

Package ‘gestate’

July 22, 2025

Title Generalised Survival Trial Assessment Tool Environment

Version 1.6.0

Description Provides tools to assist planning and monitoring of time-to-event trials under complicated censoring assumptions and/or non-proportional hazards. There are three main components: The first is analytic calculation of predicted time-to-event trial properties, providing estimates of expected hazard ratio, event numbers and power under different analysis methods. The second is simulation, allowing stochastic estimation of these same properties. Thirdly, it provides parametric event prediction using blinded trial data, including creation of prediction intervals. Methods are based upon numerical integration and a flexible object-orientated structure for defining event, censoring and recruitment distributions (Curves).

Depends R (>= 4.0.0)

Imports foreach, doParallel, shiny, shinythemes, survival, methods

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author James Bell [aut, cre],
Jasmin Ruehl [ctb]

Maintainer James Bell <james.bell.ext@boehringer-ingenelheim.com>

Repository CRAN

Date/Publication 2023-04-26 11:02:30 UTC

Contents

analyse_sim	3
Blank	6
createRFfunction	7
createRFfunction,Curve-method	7
create_tte_prior	8

Curve-class	9
evaluateCDFfunction	10
evaluateCDFfunction,Curve-method	10
evaluateInvfunction	11
evaluateInvfunction,Curve-method	11
evaluatePDFfunction	12
evaluatePDFfunction,Curve-method	12
event_prediction	13
event_prediction_KM	16
event_prediction_prior	18
Exponential	22
fit_KM	23
fit_tte_data	25
fit_tte_data_prior	26
frontierpower	28
getAssessCDFfunction	29
getAssessCDFfunction,RCurve-method	30
getCDFfunction	30
getCDFfunction,Curve-method	31
getInvfunction	31
getInvfunction,Curve-method	32
getLength	32
getLength,RCurve-method	33
getMaxF	33
getMaxF,RCurve-method	34
getN	34
getN,RCurve-method	35
getNactive	35
getNactive,RCurve-method	36
getNames	36
getNames,Curve-method	37
getNcontrol	37
getNcontrol,RCurve-method	38
getParam	38
getParam,Curve-method	39
getParams	39
getParams,Curve-method	40
getParamsV	40
getParamsV,Curve-method	41
getPatients	41
getPatients,RCurve-method	42
getPDFfunction	42
getPDFfunction,Curve-method	43
getRatio	43
getRatio,RCurve-method	44
getRFfunction	44
getRFfunction,Curve-method	45
getType	45

getType,Curve-method	46
GGamma	46
Gompertz	47
InstantR	48
LinearR	48
LogLogistic	49
Lognormal	50
MixExp	51
MixWei	52
nph_traj	53
PieceExponential	56
PieceR	57
PieceRMaxF	58
plotCDF	59
plotCDF,Curve-method	59
plotRecruitment	60
plotRecruitment,RCurve-method	61
plotSF	62
plotSF,Curve-method	62
plot_ep	63
plot_km_fit	65
plot_npht	67
random_draw	69
random_draw,Curve-method	69
RCurve-class	70
run_gestate	71
setPatients	71
setPatients,RCurve-method	72
set_assess_time	72
set_event_number	74
show,Curve-method	75
show,RCurve-method	76
simulate_trials	76
simulate_trials_strata	78
summarise_analysis	81
survfit_fast	83
Weibull	84
Index	86

Description

Function for analysing simulated time-to-event trial data produced by `simulate_trials()` or `simulate_trials_strata()`.

This function automatically reads in either list or matrix `simulate_trials()` data formats. It performs log rank test and Cox regression analysis by default, but can also/instead perform RMST and/or landmark analyses. Covariate adjusted (/ stratified) analysis may be selected by using the "stratum" argument. If a stratum is specified, it will be included as a covariate in Cox and RMST analysis, and as a stratum in a stratified log-rank test and an inverse-precision-weighted landmark test. Strata values are handled as factors, so continuous covariates are not supported.

Analysis is typically the slowest part of simulation studies, so parallel processing using the `doParallel` package is built in. Parallel processing is enabled by setting the number of cores in the "parallel_cores" argument. Use of parallel processing is recommended for largescale (e.g. 100,000 iteration) simulations. To avoid unnecessary issues, ensure that the number of cores specified does not exceed number of threads provided by hardware.

Usage

```
analyse_sim(
  data,
  LR = TRUE,
  RMST = NA,
  landmark = NA,
  stratum = "",
  parallel_cores = 1,
  Time = "Time",
  Event = "Censored",
  censoringOne = TRUE,
  Trt = "Trt",
  Iter = "Iter"
)
```

Arguments

data	Output file from <code>simulate_trials()</code> . Only <code>simulate_trials()</code> or <code>simulate_trials_strata()</code> output is supported, in either "list" or "matrix" format.
LR	Requests log-rank test and Cox regression. Default=TRUE
RMST	Requests Restricted Mean Survival Time analysis with specified (positive integer) restriction time, leave NULL for no analysis. Default=NULL (no RMST analysis).
landmark	Requests Landmark analysis at specified (positive integer) time, leave NULL for no analysis. Default=NULL (no landmark analysis).
stratum	Specify name of column of a stratification factor and turn on stratified (LR/LM) and covariate-adjusted (Cox/RMST) analysis. By default, "", and no stratification.
parallel_cores	Positive integer specifying number of cores to use. If 1 specified then no parallel processing. Default=1 (no parallel processing).
Time	String specifying the name of the time column. Default="Time"

Event	String specifying the name of the censoring/event column. Default="Censored" (and by default it is a censoring column unless censoringOne=FALSE)
censoringOne	Boolean specifying whether censoring is denoted in the censoring/event column by a one (TRUE) or zero (FALSE). Default=TRUE (censorings denoted by 1, events by 0)
Trt	String specifying the name of the treatment column. Default="Trt"
Iter	String specifying the name of the iterations column. Default="Iter"

Value

Returns a table with one row per simulation. Table contains the following columns:

- "HR" Cox Hazard Ratio (LR/Cox analysis only)
- "LogHR" Cox Log Hazard Ratio (LR/Cox analysis only)
- "LogHR_SE" Cox Standard Error of log Hazard Ratio (LR/Cox analysis only)
- "HR_Z" Cox Z-Score (LR/Cox analysis only)
- "HR_P" 1-sided Cox p-value (LR/Cox analysis only)
- "LR_Z" Log-Rank Test Z-Score (LR/Cox analysis only)
- "LR_P" 1-sided Log-Rank Test p-value (LR/Cox analysis only)
- "Events_Active" Events in Active arm (LR/Cox analysis only)
- "Events_Control" Events in Control arm (LR/Cox analysis only)
- "RMST_Time" RMST restriction time (RMST analysis only)
- "RMST_Active" RMST for Active arm (RMST analysis only)
- "RMST_Active_SE" RMST Standard Error for Active arm (RMST analysis only)
- "RMST_Control" RMST for Control arm (RMST analysis only)
- "RMST_Control_SE" RMST Standard Error for Control arm (RMST analysis only)
- "RMST_Delta" RMST difference between arms active-control (RMST analysis only)
- "RMST_Delta_SE" RMST difference between arms Standard Error (RMST analysis only)
- "RMST_Z" Z-score for RMST (RMST analysis only)
- "RMST_P" 1-sided RMST p-value (RMST analysis only)
- "LM_Time" Landmark time, i.e. time of survival function comparison (Landmark analysis only)
- "LM_Active" Survival function for active arm at landmark time (Landmark analysis only)
- "LM_Active_SE" Greenwood standard error for active arm at landmark time (Landmark analysis only)
- "LM_Control" Survival function for control arm at landmark time (Landmark analysis only)
- "LM_Control_SE" Greenwood standard error for control arm at landmark time (Landmark analysis only)
- "LM_Delta" Survival function difference between arms active-control at landmark time (Landmark analysis only)
- "LM_Delta_SE" Greenwood standard error for difference between arms at landmark time (Landmark analysis only)
- "LM_Z" Z-score for landmark analysis (Landmark analysis only)
- "LM_P" 1-sided landmark analysis p-value (Landmark analysis only)

Author(s)

James Bell

References

Uno H, Claggett B, Tian L, Inoue E, Gallo P, Miyata T, Schrag D, Takeuchi M, Uyama Y, Zhao L, Skali H, Solomon S, Jacobus S, Hughes M, Packer M, Wei LJ. Moving beyond the hazard ratio in quantifying the between-group difference in survival analysis. *Journal of clinical Oncology* 2014,32, 2380-2385. Tian L, Zhao L, Wei LJ. Predicting the restricted mean event time with the subjects baseline covariates in survival analysis. *Biostatistics* 2014, 15, 222-233.

Examples

```
example_sim <- simulate_trials(active_ecurve=Weibull(250,0.8),control_ecurve=Weibull(100,1),
rcurve=LinearR(12,100,100), assess=20,iterations=100,seed=12345,detailed_output=TRUE)

example_analysis1 <- analyse_sim(example_sim)
example_analysis2 <- analyse_sim(data=example_sim,RMST=15,landmark=15)

example_strat_sim <- simulate_trials_strata(stratum_probs=c(0.5,0.5),
active_ecurve=c(Weibull(250,0.8),Weibull(100,1)), control_ecurve=Weibull(100,1),
rcurve=LinearR(12,100,100),assess=20,iterations=100,seed=12345)

example_strat_analysis <- analyse_sim(data=example_strat_sim,RMST=15,landmark=15,stratum="Stratum")
```

Blank

*Blank Curve constructor function***Description**

This creates a Curve object for a 'Blank' pseudo-distribution.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

This distribution is 0 by definition for all times. It is not therefore a true probability distribution.

Usage

```
Blank()
```

Details

The blank pseudo-distribution is used for impossible events, notably where censoring is not possible/allowed.

$$f(x) = 0$$

$$F(x) = 0$$
Author(s)

James Bell

Examples

```
Blank()
```

createRFfunction	<i>Method for creating a random draw function from a Curve object</i>
------------------	---

Description

This creates a random draw function from the Curve object

Usage

```
createRFfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
createRFfunction(Weibull(100,1))
```

createRFfunction, Curve-method
<i>Method for creating a random draw function from a Curve object</i>

Description

This creates a random draw function from the Curve object

Usage

```
## S4 method for signature 'Curve'  
createRFfunction(theObject, n = "n")
```

Arguments

theObject	The name of the Curve Object
n	The parameter name for the number of random draws to make. Default=n

Examples

```
createRFfunction(Weibull(100,1))
```

create_tte_prior

Create an arbitrary prior data set from a specified Curve object

Description

This is a function to create a patient-level prior data set from a specified Curve.

It can be used to create a prior data set where only summary parameters are known. It requires a 'curve' object containing the desired distribution, the 'time' over which events occur and the number of 'events'. Designed to be used in conjunction with event prediction methods. Note that the output can be 'too perfect', so that when the output is used for curve-fitting it may result in over-precision. To minimise this effect, it may better to derive the prior data set using a different distribution to the one later used for fitting or specifying down-weighting.

Usage

```
create_tte_prior(
  curve,
  duration,
  events,
  Time = "Time",
  Event = "Event",
  censoringOne = FALSE
)
```

Arguments

curve	Curve object with the desired prior distribution
duration	The positive number for the time over which events should occur. This controls data maturity. When used as a prior, the longer the time, the more information is provided.
events	The positive integer for the number of events that should occur. The more events, the more information contained when used as a prior.
Time	The output column name for the times. Default is "Time"
Event	The output column name for the events column (i.e. the binary variable denoting events vs censorings). Default is "Event"
censoringOne	Specify whether output censoring is denoted in the Event column by a one (TRUE) or zero (FALSE). Default=FALSE (censorings denoted by 0, events by 1)

Value

A two-column data frame containing times in the first column and an event/censoring indicator in the second.

Author(s)

James Bell

Examples

```
example_prior <- create_tte_prior(curve=Weibull(100,0.8),duration=20,events=50)
```

Curve-class

*Curve Class for defining distributions***Description**

This class allows distributions to be defined. It contains all information needed to reproduce a distribution.

References to functions that store the PDF, CDF and random number generator.

Parameters are also stored.

Slots

type Type of Curve (character). Typically the distribution name.

PDF Name of the PDF function describing the Curve.

CDF Name of the CDF function describing the Curve.

RF Name of the random generator function describing the Curve.

inverse Name of the inverse CDF function describing the Curve. Optional; set to NULL if unavailable.

paramno Number of parameters required to define the distribution.

pnames Names of parameters defining the distribution. Should be a vector of length paramno.

pvalue Values of parameters defining the distribution. Should be a list of length paramno.

