean of the placebo arm conditioned on the subject’s observed number of events.
If `trt.weight = 1` then imputation using this mechanism will follow the MAR model whereby missing counts for subjects in each arm will be imputed according to the event rate of subjects in its treatment group conditioned on the subject's observed number of events.

See the User guide vignette for further details.

**Value**

An `ImputeMechanism` object

**See Also**

`ImputeMechanism.object`

**Examples**

```r
sim <- SimulateComplete(study.time=365, number.subjects=50, event.rates=c(0.01, 0.005), dispersions=0.25)
sim.with.MCAR.dropout <- SimulateDropout(sim, drop.mechanism = ConstantRateDrop(rate = 0.0025))
fit <- Simfit(sim.with.MCAR.dropout)
imps <- Impute(fit, weighted_j2r(trt.weight=0), 10)
```
Index

* datasets
  simData, 17
  ConstantRateDrop, 2, 7
  copy_reference, 3
  CreateNewDropoutMechanism, 4
  CreateNewImputeMechanism, 5
  CreateScenario, 6, 8, 16
  DejaData.object, 6
  DropoutMechanism.object, 3–5, 7, 14, 20
  expandEventCount, 7
  extract_results, 8
  GetImputedDataSet, 9, 13
  ImportSim, 9, 15
  Impute, 11
  ImputeMechanism.object, 3, 5, 12, 20, 26
  ImputeSim.object, 9, 13
  ImputeSimFit.object, 13
  LinearRateChangeDrop, 7, 14
  MakeDejaData, 6, 15
  numberSubjects, 15
  print.DropoutMechanism(DropoutMechanism.object), 7
  print.SingleSim(SingleSim.object), 20
  summary.ImputeSimFit.object, 22
  summary.Scenario.object, 23
  summary.SingleSim(SingleSim.object), 20
  summary.SingleSim.object, 23
  summary.SingleSimFit, 14, 21, 24
  weighted_j2r, 12, 25
  simData, 17
  Simfit, 17
  SimulateComplete, 18
  SimulateDropout, 19
  SingleSim.object, 19, 20
  SingleSimFit.object, 18, 21, 25
  subjectsPerArm, 21