Author(s)

James Bell

Examples

```
new("Curve", type="ExampleCurve",PDF="pdf_fn_name",CDF="CDF_fn_name",
    RF="random_draw_fn_name", inverse="inv_fn_name",paramno=2,pnames=c('param1','param2'),
    pvalue=list(1,2))
```

evaluateCDFfunction	<i>Method for evaluating the CDF function for a Curve object at q</i>
---------------------	---

Description

This numerically evaluates the CDF function of the Curve object at the specified q

Usage

```
evaluateCDFfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
evaluateCDFfunction(Weibull(100,1),10)
```

evaluateCDFfunction, Curve-method	<i>Method for evaluating the CDF function for a Curve object at q</i>
-----------------------------------	---

Description

This numerically evaluates the CDF function of the Curve object at the specified q

Usage

```
## S4 method for signature 'Curve'
evaluateCDFfunction(theObject, q)
```

Arguments

theObject	The name of the Curve Object
q	The time to evaluate at

Examples

```
evaluateCDFfunction(Weibull(100,1),10)
```

evaluateInvfunction	<i>Method for evaluating the inverse-CDF function for a Curve object at p</i>
---------------------	--

Description

This numerically evaluates the inverse-CDF function of the Curve object at the specified p

Usage

```
evaluateInvfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
evaluateInvfunction(Weibull(100,1), 0.5)
```

evaluateInvfunction, Curve-method	<i>Method for evaluating the inverse-CDF function for a Curve object at p</i>
-----------------------------------	--

Description

This numerically evaluates the inverse-CDF function of the Curve object at the specified p

Usage

```
## S4 method for signature 'Curve'
evaluateInvfunction(theObject, p)
```

Arguments

theObject	The name of the Curve Object
p	The probability to evaluate the time at

Examples

```
evaluateInvfunction(Weibull(100,1), 0.5)
```

evaluatePDFfunction	<i>Method for evaluating the PDF function for a Curve object at x</i>
---------------------	---

Description

This numerically evaluates the PDF function of the Curve object at the specified x

Usage

```
evaluatePDFfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
evaluatePDFfunction(Weibull(100,1),10)
```

evaluatePDFfunction, Curve-method	<i>Method for evaluating the PDF function for a Curve object at x</i>
-----------------------------------	---

Description

This numerically evaluates the PDF function of the Curve object at the specified x

Usage

```
## S4 method for signature 'Curve'
evaluatePDFfunction(theObject, x)
```

Arguments

theObject	The name of the Curve Object
x	The time to evaluate at

Examples

```
evaluatePDFfunction(Weibull(100,1),10)
```

event_prediction	<i>Event prediction using patient-level survival data and a recruitment RCurve</i>
------------------	--

Description

This is a function to perform event prediction

It uses the `fit_KM_tte_data` function to perform MLE regression of Weibull and log-normal curves to the provided survival data.

It creates an event Curve object from this, and combines it with a recruitment RCurve and an optional dropout(censoring) Curve.

Using the same numerical integration approach as `nph_curve_trajectories` it performs an unconditional event prediction.

If a conditioning time and event number (preferably also a number at risk) are provided, a conditional event prediction is also calculated.

Analytic standard errors for conditional and unconditional event numbers are provided for the whole trajectory.

SEs calculated by propagating parameter estimate errors through the integrals by the delta method and then invoking a beta-binomial distribution.

For event prediction, conditional predictions with the Conditional SE of Prediction are most accurate and appropriate.

Unconditional predictions should be close to conditional ones but technically relate to predictions if the trial were rerun, rather than this specific instance. Point estimates are usually very close to the unconditional ones, but the prediction intervals are typically much wider than necessary. The conditional and unconditional SEs of fitting relate to the accuracy of the estimated mean event number at a given time, rather than the spread of future observations. The conditional and unconditional SEs of prediction relate to the accuracy of prediction of future observations, and should therefore be used for event prediction. Note that the Prediction SEs are wider than the Fitting SEs as they also take into account the binomial uncertainty of events occurring (beta-binomial model). As of version 1.4.0, the 'CI' argument has been renamed 'PI', and the 'condition' argument has been removed entirely (conditioning automatically occurs if `cond_Event` specified).

Usage

```
event_prediction(
  data,
  Time = "Time",
  Event = "Event",
  censoringOne = FALSE,
  type = c("automatic", "Exponential", "Weibull", "Lognormal"),
  rcurve,
  max_time = 100,
  dcurve = Blank(),
  CI = NULL,
  PI = 0.95,
  condition = NULL,
```

```

    cond_Events = NULL,
    cond_NatRisk = NULL,
    cond_Time = NULL,
    units = c("Days", "Months"),
    init = NULL,
    discountHR = 1
)

```

Arguments

<code>data</code>	The dataframe object containing the patient-level survival data
<code>Time</code>	The column name for the times. Default is "Time"
<code>Event</code>	The column name for the events column (i.e. the binary variable denoting events vs censorings). Default is "Event"
<code>censoringOne</code>	Specify whether censoring is denoted in the Event column by a one (TRUE) or zero (FALSE). Default=FALSE (censorings denoted by 0, events by 1)
<code>type</code>	Type of event curve to fit. Default is "Automatic", fitting both Weibull and Log-normal curves. Alternatively accepts "Weibull", "Lognormal" or "Exponential" to force the type.
<code>rcurve</code>	Observed and/or expected recruitment distribution as an RCurve object. This should typically be of PieceR type (piecewise linear recruitment).
<code>max_time</code>	Maximum time to predict events up to.
<code>dcurve</code>	Dropout/censoring distribution as a Curve object. This is Blank() by default, i.e. no dropout.
<code>CI</code>	DEPRECATED As of version 1.4.0 this argument has been renamed to 'PI'.
<code>PI</code>	Number between 0 and 1 for the size of prediction interval to calculate. As of 1.4.0 this replaces the 'CI' argument. Default is 0.95 (95 percent prediction interval).
<code>condition</code>	DEPRECATED This argument has been removed as of version 1.4.0. Specifying a value for 'cond_Events' will now enable conditioned predictions.
<code>cond_Events</code>	Number of observed events at the conditioning time to condition on. If NULL, no conditioned event prediction will be performed. Default=NULL (no conditioning). Note that if the discountHR option is used to predict adjusted event numbers, it would be possible to condition on either observed or adjusted events, but the observed number is required by this function.
<code>cond_NatRisk</code>	Number of patients at risk to condition on. Default=NULL. By default, the program will estimate the number at risk assuming no censoring. It is highly recommended to specify this if conditioning.
<code>cond_Time</code>	Time, in months, to condition on. A non-negative integer less than max_time is required if conditioning is requested, i.e. cond_Events is non-NULL. Not required otherwise.
<code>units</code>	Units that the KM-curve is specified in. Accepts "Days", "Months". Default="Days". Note: gestate assumes conversion factor of 365.25/12.
<code>init</code>	Vector of starting values for parameter values; useful if survreg experiences convergence issues. Default=NULL (no values specified)

discountHR Hazard ratio for discounting events e.g. used to predict adjudicated events from unadjudicated data where patients remain 'at risk' after an event is adjudicated not to have occurred. Values below 1 indicate fewer events will occur than predicted by the curve-fitting. When a discountHR is user-specified (i.e. not 1), conditioning event numbers need to be specified in terms of observed values, and not adjusted ones. Note that changing this argument is only allowed if type="Weibull" since log-normal curves are not compatible with proportional hazards. If patients become not at risk following a failed adjudication (i.e. removed from study), do not use this argument and instead adjust the output event numbers by the required factor. Default=1 (No discounting for adjudication)

Value

Returns a list object with the prediction ecurve (after adjustments for unit, discountHR), dcurve, rcurve, required PI, original fitted ecurve parameters (before adjustments) and a summary table with one row per month up to max_time containing the following columns:

- "Assessment_Time"Time of assessment.
- "Patients"Number of patients recruited by the assessment time.
- "Predicted_Events"Number of events unconditionally predicted at the assessment time.
- "SE_Fitting"SE of the estimate of the fitted mean. Note that this corresponds to the accuracy of the estimate of the underlying parameter, not future observed event numbers.
- "SE_Prediction"SE of event prediction.
- "Prediction_Lower"Lower bound of X percent interval of unconditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.
- "Prediction_Upper"Upper bound of X percent interval of unconditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.
- "Conditioned_Events"Number of events conditionally predicted at the assessment time (Column present only if conditioning specified).
- "Cond_SE_Fitting"SE of the estimate of the fitted conditional mean. Note that this corresponds to the accuracy of the estimate of the underlying parameter, not future observed event numbers (Column present only if conditioning specified).
- "Cond_SE_Prediction"SE of the conditional event prediction (Column present only if conditioning specified).
- "Cond_Prediction_Lower"Lower bound of X percent interval of conditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.
- "Cond_Prediction_Upper"Upper bound of X percent interval of conditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.

Author(s)

James Bell

References

Bell J, "Are we nearly there yet?" Quantifying uncertainty in event prediction, 2019, presentation at PSI Conference.

Examples

```
recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
trial_short <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit, assess=10,iterations=1,seed=12345,detailed_output=TRUE)

predictions <- event_prediction(data=trial_short, Event="Censored", censoringOne=TRUE,
type="Weibull", rcurve=recruit, max_time=60, cond_Events=49, cond_NatRisk=451,
cond_Time=10, units="Months")
```

event_prediction_KM	<i>Event prediction using a lifetable and a recruitment RCurve</i>
---------------------	--

Description

This is a function to perform event prediction using summary-level data. As of 1.4.0 this function is deprecated as event_prediction gives improved accuracy.

It uses the fit_KM function to perform non-linear regression of Weibull and log-normal curves to the provided survival data.

It creates an event Curve object from this, and combines it with a recruitment RCurve and an optional dropout(censoring) Curve.

Using the same numerical integration approach as nph_curve_trajectories it performs an unconditional event prediction.

If a conditioning time, event number (and preferably number at risk) are provided, a conditional event prediction is also calculated.

Usage

```
event_prediction_KM(
  KMcurve,
  Survival = "Survival",
  Time = "Time",
  weighting = FALSE,
  Weights = "Weights",
  Weight_power = 1,
  rcurve,
  max_time = 100,
  dcurve = Blank(),
  type = c("automatic", "Weibull", "Lognormal", "Exponential"),
  startbeta = 1,
  startsigma = 1,
  condition = FALSE,
```

```

    cond_Events = 0,
    cond_NatRisk = NULL,
    cond_Time = 0,
    units = c("Days", "Months"),
    discountHR = 1
)

```

Arguments

KMcurve	The dataframe object containing the survival data
Survival	The column name for the survival function (i.e. the probabilities). Default is "Survival"
Time	The column name for the times. Default is "Time" Alternatively accepts "Weibull" or "Lognormal" to force the type.
weighting	Boolean for whether to use weighting. Default=TRUE as it greatly improves curve fitting.
Weights	Name of Weights column. Default="Weights". Optional if weighting=FALSE. Recommended to use number at risk or remaining.
Weight_power	Power to raise the weights to. Useful in large trials to give added weight to later points where numbers may still be high. Default=1 (Use weights as specified).
rcurve	Observed and/or expected recruitment distribution as an RCurve object. This should typically be of PieceR type (piecewise linear recruitment).
max_time	Maximum time to predict events up to.
dcurve	Dropout/censoring distribution as a Curve object. This is Blank() by default, i.e. no dropout.
type	Type of event curve to fit. Default is "Automatic", fitting both Weibull and Log-normal curves.
startbeta	Starting value for the Weibull beta (shape) parameter to be used in the non-linear regression. Default=1 (exponential).
startsigma	Starting value for the Lognormal sigma (sd) parameter to be used in the non-linear regression. Default=1.
condition	Boolean whether to also do a conditional event prediction. Default=FALSE Note that If all conditioning options are left as defaults, conditioned calculation will equal the unconditional one.
cond_Events	Number of events to condition on. Default=0. Optional unless condition=TRUE.
cond_NatRisk	Number of patients at risk to condition on. By default, the program will estimate the number at risk assuming no censoring. It is highly recommended to specify this if conditioning. Default=NULL(takes value of N - cond_Events). Optional unless condition=TRUE.
cond_Time	Time, in months, to condition on. Default=0. Optional unless condition=TRUE.
units	Units that the KM-curve is specified in. Accepts "Days", "Months". Default="Days".
discountHR	Hazard ratio for discounting events e.g. used to predict adjudicated events from unadjudicated data where patients remain 'at risk' after an event is adjudicated

not to have occurred. Values below 1 indicate fewer events will occur than predicted by the curve-fitting. Note that changing this argument is only allowed if type="Weibull" since log-normal curves are not compatible with proportional hazards. Default=1 (No discounting)

Value

Returns a list object with the fitted ecurve, the dcurve, the rcurve, the fitting details, and a summary table with one row per month up to max_time containing the following columns:

- "Time"Time of assessment.
- "Patients"Number of patients recruited by the assessment time.
- "Predicted_Events"Number of events unconditionally predicted at the assessment time.
- "Conditioned_Events"Number of events unconditionally predicted at the assessment time (Column present only if conditioning specified).

Author(s)

James Bell

Examples

```
recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
example_data_short <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit, assess=10,iterations=1,seed=12345,detailed_output=TRUE)

library(survival)

temp1 <- summary(survfit(Surv(example_data_short[, "Time"],1-example_data_short[, "Censored"])~ 1,
error="greenwood"))
out1 <- cbind(temp1$time,temp1$n.risk,temp1$urv,temp1$std.err)
out1 <- rbind(c(0,out1[1,2],1,0),out1)
colnames(out1) <- c("Time","NAR","Survival","Std.Err")
x1 <- ceiling(max(out1[, "Time"]))
example_lifetable <- out1[findInterval(0:x1,out1[, "Time"]),]
example_lifetable[, "Time"] <- 0:x1

event_prediction_KM(KMcurve=example_lifetable, weighting=TRUE, Weights="NAR", rcurve=recruit,
max_time=60, type="automatic", condition=TRUE, cond_Events=49, cond_NatRisk=451, cond_Time=10,
units="Months")
```

event_prediction_prior

Event prediction using patient-level survival data, prior data and a recruitment RCurve

Description

This performs event prediction with a Weibull distribution integrating (weighted) prior data . Where relevant prior data is available, this function can increase the precision of curve fitting, particularly in cases where there is low data maturity. This function uses the `fit_tte_data_prior` function to fit a Weibull distribution to the provided survival data by including weighted prior patient-level data. An event Curve object is created, which is then used in the same way as that from the frequentist approach in `event_prediction`. It consequently operates very similarly to `event_prediction` in terms of inputs, outputs and methods. Consult its documentation for aspects unrelated to the prior approach. If a patient-level prior data set is not available, use the `create_tte_prior` function to create an artificial one from a distribution. Note that this may cause a small amount of over-precision; this may be minimised by deriving the prior data using a different distribution or specifying down-weighting. Prior data can be down-weighted using the `priorweight` variable. Some degree of down-weighting is generally recommended to reflect differences with prior trials. Column names and the censoring parity in the prior data set are by default the same as those specified for the main data set, but may be manually changed to be different.

Usage

```
event_prediction_prior(
  data,
  Time = "Time",
  Event = "Event",
  censoringOne = FALSE,
  priordata,
  priorTime = Time,
  priorEvent = Event,
  priorcensoringOne = censoringOne,
  priorweight = 1,
  type = c("Weibull"),
  rcurve,
  max_time = 100,
  dcurve = Blank(),
  CI = NULL,
  PI = 0.95,
  condition = NULL,
  cond_Events = NULL,
  cond_NatRisk = NULL,
  cond_Time = NULL,
  units = c("Days", "Months"),
  init = NULL,
  discountHR = 1
)
```

Arguments

<code>data</code>	The dataframe object containing the patient-level survival data
<code>Time</code>	The column name for the times in the 'data' set. Default is "Time"
<code>Event</code>	The column name for the events column (i.e. the binary variable denoting events vs censorings) in the 'data' set. Default is "Event"
<code>censoringOne</code>	Specify whether in the 'data' set censoring is denoted by a one (TRUE) or zero (FALSE). Default=FALSE (censorings denoted by 0, events by 1)
<code>priordata</code>	The dataframe object containing the patient-level survival data for the prior
<code>priorTime</code>	The column name for the times in the 'priordata' set. Default= Value specified for Time
<code>priorEvent</code>	The column name for the events column (i.e. the binary variable denoting events vs censorings) in the 'priordata' set. Default= Value specified for Event
<code>priorcensoringOne</code>	Specify whether in the 'priordata' set censoring is denoted by a one (TRUE) or zero (FALSE). Default= Value specified for censoringOne
<code>priorweight</code>	The weight that should be allocated to each patient's data in the prior data, typically between 0 and 1. 0 implies prior data is ignored, 1 that prior patient data is given full weight, i.e. prior patients are exchangeable with observed patients. Default = 1
<code>type</code>	Type of event curve to fit. Default is "Automatic", fitting both Weibull and Log-normal curves. Alternatively accepts "Weibull", "Lognormal" or "Exponential" to force the type.
<code>rcurve</code>	Observed and/or expected recruitment distribution as an RCurve object. This should typically be of PieceR type (piecewise linear recruitment).
<code>max_time</code>	Maximum time to predict events up to.
<code>dcurve</code>	Dropout/censoring distribution as a Curve object. This is Blank() by default, i.e. no dropout.
<code>CI</code>	DEPRECATED As of version 1.4.0 this argument has been renamed to 'PI'.
<code>PI</code>	Number between 0 and 1 for the size of prediction interval to calculate. As of 1.4.0 this replaces the 'CI' argument. Default is 0.95 (95 percent prediction interval).
<code>condition</code>	DEPRECATED This argument has been removed as of version 1.4.0. Specifying a value for 'cond_Events' will now enable conditioned predictions.
<code>cond_Events</code>	Number of observed events at the conditioning time to condition on. If NULL, no conditioned event prediction will be performed. Default=NULL (no conditioning). Note that if the discountHR option is used to predict adjusted event numbers, it would be possible to condition on either observed or adjusted events, but the observed number is required by this function.
<code>cond_NatRisk</code>	Number of patients at risk to condition on. Default=NULL. By default, the program will estimate the number at risk assuming no censoring. It is highly recommended to specify this if conditioning.
<code>cond_Time</code>	Time, in months, to condition on. A non-negative integer less than max_time is required if conditioning is requested, i.e. cond_Events is non-NULL. Not required otherwise.

units	Units that the KM-curve is specified in. Accepts "Days", "Months". Default="Days".
init	Vector of starting values for parameter values; useful if survreg experiences convergence issues. Default=NULL (no values specified)
discountHR	Hazard ratio for discounting events e.g. used to predict adjudicated events from unadjudicated data where patients remain 'at risk' after an event is adjudicated not to have occurred. Values below 1 indicate fewer events will occur than predicted by the curve-fitting. When a discountHR is user-specified (i.e. not 1), conditioning event numbers need to be specified in terms of observed values, and not adjusted ones. Note that changing this argument is only allowed if type="Weibull" since log-normal curves are not compatible with proportional hazards. If patients become not at risk following a failed adjudication (i.e. removed from study), do not use this argument and instead adjust the output event numbers by the required factor. Default=1 (No discounting for adjudication)

Value

Returns a list object with the prediction ecurve (after adjustments for unit, discountHR), dcurve, rcurve, required PI, original fitted ecurve parameters (before adjustments) and a summary table with one row per month up to max_time containing the following columns:

- "Assessment_Time"Time of assessment.
- "Patients"Number of patients recruited by the assessment time.
- "Predicted_Events"Number of events unconditionally predicted at the assessment time.
- "SE_Fitting"SE of the estimate of the fitted mean. Note that this corresponds to the accuracy of the estimate of the underlying parameter, not future observed event numbers.
- "SE_Prediction"SE of event prediction.
- "Prediction_Lower"Lower bound of X percent interval of unconditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.
- "Prediction_Upper"Upper bound of X percent interval of unconditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.
- "Conditioned_Events"Number of events conditionally predicted at the assessment time (Column present only if conditioning specified).
- "Cond_SE_Fitting"SE of the estimate of the fitted conditional mean. Note that this corresponds to the accuracy of the estimate of the underlying parameter, not future observed event numbers (Column present only if conditioning specified).
- "Cond_SE_Prediction"SE of the conditional event prediction (Column present only if conditioning specified).
- "Cond_Prediction_Lower"Lower bound of X percent interval of conditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.
- "Cond_Prediction_Upper"Upper bound of X percent interval of conditional event prediction, where X is the 'PI' argument. This PI is based on the quantiles of the beta-binomial distribution and so is discrete and asymmetric.

Author(s)

James Bell

References

Bell J, unpublished work.

Examples

```

recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
trial_short <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit, assess=10,iterations=1,seed=12345,detailed_output=TRUE,Event="Event",
censoringOne=FALSE)
example_prior <- create_tte_prior(curve=Weibull(alpha=100,beta=0.8),duration=20,events=50)

predictions <- event_prediction_prior(data=trial_short, priordata=example_prior,
type="Weibull", rcurve=recruit, max_time=60, cond_Events=49, cond_NatRisk=451,
cond_Time=10, units="Months")

```

Exponential

Exponential Curve constructor function

Description

This creates a Curve object for a Exponential distribution.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

Parameterisation follows that used by pexp etc. Note that $\lambda = 1/\alpha$ from the Weibull constructor. See Details for more information on parameterisation.

Usage

```
Exponential(lambda)
```

Arguments

`lambda` Rate parameter for Exponential distribution.

Details

The exponential distribution has parameterisation:

$$f(x) = \lambda e^{-\lambda x}$$

$$F(x) = 1 - e^{-\lambda x}$$

Author(s)

James Bell

Examples

```
Exponential(0.01)
```

fit_KM	<i>Fit Weibull, Log-Normal or Exponential survival curves to Kaplan Meier estimates</i>
--------	---

Description

This is a function to fit Weibull and log-normal curves to Survival data in life-table form using non-linear regression.

By default it fits both, then picks the best fit based on the lowest (un)weighted residual sum of squares.

Alternatively, just one shape may be fitted, by changing the 'type' argument to either "Weibull" or "Lognormal". Weighted or unweighted fitting are possible. In general, weighted fitting using the number at risk as the weights seems to work best.

This function is primarily used by event_prediction function, but also useful for general KM curve fitting.

One useful aspect of this is for fitting the 'inverse KM', where drop-outs are events, while events and 'time-outs' are censored. This allows for finding a suitable parameterisation for the censoring curve.

Primary advantage over likelihood-based methods is ability to use aggregated, rather than patient-level data. Primary disadvantage is that the covariance matrix is unusable due to strong correlation between the input data points going into the regression.

Usage

```
fit_KM(
  KMcurve,
  Survival = "Survival",
  Time = "Time",
  type = c("automatic", "Weibull", "Lognormal", "Exponential"),
  weighting = TRUE,
  Weights = "Weights",
  Weight_power = 1,
  startbeta = 1,
  startsigma = 1
)
```

Arguments

KMcurve	The dataframe object containing the survival data in lifetable form
Survival	The column name for the survival function (i.e. the probabilities). Default is "Survival"
Time	The column name for the times. Default is "Time"

type	Type of event curve to fit. Default is "Automatic", fitting both Weibull and Lognormal curves. Alternatively accepts "Weibull", "Lognormal" or "Exponential" to force the type.
weighting	Boolean for whether to use weighting. Default=TRUE as it greatly improves curve fitting.
Weights	Name of Weights column. Default="Weights". Optional if weighting=FALSE. Recommended to use number at risk or remaining.
Weight_power	Power to raise the weights to. Useful in large trials to give added weight to later points where numbers may still be high. Default=1 (Use weights as specified)
startbeta	Starting value for the Weibull beta (shape) parameter to be used in the non-linear regression. Default=1 (exponential).
startsigma	Starting value for the Lognormal sigma (sd) parameter to be used in the non-linear regression. Default=1.

Value

Returns a 3-item list providing information needed to define a Curve object:

- "Item 1" The type of Curve object fitted.
- "Item 2" A list of fitted parameters for the curve type.
- "Item 3" A placeholder vector of NAs where the covariance-matrix parameters should be.
- "Item 4" A data frame containing the goodness of fit metrics for each curve type.

Author(s)

James Bell

Examples

```
recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
example_data_short <- simulate_trials(active_ecurve=Weibull(50,0.8),
control_ecurve=Weibull(50,0.8), rcurve=recruit, assess=10, iterations=1, seed=12345,
detailed_output=TRUE)

library(survival)

temp1 <- summary(survfit(Surv(example_data_short[, "Time"], 1-example_data_short[, "Censored"]))~ 1,
error="greenwood"))
out1 <- cbind(temp1$time, temp1$n.risk, temp1$surv, temp1$std.err)
out1 <- rbind(c(0, out1[1,2], 1, 0), out1)
colnames(out1) <- c("Time", "NAR", "Survival", "Std.Err")
x1 <- ceiling(max(out1[, "Time"]))
example_lifetable <- out1[findInterval(0:x1, out1[, "Time"]), ]
example_lifetable[, "Time"] <- 0:x1

fit_KM(KMcurve=example_lifetable, Survival="Survival", Time="Time", Weights="NAR", type="automatic")
```

fit_tte_data	<i>Fit Weibull, Log-Normal or Exponential survival curves to patient-level time-to-event data</i>
--------------	---

Description

This is a function to fit Weibull and log-normal curves to patient-level Survival data using maximum likelihood estimation.

By default it fits both, then picks the best fit based on the log-likelihood (and implicitly the AIC). Alternatively, just one shape may be fitted, by changing the 'type' argument to either "Weibull" or "Lognormal". This function is primarily used by event_prediction_data function, but also useful for general Survival function curve fitting.

One useful aspect of this is for fitting the 'inverse KM', where drop-outs are events, while events and 'time-outs' are censored. This allows for finding a suitable parameterisation for the censoring curve.

Where patient-level data is available, this function will typically perform substantially better than fit_KM, with lower variability of point estimates (and more accurate quantification of it).

Usage

```
fit_tte_data(
  data,
  Time = "Time",
  Event = "Event",
  censoringOne = FALSE,
  type = c("automatic", "Weibull", "Lognormal", "Exponential"),
  init = NULL
)
```

Arguments

data	The dataframe object containing the patient-level survival data
Time	The column name for the times. Default is "Time"
Event	The column name for the events column (i.e. the binary variable denoting events vs censorings). Default is "Event"
censoringOne	Specify whether censoring is denoted in the Event column by a one (TRUE) or zero (FALSE). Default=FALSE (censorings denoted by 0, events by 1)
type	Type of event curve to fit. Default is "Automatic", fitting both Weibull and Log-normal curves. Alternatively accepts "Weibull" or "Lognormal" to force the type.
init	Vector of starting values for parameter values; useful if survreg experiences convergence issues. Default=NULL (no values specified)

Value

Returns a 3-item list providing information needed to define a Curve object:

- "Item 1" The type of Curve object fitted.
- "Item 2" A list of fitted parameters for the curve type.
- "Item 3" A vector containing the covariance-matrix parameters for the curve type.
- "Item 4" A data frame containing the goodness of fit metrics for each curve type.

Author(s)

James Bell

Examples

```
recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
example_data <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit, assess=10,iterations=1,seed=12345,detailed_output=TRUE)

fit_tte_data(data=example_data,Time="Time",Event="Censored",censoringOne=TRUE,type="automatic")
```

fit_tte_data_prior	<i>Fit Weibull survival curves to patient-level time-to-event data by including patient-level weighted prior data</i>
--------------------	---

Description

This is a function to fit Weibull curves to patient-level Survival data integrating (weighted) prior data.

Where relevant prior data is available, this function can increase the precision of curve fitting, particularly in cases where there is low data maturity.

Usage

```
fit_tte_data_prior(
  data,
  Time = "Time",
  Event = "Event",
  censoringOne = FALSE,
  type = c("Weibull"),
  priordata,
  priorTime = Time,
  priorEvent = Event,
  priorcensoringOne = censoringOne,
  priorweight = 1,
  init = NULL
)
```

Arguments

<code>data</code>	The dataframe object containing the patient-level survival data
<code>Time</code>	The column name for the times. Default is "Time"
<code>Event</code>	The column name for the events column (i.e. the binary variable denoting events vs censorings). Default is "Event"
<code>censoringOne</code>	Specify whether censoring is denoted in the Event column by a one (TRUE) or zero (FALSE). Default=FALSE (censorings denoted by 0, events by 1)
<code>type</code>	Type of event curve to fit. Default is "Weibull", the only type currently accepted
<code>priordata</code>	The dataframe object containing a patient-level prior dataset. Where a patient-level prior data set is not readily available, the function <code>create_tte_prior</code> can be used to create one from summary parameters. Naming and format of time and event columns should match that of 'data'. Default=NULL (no prior data)
<code>priorTime</code>	The column name for the prior times. Default is the same value as <code>Time</code>
<code>priorEvent</code>	The column name for the prior events (i.e. the binary variable denoting events vs censorings). Default is the same value as <code>Event</code>
<code>priorcensoringOne</code>	Specify whether censoring is denoted in the prior Event column by a one (TRUE) or zero (FALSE). Default is the same as <code>censoringOne</code>
<code>priorweight</code>	The weight to assign the prior data; a non-negative number, where 0 corresponds to no weight, i.e. ignore prior data, and 1 corresponds to equal weighting of prior and observed data. The prior weight should typically not be greater than 1. Default=1 (prior patients weighted equivalently to observed data)
<code>init</code>	Vector of starting values for parameter values; useful if <code>survreg</code> experiences convergence issues. Default=NULL (no values specified)

Details

This function is primarily used by the `event_prediction_prior` function, but also useful for Weibull curve fitting across two patient-level data sets.

If a patient-level prior data set is not available, use the `create_tte_prior` function to create an artificial one from a distribution. Note that this may cause a small amount of over-precision; this may be minimised by deriving the prior data using a different distribution or specifying down-weighting.

Value

Returns a 3-item list providing information needed to define a Curve object:

- "Item 1" The type of Curve object fitted.
- "Item 2" A list of fitted parameters for the curve type.
- "Item 3" A vector containing the covariance-matrix parameters for the curve type.
- "Item 4" A data frame containing the goodness of fit metrics for each curve type.

Author(s)

James Bell

References

Bell J, unpublished work. D Fink, A Compendium of Conjugate Priors, 1997

Examples

```
recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
example_data <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit, assess=10,iterations=1,seed=12345,detailed_output=TRUE,Event="Event",
censoringOne=FALSE)
example_prior <- create_tte_prior(curve=Weibull(alpha=100,beta=0.8),duration=20,events=50)

fit_tte_data_prior(data=example_data,priordata=example_prior,priorweight=0.5)
```

frontierpower	<i>Calculate Frontier power from number of events</i>
---------------	---

Description

Calculate Frontier power from number of events

Usage

```
frontierpower(
  events,
  HR,
  Eratio,
  Rratio = 1,
  startpower = 0.5,
  alpha1 = 0.025,
  HRbound = 1,
  iter = 10
)
```

Arguments

events	Number of events.
HR	Hazard Ratio. Values below 1 indicate a benefit to the active arm vs control.
Eratio	Event ratio.
Rratio	Randomisation ratio. Default=1.
startpower	Initial estimate of power. Default=0.5.
alpha1	1-sided alpha. Default=0.025.
HRbound	HRvalue to test against. Default=1 (superiority).
iter	Number of iterations to perform. Default=10.

Value

Power as a decimal

Author(s)

James Bell

References

Bell J, Power Calculations for Time-to-Event Trials Using Predicted Event Proportions, 2019, paper under review.

Examples

```
frontierpower(events=300,HR=0.7,Eratio=1.2,Rratio=1.5,alpha1=0.025)
```

`getAssessCDFfunction` *Method for returning the CDF function for a RCurve object*

Description

This retrieves the CDF function of the specified RCurve object as a string, writing in the assessment time parameter

Usage

```
getAssessCDFfunction(theObject, ...)
```

Arguments

<code>theObject</code>	The name of the RCurve Object
<code>...</code>	Pass-through arguments

Examples

```
getAssessCDFfunction(LinearR(12,100,100))
```

getAssessCDFfunction, RCurve-method

Method for returning the CDF function for a RCurve object

Description

This retrieves the CDF function of the specified RCurve object as a string, writing in the assessment time parameter

Usage

```
## S4 method for signature 'RCurve'
getAssessCDFfunction(theObject, q = "q")
```

Arguments

theObject	The name of the RCurve Object
q	The parameter name to use in the CDF function. Default=q

Examples

```
getAssessCDFfunction(LinearR(12,100,100))
```

getCDFfunction

Method for returning the CDF function for a Curve object

Description

This retrieves the full CDF function of the Curve object as a string

Usage

```
getCDFfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
getCDFfunction(Weibull(100,1))
```

getCDFfunction, Curve-method

Method for returning the CDF function for a Curve object

Description

This retrieves the CDF function of the specified Curve object as a string

Usage

```
## S4 method for signature 'Curve'
getCDFfunction(theObject, q = "q")
```

Arguments

theObject	The name of the Curve Object
q	The parameter name to use in the CDF function. Default=q

Examples

```
getCDFfunction(Weibull(100,1))
```

getInvfunction

Method for returning the inverse-CDF function for a Curve object

Description

This retrieves the full inverse-CDF function of the Curve object as a string

Usage

```
getInvfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
getInvfunction(Weibull(100,1))
```

getInvfunction, Curve-method

Method for returning the inverse-CDF function for a Curve object

Description

This retrieves the inverse CDF function of the specified Curve object as a string

Usage

```
## S4 method for signature 'Curve'
getInvfunction(theObject, p = "p")
```

Arguments

theObject	The name of the Curve Object
p	The probability parameter name to use in the CDF function. Default=p

Examples

```
getCdfFunction(Weibull(100,1))
```

getLength

Method for returning the recruitment length from a RCurve

Description

This returns the RCurve recruitment length

Usage

```
getLength(theObject)
```

Arguments

theObject	The name of the RCurve Object
-----------	-------------------------------

Examples

```
getLength(LinearR(12,100,100))
```

getLength,RCurve-method

Method for returning the recruitment length from a RCurve

Description

This returns the RCurve recruitment length

Usage

```
## S4 method for signature 'RCurve'
getLength(theObject)
```

Arguments

theObject The name of the RCurve Object

Examples

```
getLength(LinearR(12,100,100))
```

getMaxF

Method for returning maximum duration of patient follow-up from a RCurve

Description

This returns the RCurve maximum patient follow-up

Usage

```
getMaxF(theObject)
```

Arguments

theObject The name of the RCurve Object

Examples

```
getMaxF(LinearR(12,100,100))
```

getMaxF, RCurve-method	<i>Method for returning maximum duration of patient follow-up from a RCurve</i>
------------------------	---

Description

This returns the RCurve maximum patient follow-up

Usage

```
## S4 method for signature 'RCurve'  
getMaxF(theObject)
```

Arguments

theObject	The name of the RCurve Object
-----------	-------------------------------

Examples

```
getMaxF(LinearR(12,100,100))
```

getN	<i>Method for returning the total patient number from a RCurve</i>
------	--

Description

This returns the RCurve total patient number

Usage

```
getN(theObject)
```

Arguments

theObject	The name of the RCurve Object
-----------	-------------------------------

Examples

```
getN(LinearR(12,100,100))
```

getN,RCurve-method	<i>Method for returning the total patient number from a RCurve</i>
--------------------	--

Description

This returns the RCurve total patient number

Usage

```
## S4 method for signature 'RCurve'  
getN(theObject)
```

Arguments

theObject	The name of the RCurve Object
-----------	-------------------------------

Examples

```
getN(LinearR(12,100,100))
```

getNactive	<i>Method for returning the active arm patient number from a RCurve</i>
------------	---

Description

This returns the RCurve active arm patient number

Usage

```
getNactive(theObject)
```

Arguments

theObject	The name of the RCurve Object
-----------	-------------------------------

Examples

```
getNactive(LinearR(12,100,100))
```

getNactive,RCurve-method

Method for returning the active arm patient number from a RCurve

Description

This returns the RCurve active arm patient number

Usage

```
## S4 method for signature 'RCurve'
getNactive(theObject)
```

Arguments

theObject The name of the RCurve Object

Examples

```
getNactive(LinearR(12,100,100))
```

getNames

Method for returning all parameter names from a Curve object

Description

This retrieves all parameter names from a Curve object

Usage

```
getNames(theObject)
```

Arguments

theObject The name of the Curve Object

Examples

```
getNames(Weibull(100,1))
```

getNames, Curve-method *Method for returning all parameter names from a Curve object*

Description

This retrieves all parameter names from a Curve object

Usage

```
## S4 method for signature 'Curve'
getNames(theObject)
```

Arguments

theObject The name of the Curve Object

Examples

```
getNames(Weibull(100,1))
```

getNcontrol *Method for returning the control arm patient number from a RCurve*

Description

This returns the RCurve control arm patient number

Usage

```
getNcontrol(theObject)
```

Arguments

theObject The name of the RCurve Object

Examples

```
getNcontrol(LinearR(12,100,100))
```

getNcontrol, RCurve-method

Method for returning the control arm patient number from a RCurve

Description

This returns the RCurve control arm patient number

Usage

```
## S4 method for signature 'RCurve'  
getNcontrol(theObject)
```

Arguments

theObject The name of the RCurve Object

Examples

```
getNcontrol(LinearR(12,100,100))
```

getParam

Method for returning a single parameter from a Curve object

Description

This retrieves a single parameter from a Curve object

Usage

```
getParam(theObject, ...)
```

Arguments

theObject The name of the Curve Object
... Pass-through arguments

Examples

```
getParam(Weibull(100,1),1)
```

getParam, Curve-method *Method for returning a single parameter from a Curve object*

Description

This retrieves a single parameter from a Curve object

Usage

```
## S4 method for signature 'Curve'
getParam(theObject, param)
```

Arguments

theObject	The name of the Curve Object
param	The number of the parameter that is required

Examples

```
getParam(Weibull(100,1),1)
```

getParams *Method for returning all parameters from a Curve object as a list*

Description

This retrieves all parameters from a Curve object as a list

Usage

```
getParams(theObject)
```

Arguments

theObject	The name of the Curve Object
-----------	------------------------------

Examples

```
getParams(Weibull(100,1))
```

getParams, Curve-method

Method for returning all parameters from a Curve object as a list

Description

This retrieves all parameters from a Curve object as a list

Usage

```
## S4 method for signature 'Curve'
getParams(theObject)
```

Arguments

theObject The name of the Curve Object

Examples

```
getParams(Weibull(100,1))
```

getParamsV

Method for returning all parameters from a Curve object as a vector

Description

This retrieves all parameters from a Curve object as a vector

Usage

```
getParamsV(theObject)
```

Arguments

theObject The name of the Curve Object

Examples

```
getParamsV(Weibull(100,1))
```

`getParamsV, Curve-method`*Method for returning all parameters from a Curve object as a vector*

Description

This retrieves all parameters from a Curve object as a vector

Usage

```
## S4 method for signature 'Curve'
getParamsV(theObject)
```

Arguments

`theObject` The name of the Curve Object

Examples

```
getParamsV(Weibull(100,1))
```

`getPatients`*Method for calculating expected number of recruited patients at a given time from an RCurve*

Description

This calculates the expected number of recruited patients at a given time based upon an RCurve

Usage

```
getPatients(theObject, ...)
```

Arguments

`theObject` The name of the RCurve Object
`...` Pass-through arguments

Examples

```
getPatients(LinearR(12,100,100),7)
```

getPatients,RCurve-method

Method for calculating expected number of recruited patients at a given time from an RCurve

Description

This calculates the expected number of recruited patients at a given time based upon an RCurve

Usage

```
## S4 method for signature 'RCurve'
getPatients(theObject, x)
```

Arguments

theObject	The name of the RCurve Object
x	time at which to calculate expected patients recruited"

Examples

```
getPatients(LinearR(12,100,100),7)
```

getPDFfunction

Method for returning the PDF function for a Curve object

Description

This retrieves the PDF function of the Curve object as a string

Usage

```
getPDFfunction(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
getPDFfunction(Weibull(100,1))
```

`getPDFfunction, Curve-method`*Method for returning the PDF function for a Curve object*

Description

This retrieves the PDF function of the Curve object as a string

Usage

```
## S4 method for signature 'Curve'
getPDFfunction(theObject, x = "x")
```

Arguments

<code>theObject</code>	The name of the Curve Object
<code>x</code>	The parameter name to use in the PDF function. Default=x

Examples

```
getPDFfunction(Weibull(100,1))
```

`getRatio`*Method for returning the recruitment ratio from a RCurve*

Description

This returns the RCurve recruitment ratio

Usage

```
getRatio(theObject)
```

Arguments

<code>theObject</code>	The name of the RCurve Object
------------------------	-------------------------------

Examples

```
getRatio(LinearR(12,100,100))
```

getRatio, RCurve-method

Method for returning the recruitment ratio from a RCurve

Description

This returns the RCurve recruitment ratio

Usage

```
## S4 method for signature 'RCurve'
getRatio(theObject)
```

Arguments

theObject The name of the RCurve Object

Examples

```
getRatio(LinearR(12,100,100))
```

getRFfunction

Method for returning the RF function for a Curve object

Description

This retrieves the full RF function of the Curve object as a string

Usage

```
getRFfunction(theObject, ...)
```

Arguments

theObject The name of the Curve Object
 ... Pass-through arguments

Examples

```
getRFfunction(Weibull(100,1))
```

`getRFfunction, Curve-method`*Method for returning the RF function for a Curve object*

Description

This retrieves the full RF function of the Curve object as a string

Usage

```
## S4 method for signature 'Curve'  
getRFfunction(theObject, n = "n")
```

Arguments

<code>theObject</code>	The name of the Curve Object
<code>n</code>	The parameter name for the number of random draws to make. Default=n

Examples

```
getRFfunction(Weibull(100,1))
```

`getType`*Method for returning the Curve type*

Description

This returns the Curve object type

Usage

```
getType(theObject)
```

Arguments

<code>theObject</code>	The name of the Curve Object
------------------------	------------------------------

Examples

```
getType(Weibull(100,1))
```

getType, Curve-method	<i>Method for returning the Curve type</i>
-----------------------	--

Description

This returns the Curve object type

Usage

```
## S4 method for signature 'Curve'
getType(theObject)
```

Arguments

theObject	The name of the Curve Object
-----------	------------------------------

Examples

```
getType(Weibull(100,1))
```

GGamma	<i>Generalised Gamma Curve constructor function</i>
--------	---

Description

This creates a Curve object for a Generalised Gamma distribution.
 Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.
 See Details for information on parameterisation.

Usage

```
GGamma(theta, eta, rho)
```

Arguments

theta	Scale parameter for Generalised Gamma distribution.
eta	Shape parameter for Generalised Gamma distribution.
rho	Family parameter for Generalised Gamma distribution.

Details

The Generalised Gamma distribution has parameterisation:
 $f(x) = (\rho x^{(\rho \eta)-1} e^{-(x/\theta)^\rho} \theta^{(-\rho \eta)}) / \Gamma(\eta)$
 $F(x) = \text{LPGamma}(\eta, (x/\theta)^\rho) / \Gamma(\eta)$
 where Γ is the gamma function, and LPGamma is the lower partial gamma function.
 As of v1.4.0, all values of η are now fully supported.

Author(s)

Jasmin Ruehl

References

Tadikamalla PR, Random Sampling from the Generalized Gamma Distribution. Computing, 1979, 23(2), 199-203.

Examples

```
GGamma(theta=20,eta=2,rho=0.7)
```

Gompertz

Gompertz Curve constructor function

Description

This creates a Curve object for a Gompertz distribution.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

See Details for information on parameterisation.

Usage

```
Gompertz(theta, eta)
```

Arguments

theta	Scale parameter for Log-logistic distribution.
eta	Shape parameter for Log-logistic distribution.

Details

The Gompertz distribution has parameterisation:

$$f(x) = \theta \eta e^{(\eta + \theta x - \eta e^{\theta x})}$$

$$F(x) = 1 - \exp(\eta - \eta e^{\theta x})$$
Author(s)

Jasmin Ruehl

Examples

```
Gompertz(theta=0.02,eta=2)
```

InstantR

InstantR RCurve constructor function

Description

This creates a RCurve object for an instant recruitment distribution. RCurve objects contain all necessary information to describe a recruitment distribution. They are a particular type of Curve object containing additional recruitment-related information, including patient numbers and the randomisation ratio.

Usage

```
InstantR(Nactive, Ncontrol)
```

Arguments

Nactive	Number of patients recruited in the active arm.
Ncontrol	Number of patients recruited in the control arm.

Details

This RCurve is used when either all patients enter at the same time, or a fixed-length follow-up design is used. Note that a PDF function is not provided for this RCurve type, but is not required for standard calculations.

Author(s)

James Bell

Examples

```
InstantR(Nactive=100,Ncontrol=100)
```

LinearR

LinearR RCurve constructor function

Description

This creates a RCurve object for a linear recruitment distribution. RCurve objects contain all necessary information to describe a recruitment distribution. They are a particular type of Curve object containing additional recruitment-related information, including patient numbers and the randomisation ratio.

Usage

```
LinearR(rlength, Nactive, Ncontrol)
```

Arguments

<code>rlength</code>	Length of recruitment.
<code>Nactive</code>	Number of patients recruited in the active arm.
<code>Ncontrol</code>	Number of patients recruited in the control arm.

Details

This RCurve is used when it is expected that patients enter a trial at a constant rate until the required number is achieved.

Author(s)

James Bell

Examples

```
LinearR(rlength=12,Nactive=100,Ncontrol=100)
```

LogLogistic	<i>Log-logistic Curve constructor function</i>
-------------	--

Description

This creates a Curve object for a Log-logistic distribution.
 Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.
 See Details for information on parameterisation.

Usage

```
LogLogistic(theta, eta)
```

Arguments

<code>theta</code>	Scale parameter for Log-logistic distribution.
<code>eta</code>	Shape parameter for Log-logistic distribution.

Details

The log-logistic distribution has parameterisation:
 $f(x) = \eta (\theta^\eta) x^{\eta-1} (\theta^\eta + x^\eta)^{-2}$
 $F(x) = (x^\eta) / (\theta^\eta + x^\eta)$

Author(s)

Jasmin Ruehl

Examples

```
LogLogistic(theta=20,eta=2)
```

Lognormal

Log-normal Curve constructor function

Description

This creates a Curve object for a Log-normal distribution.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

Parameterisation follows that used by plnorm etc. See Details for more information on parameterisation.

Usage

```
Lognormal(mu, sigma = 1)
```

Arguments

mu Mean (on log scale) parameter for Log-normal distribution.

sigma Standard deviation (on log scale) parameter for Log-normal distribution.

Details

The log normal distribution has parameterisation:

$$f(x) = 1/(\sqrt{2\pi} \sigma x) e^{-((\log x - \mu)^2 / (2 \sigma^2))}$$

$$F(x) = 0.5(1 + \operatorname{erf}((\log(x)-\mu)/(\sigma \sqrt{2})))$$

where erf is the error function.

Author(s)

James Bell

Examples

```
Lognormal(mu=5,sigma=1.2)
```

```
Lognormal(6)
```

MixExp

Mixture Exponential Curve constructor function

Description

This creates a Curve object for a Mixture Exponential distribution, commonly used for modelling distributions with subpopulations.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

Parameterisation follows that used by pexp etc. See Details for more information on parameterisation.

Usage

```
MixExp(props, lambdas)
```

Arguments

props	Vector of length x for the probabilities of the subpopulations. Must sum to 1.
lambdas	Vector of length x for the rate parameters for the corresponding subpopulations define by props.

Details

The mixture distribution with rates lambda1 to lambda2 etc and prevalence p1 and p2 etc has parameterisation:

$$f(x) = p1 \lambda_1 e^{(-\lambda_1 x)} + p2 \lambda_2 e^{(-\lambda_2 x)} + \dots$$

$$F(x) = p1 (1 - e^{(-\lambda_1 x)}) + p2 (1 - e^{(-\lambda_2 x)}) + \dots$$

Author(s)

James Bell

Examples

```
MixExp(props=c(0.8,0.2),lambdas=c(0.01,0.1))
```

MixWei

Mixture Weibull Curve constructor function

Description

This creates a Curve object for a Mixture Weibull distribution. Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it. Parameterisation follows that used by pweibull etc. See Details for more information on parameterisation.

Usage

```
MixWei(props, alphas, betas = rep(1, length(props)))
```

Arguments

props	Vector of length x for the probabilities of the two subpopulations. Must sum to 1.
alphas	Vector of length x for the scale parameters for the corresponding subpopulations define by props.
betas	Vector of length x for the shape parameters for the corresponding subpopulations define by props. Default is rep(1,length(props)), i.e. all exponential distributions.

Details

The mixture distribution with scales alpha1 and alpha2 etc, shapes beta1 and beta2 etc, and prevalences p1 and p2 etc has parameterisation:

$$f(x) = p1 (beta1/alpha1) (x/alpha1)^{(beta1-1)} \exp(- (x/alpha1)^{beta1}) + p2 (beta2/alpha2) (x/alpha2)^{(beta2-1)} \exp(- (x/alpha2)^{beta2}) + \dots$$

$$F(x) = p1 (1 - \exp(- (x/alpha1)^{beta1})) + p2 (1 - \exp(- (x/alpha2)^{beta2})) + \dots$$

Author(s)

James Bell

Examples

```
MixWei(props=c(0.8,0.2),alphas=c(100,10),betas=c(1.1,0.9))
```

nph_traj	<i>Calculate analytic time-to-event trial properties under non-proportional hazards and complex assumptions</i>
----------	---

Description

This function calculates the expected parameters/outputs/properties for a two-arm Time-To-Event trial under complex assumptions. It is designed to work with non-proportional hazards and ought to be able to accommodate any distributional assumptions for events, censoring and recruitment, so long as they are correctly detailed in Curve or RCurve objects.

The function performs power calculations and hence can be used for sample size planning. By creating trajectories of properties over time, the function also assists with identifying the optimum assessment time.

The function uses numerical integration across event, censoring and recruitment functions to estimate observed and expected event numbers. From these, it estimates an expected HR, with the same interpretation as that found using Cox regression, using the Pike method. The estimated event numbers and HR can be used to calculate power by one of several methods, including the Schoenfeld and Frontier methods. A separate, direct, power calculation can also be performed using the log-rank test formula and its Z-distribution.

To assist sample size finding, the function will also optionally estimate the required sample size to reach a given power keeping all variables other than recruitment.

Expected RMST and landmark analysis properties may also be calculated. This also uses numerical integration techniques. Power is also then estimated for such analyses.

Usage

```
nph_traj(
  active_ecurve,
  control_ecurve,
  active_dcurve = Blank(),
  control_dcurve = Blank(),
  rcurve,
  max_assessment = 100,
  landmark = NULL,
  RMST = NULL,
  HRbound = 1,
  alpha1 = 0.025,
  required_power = NULL,
  detailed_output = FALSE
)
```

Arguments

`active_ecurve` Event distribution for the active arm, specified as a Curve object
`control_ecurve` Event distribution for the control arm, specified as a Curve object

active_dcurve	Dropout/censoring distribution for the active arm, specified as a Curve object. By default, a Blank() object, i.e. no dropout.
control_dcurve	Dropout/censoring distribution for the control arm, specified as a Curve object. By default, a Blank() object, i.e. no dropout.
rcurve	Recruitment distribution, specified as an RCurve object
max_assessment	Maximum assessment time to calculate properties up to
landmark	(Optional) Time in months of landmark analysis, if required. Otherwise NULL (Not calculated; default).
RMST	(Optional) Restriction time for RMST analysis in months, if required. Otherwise NULL (Not calculated; default).
HRbound	(Optional) Specify HR value to test landmark significance against. Default is 1 (superiority testing). Values above 1 would be non-inferiority.
alpha1	One-sided alpha required, as a decimal. 0.025 by default. Requires $0 < \alpha_1 \leq 0.5$.
required_power	(Optional) Power required for estimated sample sizes. Otherwise NULL (not calculated; default).
detailed_output	Boolean to require a more detailed output table, including Peto LogHR, expectations of various quantities and alternative power calculations. Default = FALSE (detailed outputs omitted).

Value

Returns a table with one row per assessment time. Table contains both all input parameters as well as the following expected quantities:

- "Time" Time at which the assessment is made
- "Patients" Number of patients expected to be recruited to date
- "Events_Active" Expected number of observed events in active arm
- "Events_Control" Expected number of observed events in control arm
- "Events_Total" Expected number of events across both arms
- "HR" Expected Hazard Ratio (using the Pike method)
- "LogHR" Log of the expected Hazard Ratio
- "LogHR_SE" SE of the log of the expected Hazard Ratio
- "Schoenfeld_Power" Estimated power based on Schoenfeld formula
- "Frontier_Power" Estimated power based on Frontier method, using estimated event ratio at 0.5 power

In addition, if the detailed_output argument is set to TRUE, the following additional columns are provided:

- "E_Events_Active" Expected number of expected events in active arm
- "E_Events_Control" Expected number of expected events in control arm
- "HR_CI_Upper" Estimated Upper Bound of the CI for the Hazard Ratio

- "HR_CI_Lower" Estimated Lower Bound of the CI for the Hazard Ratio
- "Peto_LogHR" Expected Log Hazard Ratio using the Peto method
- "Expected_Z" Estimated Z-score based on expected quantities for O, E and V, and log-rank test formula
- "Expected_P" Estimated p-value based on estimated Z-score
- "Log_Rank_Stat" Expected log-rank statistic
- "Variance" Expected variance of LR-statistic by integration of V_function
- "V_Pike_Peto" Expected variance based upon Pike and Peto approximations
- "Event_Ratio" Expected ratio of events between arms; active divided by control
- "Event_Prop_Power" Estimated power based on event proportion method, using event ratio rather than randomisation ratio
- "Z_Power" Estimated power based on expected value of Z

If RMST calculations are requested, the following columns are included:

- "RMST_Restrict" Specified RMST restriction time
- "RMST_Active" Expected RMST for active arm
- "RMST_Control" Expected RMST for control arm
- "RMST_Delta" Absolute difference in expected RMSTs between arms (active minus control)
- "RMST_SE" Estimated SE of the RMST delta
- "RMST_Z" Estimated RMST Z score
- "RMST_Failure" Estimated probability of RMST difference being uncomputable for the specified restriction time
- "RMST_Power" Estimated RMST Power

If landmark calculations are requested, the following columns are included:

- "LM_Time" Time of landmark analysis
- "LM_Active" Expected Kaplan Meier estimate of active arm at landmark time
- "LM_Control" Expected Kaplan Meier estimate of control arm at landmark time
- "LM_Delta" Expected absolute difference in Kaplan Meiers estimates at landmark time (active-control)
- "LM_A_SE" Estimated Greenwood SE for active arm at landmark time
- "LM_C_SE" Estimated Greenwood SE for control arm at landmark time
- "LM_D_SE" Estimated Greenwood SE for delta at landmark time
- "LM_Z" Estimated landmark analysis Z-score based on Greenwood SE
- "LM_Power" Estimated landmark analysis power based on Greenwood SE

If a required power is requested, the following column is included:

- "Estimated_SS" Estimated sample size required, keeping constant all parameters other than rate of recruitment and total sample size

Author(s)

James Bell

References

Bell J, Accurate Sample Size Calculations in Trials with Non-Proportional Hazards, 2018, presentation at PSI Conference. https://www.psiweb.org/docs/default-source/default-document-library/james-bell-slides.pdf?sfvrsn=3324dedb_0 Bell J, Power Calculations for Time-to-Event Trials Using Predicted Event Proportions, 2019, unpublished. Ruehl J, Sample Size Calculation in Time-To-Event Trials with Non-Proportional Hazards Using GESTATE, 2018, BSc thesis at University of Ulm. Pike MC, Contribution to discussion in Asymptotically efficient rank invariant test procedures by Peto R and Peto J, Journal of the Royal Statistical Society Series A, 135(2), 201-203.

Examples

```
nph_traj(max_assessment=100,rcurve=LinearR(12,100,100),control_ecurve=Weibull(100,1),
active_ecurve=Weibull(250,0.8))
```

PieceExponential

Piecewise Exponential Curve constructor function

Description

This creates a Curve object for a Piecewise Exponential distribution.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

Parameterisation follows that used by pexp etc. This function requires a vector of start times (beginning with 0) and a corresponding vector of rates. See Details for more information on parameterisation.

Usage

```
PieceExponential(start, lambda)
```

Arguments

start	Vector of start times for each period. First element must be 0. Must be same length as lambda vector.
lambda	Vector of rate parameters from the corresponding respective time defined in start vector until the start of the next period. Must be same length as start vector.

Details

The piecewise exponential distribution with rates λ_1 to λ_n and start times t_1 to t_n has parameterisation:

Product($x=1:\text{length}(\lambda)$) of $(e^{-\lambda[x].t[x]})$ where $t[x] = \min(\text{start}[x+1], \max(0, t - \text{start}[x]))$. $\text{start}[x+1]$ is defined as Inf if otherwise undefined.

Author(s)

James Bell

Examples

```
PieceExponential(start=c(0,6,24),lambda=c(0.05,0.01,0.001))
```

 PieceR

PieceR RCurve constructor function

Description

This creates a RCurve object for a piecewise-linear recruitment distribution. RCurve objects contain all necessary information to describe a recruitment distribution. They are a particular type of Curve object containing additional recruitment-related information, including patient numbers and the randomisation ratio.

Usage

```
PieceR(recruitment, ratio)
```

Arguments

recruitment	2-column matrix with recruitment parameters. First column gives the lengths of each period of recruitment. Second column gives the corresponding rates of recruitment for each period.
ratio	Randomisation ratio; active arm divided by control arm.

Details

This RCurve is used when it is expected that patients enter a trial at a rate that varies over time.

Author(s)

James Bell

Examples

```
rmatrix <- matrix(c(rep(4,3),5,10,15),ncol=2)
rmatrix
PieceR(rmatrix,1)
```

PieceRMaxF

PieceR RCurve constructor function

Description

This creates a RCurve object for a piecewise-linear recruitment distribution with a fixed (maximum) per-patient follow-up time.

RCurve objects contain all necessary information to describe a recruitment distribution. They are a particular type of Curve object containing additional recruitment-related information, including patient numbers and the randomisation ratio.

Usage

```
PieceRMaxF(recruitment, ratio, maxF)
```

Arguments

recruitment	2-column matrix with recruitment parameters. First column gives the lengths of each period of recruitment. Second column gives the corresponding rates of recruitment for each period.
ratio	Randomisation ratio; active arm divided by control arm.
maxF	Fixed follow-up time per patient, i.e. maximum time a patient will be at risk independent of length of study.

Details

This RCurve is used when it is expected that patients enter a trial at a rate that varies over time and there is a fixed maximum follow-up time per patient.

Author(s)

James Bell

Examples

```
rmatrix <- matrix(c(rep(4,3),5,10,15),ncol=2)
rmatrix
PieceRMaxF(recruitment=rmatrix,ratio=1,maxF=12)
```

plotCDF	<i>Method for plotting the CDF of a Curve object</i>
---------	--

Description

This plots a Curve CDF

Usage

```
plotCDF(theObject, ...)
```

Arguments

theObject	The name of the RCurve Object
...	Pass-through arguments

Examples

```
plotCDF(Weibull(100,1))
plotCDF(Weibull(100,1),xlab="Test x label",maxT=60)
plotCDF(Weibull(80,0.8),overlay=TRUE,col=2,lty=2)
```

plotCDF, Curve-method	<i>Method for plotting the CDF of a Curve object</i>
-----------------------	--

Description

This plots a Curve CDF

Usage

```
## S4 method for signature 'Curve'
plotCDF(
  theObject,
  overlay = FALSE,
  maxT = 100,
  maxCDF = 1,
  increment = 0.1,
  xlab = "Time",
  ylab = "CDF",
  main = "CDF plot",
  type = "l",
  ...
)
```

Arguments

theObject	The name of the RCurve Object
overlay	Boolean whether to overlay on existing plot (vs start a new one). Default=FALSE
maxT	Maximum time to plot up to. Default=100
maxCDF	Maximum CDF to plot up to. Default=1
increment	Plotting time increment. Default=0.1
xlab	X-axis label. Default="Time"
ylab	Y-axis label. Default="CDF"
main	title of plot. Default="CDF plot"
type	type of plot (see standard graphical parameters). Default="l" (lines).
...	Standard graphical parameter arguments to be passed on to 'plot'/'lines', e.g. to change appearance of plot.

Examples

```
plotCDF(Weibull(100,1))
plotCDF(Weibull(100,1),xlab="Test x label",maxT=60)
plotCDF(Weibull(80,0.8),overlay=TRUE,col=2,lty=2)
```

plotRecruitment

Method for plotting the Recruitment Function of a RCurve object

Description

This plots an RCurve Recruitment Function

Usage

```
plotRecruitment(theObject, ...)
```

Arguments

theObject	The name of the RCurve Object
...	Pass-through arguments

Examples

```
plotRecruitment(LinearR(12,100,100))
plotRecruitment(LinearR(12,100,100),xlab="Test x label",maxT=60)
plotRecruitment(LinearR(20,90,90),overlay=TRUE,col=2,lty=2)
```

plotRecruitment,RCurve-method

Method for plotting the Recruitment Function of a RCurve object

Description

This plots an RCurve Recruitment Function

Usage

```
## S4 method for signature 'RCurve'
plotRecruitment(
  theObject,
  overlay = FALSE,
  maxT = 100,
  increment = 0.1,
  xlab = "Time",
  ylab = "Patients",
  main = "Recruitment plot",
  type = "l",
  ...
)
```

Arguments

theObject	The name of the RCurve Object
overlay	Boolean whether to overlay on existing one (vs start a new one). Default=FALSE
maxT	Maximum time to plot up to. Default=100
increment	Plotting time increment. Default=0.1
xlab	X-axis label. Default="Time"
ylab	Y-axis label. Default="Patients"
main	title of plot. Default="Recruitment plot"
type	type of plot (see standard graphical parameters). Default="l" (lines).
...	Standard graphical parameter arguments to be passed on to 'plot'/'lines', e.g. to change appearance of plot.

Examples

```
plotRecruitment(LinearR(12,100,100))
plotRecruitment(LinearR(12,100,100),xlab="Test x label",maxT=60)
plotRecruitment(LinearR(20,90,90),overlay=TRUE,col=2,lty=2)
```

plotSF	<i>Method for plotting the Survival Function of a Curve object</i>
--------	--

Description

This plots a Curve Survival Function

Usage

```
plotSF(theObject, ...)
```

Arguments

theObject	The name of the RCurve Object
...	Pass-through arguments

Examples

```
plotSF(Weibull(100,1))
plotSF(Weibull(100,1),xlab="Test x label",maxT=60)
plotSF(Weibull(80,0.8),overlay=TRUE,col=2,lty=2)
```

plotSF, Curve-method	<i>Method for plotting the Survival Function of a Curve object</i>
----------------------	--

Description

This plots a Curve Survival Function

Usage

```
## S4 method for signature 'Curve'
plotSF(
  theObject,
  overlay = FALSE,
  maxT = 100,
  minSF = 0,
  increment = 0.1,
  xlab = "Time",
  ylab = "S(t)",
  main = "Kaplan Meier plot",
  type = "l",
  ...
)
```

Arguments

theObject	The name of the RCurve Object
overlay	Boolean whether to overlay on existing one (vs start a new one). Default=FALSE
maxT	Maximum time to plot up to. Default=100
minSF	Minimum SF to plot down to. Default=0
increment	Plotting time increment. Default=0.1
xlab	X-axis label. Default="Time"
ylab	Y-axis label. Default="S(t)"
main	title of plot. Default="Kaplan Meier plot"
type	type of plot (see standard graphical parameters). Default="l" (lines).
...	Standard graphical parameter arguments to be passed on to 'plot'/'lines', e.g. to change appearance of plot.

Examples

```
plotSF(Weibull(100,1))
plotSF(Weibull(100,1),xlab="Test x label",maxT=60)
plotSF(Weibull(80,0.8),overlay=TRUE,col=2,lty=2)
```

plot_ep	<i>Plot event prediction output</i>
---------	-------------------------------------

Description

This function plots the output from event_prediction().

By default, produces a plot of predicted events over time, with confidence intervals if available.

Alternatively, produces a plot with the fitting CI over time with same percentage as prediction interval.

By default, both conditional and unconditional trajectories are plotted (if conditional event prediction is available).

Options are available to customise inclusion.

Usage

```
plot_ep(
  data,
  trajectory = c("both", "conditional", "unconditional"),
  which_PI = c("both", "conditional", "unconditional", "none"),
  prediction_fitting = c("prediction", "fitting"),
  observed = NULL,
  target = NULL,
  max_time = NULL,
  max_E = NULL,
```

```

    legend_position = c("top_left", "bottom_right"),
    no_legend = FALSE,
    ...
)

```

Arguments

<code>data</code>	Full output list from <code>event_prediction()</code> .
<code>trajectory</code>	String, choice of "both", "conditional", "unconditional", for which trajectory to plot. (Default="both")
<code>which_PI</code>	String, choice of "both", "conditional", "unconditional", "none", for which PIs to plot. (Default="both")
<code>prediction_fitting</code>	String, choice of "prediction" or "fitting" (Default = "prediction"). Determines the nature of the intervals; PIs are relevant for prediction of the observation of future trajectories (sample level), whereas fitting CIs concern the interval for the mean trajectory itself (population level).
<code>observed</code>	Optional trajectory of observed event numbers. If vector specified, will plot values at integer times starting from 1. If 2-column matrix specified, will take x-values from column 1 and y-values from column 2. (Default=NULL; not plotted).
<code>target</code>	Optional target number of events to plot. (Default=NULL; not plotted)
<code>max_time</code>	Optional maximum time to plot up to if you do not want to plot full trajectory. (Default=NULL; maximum time determined by input data)
<code>max_E</code>	Optional maximum number of events to plot up to. (Default=NULL; maximum event number is the number of patients)
<code>legend_position</code>	String with any of "top_left", or "bottom_right", corresponding to legend position in power plot. (Default="top_left").
<code>no_legend</code>	Boolean to turn off legend. Default is FALSE; legend shown.
<code>...</code>	Additional graphical parameters.

Value

Returns NULL

Author(s)

James Bell

Examples

```

recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
trial_long <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit,fix_events=200, iterations=1,seed=12345,detailed_output=TRUE)
trial_short <- set_assess_time(data=trial_long,time=10,detailed_output = FALSE)

```

```

maxtime <- max(ceiling(trial_long[, "Assess"]))
events <- rep(NA, maxtime)
for (i in 1:maxtime){events[i] <- sum(1-set_assess_time(trial_long,i)[, "Censored"])}

predictions <- event_prediction(data=trial_short, Event="Censored", censoringOne=TRUE,
type="Weibull", rcurve=recruit, max_time=60, cond_Events=49, cond_NatRisk=451,
cond_Time=10, units="Months")

plot_ep(predictions, trajectory="conditional", which_PI="conditional", max_time=40, observed=events,
target=200, max_E=200)

plot_ep(predictions, trajectory="unconditional", which_PI="unconditional", max_time=40,
observed=events, target=200, max_E=200)

plot_ep(predictions, trajectory="conditional", which_PI="none", observed=events[1:10], max_time=20,
max_E=150)

```

plot_km_fit

Kaplan Meier Plot of Curve-Fit

Description

This function creates a Kaplan Meier plot with the fitted curve from the output of `event_prediction()`, `fit_tte_data()` or `fit_KM()`.

Where available, it will include fitting confidence intervals based upon the variance derived by the delta method.

Options are available to customise inclusion.

Usage

```

plot_km_fit(
  fit,
  data,
  Time = "Time",
  Event = "Event",
  censoringOne = FALSE,
  CI = 0.95,
  colour_CI = TRUE,
  maxT = NULL,
  xlim = NULL,
  ylim = c(0, 1),
  main = "Kaplan Meier Curve Fit Plot",
  fit_col = 2,
  km_col = 1,
  area_col = "skyblue",
  CI_col = 4,

```

```

    CI_lty = 2,
    no_legend = FALSE,
    legend_position = c("bottom_left", "top_right"),
    overlay = FALSE,
    ...
)

```

Arguments

<code>fit</code>	Full output list from <code>event_prediction()</code> , <code>fit_tte_data()</code> or <code>fit_KM()</code> .
<code>data</code>	Name of patient-level data set, used to generate the KM plot.
<code>Time</code>	The column name for the times. Default is "Time"
<code>Event</code>	The column name for the events column (i.e. the binary variable denoting events vs censorings). Default is "Event"
<code>censoringOne</code>	Specify whether censoring is denoted in the Event column by a one (TRUE) or zero (FALSE). Default=FALSE (censorings denoted by 0, events by 1)
<code>CI</code>	Number between 0 and 1 for the size of Kaplan Meier confidence interval to calculate. Default is 0.95 (95 percent confidence interval).
<code>colour_CI</code>	Boolean for whether to colour the fitting confidence interval area. Default=TRUE (colour area)
<code>maxT</code>	Maximum time to calculate point estimate and CIs up to. Default=NULL (Calculate up to last time in patient data)
<code>xlim</code>	Standard graphical parameter for x-axis limits. Default=NULL (Plots from 0 to maximum patient time)
<code>ylim</code>	Graphical parameter for y-axis limits. Default=c(0,1) (Plots survival function from 0 to 1)
<code>main</code>	String for plot title. Default="Kaplan Meier Curve Fit Plot"
<code>fit_col</code>	Colour for fitting curve Default=2 (red)
<code>km_col</code>	Colour for km curve Default=1 (black)
<code>area_col</code>	Colour for CI area Default="skyblue" (sky blue)
<code>CI_col</code>	Colour for CI Default=4 (blue)
<code>CI_lty</code>	Line type for CI Default=2 (dashed)
<code>no_legend</code>	Boolean to turn off legend. Default is FALSE; legend shown.
<code>legend_position</code>	String with "top_right", or "bottom_left", corresponding to legend position in power plot. (Default="bottom_left").
<code>overlay</code>	Boolean whether to overlay on existing plot (vs start a new one). Default=FALSE
<code>...</code>	Additional graphical parameters.

Value

Returns NULL

Author(s)

James Bell

Examples

```

recruit <- PieceR(matrix(c(rep(1,12),10,15,25,30,45,60,55,50,65,60,55,30),ncol=2),1)
trial_long <- simulate_trials(active_ecurve=Weibull(50,0.8),control_ecurve=Weibull(50,0.8),
rcurve=recruit,fix_events=200, iterations=1,seed=12345,detailed_output=TRUE)
trial_short <- set_assess_time(data=trial_long,time=10,detailed_output = FALSE)

maxtime <- max(ceiling(trial_long[, "Assess"]))
events <- rep(NA,maxtime)
for (i in 1:maxtime){events[i] <- sum(1-set_assess_time(trial_long,i)[, "Censored"])}

predictions <- event_prediction(data=trial_short, Event="Censored", censoringOne=TRUE,
type="Weibull", rcurve=recruit, max_time=60, cond_Events=49, cond_NatRisk=451,
cond_Time=10, units="Months")

plot_km_fit(fit=predictions,data=trial_short,Event="Censored",censoringOne=TRUE,maxT=70)

```

plot_nph

*Plot output from nph_traj***Description**

This function plots the output from `nph_traj()`.

By default, it produces 6 plots:

- "KM plot" Kaplan Meier plot for events. This is in patient time.
- "Censoring plot" Plot of CDFs for censoring functions. This is in patient time.
- "Recruitment plot" Number of patients expected to have been recruited over time. This is in trial time.
- "Event plot" Total number of events expected to occur over time. This is in trial time.
- "log(HR) plot" Expected log(HR), with expected confidence interval, over time. This is in trial time.
- "Power plot" Expected power over time for various methods. This is in trial time.

Plots may be omitted via arguments.

All calculated powers automatically plotted unless specified otherwise.

Usage

```
plot_npht(
  data,
  KM = TRUE,
  censor = TRUE,
  recruitment = TRUE,
  events = TRUE,
  logHR = TRUE,
  power = TRUE,
  include_frontier = TRUE,
  include_RMST = TRUE,
  include_landmark = TRUE,
  alpha1 = 0.025,
  legend_position = c("top_left", "top_right", "bottom_right")
)
```

Arguments

<code>data</code>	Full output list from <code>nph_traj()</code>
<code>KM</code>	Boolean to include KM plot (Default = TRUE)
<code>censor</code>	Boolean to include censoring plot (Default = TRUE)
<code>recruitment</code>	Boolean to include recruitment plot (Default = TRUE)
<code>events</code>	Boolean to include events plot (Default = TRUE)
<code>logHR</code>	Boolean to include log(HR) plot (Default = TRUE)
<code>power</code>	Boolean to include power plot (Default = TRUE)
<code>include_frontier</code>	Boolean to include frontier power curve in power plot (Default = TRUE)
<code>include_RMST</code>	Boolean to include RMST power curve in power plot if available (Default = TRUE)
<code>include_landmark</code>	Boolean to include landmark power curve in power plot if available (Default = TRUE)
<code>alpha1</code>	One-sided alpha to use for estimation of log(HR) confidence intervals (Default = 0.025)
<code>legend_position</code>	String with any of "top_left", "top_right" or "bottom_right", corresponding to legend position in power plot. Default is "top_left".

Value

Returns NULL

Author(s)

James Bell

Examples

```
trial <- nphtraj(Weibull(100,1),Weibull(70,1),rcurve=LinearR(12,100,100),RMST=20,
  landmark=20,max_assessment=30)

plot_npht(trial)
plot_npht(data=trial,KM=FALSE,censor=FALSE,recruitment=FALSE)
plot_npht(data=trial,KM=FALSE,censor=FALSE,recruitment=FALSE,events=FALSE,logHR=FALSE,
  include_frontier=FALSE, include_RMST=FALSE,include_landmark=FALSE,legend_position="top_right")
```

random_draw

*Method for taking random draws from a Curve object distribution***Description**

This takes random draws from the Curve object distribution Note that the seed should always be appropriately set before invoking this

Usage

```
random_draw(theObject, ...)
```

Arguments

theObject	The name of the Curve Object
...	Pass-through arguments

Examples

```
random_draw(Weibull(100,1),1000)
```

random_draw, Curve-method

*Method for taking random draws from a Curve object distribution***Description**

This takes random draws from the Curve object distribution Note that the seed should always be appropriately set before invoking this

Usage

```
## S4 method for signature 'Curve'
random_draw(theObject, n)
```

Arguments

theObject The name of the Curve Object
 n Number of random draws required

Examples

```
random_draw(Weibull(100,1),1000)
```

 RCurve-class

RCurve Class for defining recruitment distributions

Description

This class extends the Curve class, adding recruitment-related quantities such as patient numbers.

Slots

N Total number of patients recruited.
 Nactive Number of patients recruited in active arm. $N_{\text{active}} + N_{\text{control}} = N$.
 Ncontrol Number of patients recruited in control arm. $N_{\text{active}} + N_{\text{control}} = N$.
 Ratio Randomisation ratio. $N_{\text{active}} \text{ divided by } N_{\text{control}} = \text{Ratio}$.
 Length Total length of the recruitment period.
 RF Name of the random generator function describing the Curve.
 inverse Name of the inverse CDF function describing the Curve. Optional; set to NULL if unavailable.
 paramno Number of parameters required to define the distribution.
 pnames Names of parameters defining the distribution. Should be a vector of length paramno.
 pnames Values of parameters defining the distribution. Should be a list of length paramno.
 maxF Maximum length of patient follow-up. Typically should be Inf.

Author(s)

James Bell

Examples

```
new("RCurve", type="ExampleCurve", PDF="pdf_fn_name", CDF="CDF_fn_name",
    RF="random_draw_fn_name", inverse="inv_fn_name", paramno=2, pnames=c('param1','param2'),
    pvalue=list(1,2),
    N=100, Nactive=50, Ncontrol=40, Ratio=50/40, Length = 5, maxF = Inf)
```

run_gestate	<i>Load Shiny for Gestate Loads the Shiny interactive GUI for gestate</i>
-------------	---

Description

Load Shiny for Gestate Loads the Shiny interactive GUI for gestate

Usage

```
run_gestate()
```

Examples

```
run_gestate()
```

setPatients	<i>Method for setting N's in an RCurve</i>
-------------	--

Description

This sets the RCurve patient numbers per arm directly and updates N and Ratio accordingly

Usage

```
setPatients(theObject, ...)
```

Arguments

theObject	The name of the RCurve Object
...	Pass-through arguments

Examples

```
setPatients(LinearR(12,100,100),200,150)
```

 setPatients, RCurve-method

Method for setting N's in an RCurve

Description

This sets the RCurve patient numbers per arm directly and updates N and Ratio accordingly

Usage

```
## S4 method for signature 'RCurve'
setPatients(theObject, Nactive, Ncontrol)
```

Arguments

theObject	The name of the RCurve Object
Nactive	Number of patients to set in active arm
Ncontrol	Number of patients to set in control arm

Examples

```
setPatients(LinearR(12,100,100),200,150)
```

 set_assess_time

Adjusts assessment time for simulations

Description

Function for modifying the assessment time of simulate_trials() or simulate_trials_strata() simulations.

It is set up to automatically read in either matrix or list formats from simulate_trials() or simulate_trials_strata(), and only these inputs are supported.

Note that if recruitment had not finished in the input then any increases in assessment time cannot account for the missing patients. It is therefore strongly recommended to initially simulate for at least the duration of the recruitment before reducing the number to missing patients.

This function can also be used to change format and/or slim down data for time-driven simulations.

Usage

```
set_assess_time(
  data,
  time,
  output_type = c("input", "matrix", "list"),
  detailed_output = TRUE,
  Time = "Time",
  Event = "Censored",
  censoringOne = TRUE,
  Iter = "Iter"
)
```

Arguments

data	Output file from simulate_trials() or simulate_trials_strata() created with detailed_output=TRUE, in either "list" or "matrix" format. Only these formats are supported.
time	Positive number specifying the required assessment time.
output_type	Choice of "input" (output in same format as input), "matrix" (matrix format output) or "list" (list format output). Default="input".
detailed_output	Boolean to require full details of timings of competing processes. If FALSE, the simplified data only includes the *'ed output columns - this approximately halves RAM requirements. Default=TRUE (detailed).
Time	String specifying the name of the time column. Default="Time"
Event	String specifying the name of the censoring/event column. Default="Censored" (and by default it is a censoring column unless censoringOne=FALSE)
censoringOne	Boolean specifying whether censoring is denoted in the censoring/event column by a one (TRUE) or zero (FALSE). Default=TRUE (censorings denoted by 1, events by 0)
Iter	String specifying the name of the iterations column. Default="Iter"

Value

Returns the input simulated trial, in either matrix or list format, with modified assessment times. All columns dependent on this are also updated.

Author(s)

James Bell

Examples

```
example_sim <- simulate_trials(active_ecurve=Weibull(250,0.8),control_ecurve=Weibull(100,1),
  rcurve=LinearR(12,100,100), assess=20, iterations=5,seed=12345,detailed_output=TRUE)

adjusted_example <- set_assess_time(data=example_sim,time=10)
```

set_event_number	<i>Adjusts simulations so that administrative censoring occurs at a fixed event number, rather than a fixed time</i>
------------------	--

Description

Function for converting trials simulated by `simulate_trials()` or `simulate_trials_strata()` from a fixed censoring time to a fixed number of total events.

It is set up to automatically read in either matrix or list formats from `simulate_trials()` or `simulate_trials_strata()`, and only these inputs are supported.

Note that if recruitment had not finished in the input then any increases in assessment time cannot account for the missing patients. It is therefore strongly recommended to initially simulate for at least the duration of the recruitment before fixing the event number.

This function can also be used to change format and/or slim down data for event-driven simulations.

Usage

```
set_event_number(
  data,
  events,
  output_type = c("input", "matrix", "list"),
  detailed_output = TRUE,
  Time = "Time",
  Event = "Censored",
  censoringOne = TRUE,
  Iter = "Iter"
)
```

Arguments

data	Output file from <code>simulate_trials()</code> or <code>simulate_trials_strata()</code> created with <code>detailed_output=TRUE</code> , in either "list" or "matrix" format. Only these formats are supported.
events	Positive integer specifying the required number of events.
output_type	Choice of "input" (output in same format as input), "matrix" (matrix format output) or "list" (list format output). Default="input".
detailed_output	Boolean to require full details of timings of competing processes. If FALSE, the simplified data only includes the *'ed output columns - this approximately halves RAM requirements. Default=TRUE (detailed).
Time	String specifying the name of the time column. Default="Time"
Event	String specifying the name of the censoring/event column. Default="Censored" (and by default it is a censoring column unless <code>censoringOne=FALSE</code>)

censoringOne	Boolean specifying whether censoring is denoted in the censoring/event column by a one (TRUE) or zero (FALSE). Default=TRUE (censorings denoted by 1, events by 0)
Iter	String specifying the name of the iterations column. Default="Iter"

Value

Returns the input simulated trial, in either matrix or list format, with modified assessment times. All columns dependent on this are also updated.

Author(s)

James Bell

Examples

```
example_sim <- simulate_trials(active_ecurve=Weibull(250,0.8),control_ecurve=Weibull(100,1),
rcurve=LinearR(12,100,100), assess=20, iterations=5, seed=12345, detailed_output=TRUE)

adjusted_examples <- set_event_number(data=example_sim, events=50)
```

show, Curve-method	<i>Method for displaying Curve objects neatly - replaces standard show method</i>
--------------------	---

Description

This is the standard print method for a Curve object

Usage

```
## S4 method for signature 'Curve'
show(object)
```

Arguments

object	The name of the Curve Object
--------	------------------------------

Examples

```
Weibull(100,1)
```

show, RCurve-method	<i>Method for displaying RCurve objects neatly - replaces standard show method</i>
---------------------	--

Description

This is the standard print method for an RCurve Object

Usage

```
## S4 method for signature 'RCurve'
show(object)
```

Arguments

object	The name of the RCurve Object
--------	-------------------------------

Examples

```
LinearR(12,100,100)
```

simulate_trials	<i>Perform simulations of time-to-event data using arbitrary event, censoring and recruitment distributions.</i>
-----------------	--

Description

Function for simulating generalised two-arm time-to-event trial data for NPH trials with arbitrary event, censoring and recruitment distributions.

Event and censoring distributions are specified via Curve objects, with recruitment specified through an RCurve object. As it uses same architecture and similar syntax to `nph_traj()`, analysis results ought to be directly comparable. It is designed to complement `nph_traj()`, either as a stochastic alternative, or as a means to validate its outputs. It can also be used to build more complex simulations by combining the outputs of multiple runs; e.g. multi-arm trials.

Data sets created by this function are formatted so they may be automatically recognised and analysed by `analyse_sim()`.

Usage

```
simulate_trials(
  active_ecurve,
  control_ecurve,
  active_dcurve = Blank(),
  control_dcurve = Blank(),
  rcurve,
```

```

    assess = NULL,
    fix_events = NULL,
    iterations,
    seed,
    detailed_output = FALSE,
    output_type = c("matrix", "list"),
    Time = "Time",
    Event = "Censored",
    censoringOne = TRUE,
    Trt = "Trt",
    Iter = "Iter"
)

```

Arguments

active_ecurve	Event distribution for the active arm, specified as a Curve object
control_ecurve	Event distribution for the control arm, specified as a Curve object
active_dcurve	Dropout/censoring distribution for the active arm, specified as a Curve object. By default, Blank(), i.e. no dropout.
control_dcurve	Dropout/censoring distribution for the control arm, specified as a Curve object. By default, Blank(), i.e. no dropout.
rcurve	Recruitment distribution, specified as an RCurve object
assess	Positive number for the assessment time at which administrative censoring will be performed.
fix_events	Positive integer for the number of events to fix (if required), letting the assessment time vary. Alternatively, NULL for fixed time assessment with variable event numbers. Notes: Fixing event numbers overrides any specified assessment time and slows simulation considerably. Default = NULL (fixed analysis time)
iterations	Number of simulations to perform. Depending on trial size, 10,000-20,000 is typically OK to analyse on 8GB RAM.
seed	Seed number to use. Numerical, although if "Rand" is specified, a system-time-derived number will be used.
detailed_output	Boolean to require full details of timings of competing processes. Full details required for any future adjustments to assessment time. Simplified option approximately halves RAM requirements. Default=FALSE (simplified).
output_type	"matrix" or "list" specifying the type of output required. "matrix" requests a single matrix with a column "iter" to denote the simulation, while "list" creates a list with one entry per simulation. Default="matrix".
Time	String specifying the name of the output time column. Default="Time"
Event	String specifying the name of the output censoring/event column. Default="Censored" (and by default it is a censoring column unless censoringOne=FALSE)
censoringOne	Boolean specifying whether censoring is denoted in the censoring/event column by a one (TRUE) or zero (FALSE). Default=TRUE (censorings denoted by 1, events by 0)

Trt	String specifying the name of the output treatment column. Default="Trt"
Iter	String specifying the name of the output iterations column. Default="Iter"

Value

Returns a table with one row per patient per simulation. Table contains the following columns:

- "Time" Simulated actually observed (patient) time of event or censoring: This is the main column of interest for analysis. Named by Time argument.
- "Event" Simulated event indicator denoting censorings/events as 1/0 if censoringOne=TRUE and 0/1 if censoringOne=FALSE. Named by Event argument.
- "Trt" Treatment group number - 1 is active, 2 is control. Named by Trt argument.
- "Iter" Simulation number. Named by Iter argument.
- "ETime" Simulated actual event (patient) time (may or may not be observed). Only produced when detailed_output=TRUE.
- "CTime" Simulated actual censoring/dropout (patient) time (may or may not be observed). Only produced when detailed_output=TRUE.
- "Rec_Time" Simulated (trial) time of recruitment. Only produced when detailed_output=TRUE.
- "Assess" Prespecified (trial) time of assessment. Only produced when detailed_output=TRUE.
- "Max_F" Prespecified maximum patient follow-up time. Only produced when detailed_output=TRUE.
- "RCTime" Simulated actual administrative censoring (patient) time (may or may not be observed)

Author(s)

James Bell

Examples

```
example_sim <- simulate_trials(active_ecurve=Weibull(250,0.8),control_ecurve=Weibull(100,1),
rcurve=LinearR(12,100,100), assess=20, iterations=5,seed=12345)
```

simulate_trials_strata

Perform multi-strata simulations of time-to-event data using arbitrary event, censoring and recruitment distributions.

Description

Function for simulating generalised two-arm multi-strata time-to-event trial data for NPH trials with arbitrary event, censoring and recruitment distributions.

Acts as a wrapper for `simulate_trials`.

Vector of strata proportions supplies number of strata. Event and censoring distributions specified via lists of Curve objects. If only one Curve supplied then assumed to be common to all strata. Recruitment specified via a single RCurve object.

As it uses same architecture and similar syntax to `nph_traj()`, results in simple cases may be directly comparable to e.g. use of `MixExp()` or `MixWei()` Curves.

Can be used to validate outputs from `nph_traj()`.

Data sets from this are set up to be automatically analysed with the `analyse_sim` function (including stratified analysis if you provide it the name of stratum column).

Usage

```
simulate_trials_strata(
  stratum_probs,
  active_ecurve,
  control_ecurve,
  active_dcurve = Blank(),
  control_dcurve = Blank(),
  rcurve,
  assess = NULL,
  fix_events = NULL,
  stratum_name = "Stratum",
  iterations,
  seed,
  detailed_output = FALSE,
  output_type = c("matrix", "list"),
  Time = "Time",
  Event = "Censored",
  censoringOne = TRUE,
  Trt = "Trt",
  Iter = "Iter"
)
```

Arguments

- | | |
|----------------|---|
| stratum_probs | Vector of probabilities that patients belong to each stratum. Must sum to 1. Its length determines the number of strata. |
| active_ecurve | List of event distributions for the active arm, specified as a list of Curve objects. If single Curve is specified, will be used for all strata. |
| control_ecurve | List of event distributions for the control arm, specified as a list of Curve objects. If single Curve is specified, will be used for all strata. |
| active_dcurve | List of dropout/censoring distribution for the active arm, specified as a Curve object. If single Curve is specified, will be used for all strata. By default, <code>Blank()</code> , i.e. no dropout in any stratum. |

control_dcurve	List of dropout/censoring distribution for the control arm, specified as a Curve object. If single Curve is specified, will be used for all strata. By default, Blank(), i.e. no dropout in any stratum.
rcurve	Recruitment distribution, specified as a single RCurve object.
assess	Positive number for the assessment time at which administrative censoring will be performed.
fix_events	Positive integer for the number of events to fix (if required), letting the assessment time vary. Alternatively, NULL for fixed time assessment with variable event numbers. Notes: Fixing event numbers overrides any specified assessment time and slows simulation considerably. Default = NULL (fixed analysis time)
stratum_name	Name of the column defining the stratum. Default="Stratum".
iterations	Number of simulations to perform. Depending on trial size, 10,000-20,000 is typically OK to analyse on 8GB RAM.
seed	Seed number to use. Numerical, although if "Rand" is specified, a system-time-derived number will be used.
detailed_output	Boolean to require full details of timings of competing processes. Full details required for any future adjustments to assessment time. Simplified option approximately halves RAM requirements. Default=FALSE (simplified).
output_type	"matrix" or "list" specifying the type of output required. "matrix" requests a single matrix with a column "iter" to denote the simulation, while "list" creates a list with one entry per simulation. Default="matrix".
Time	String specifying the name of the output time column. Default="Time"
Event	String specifying the name of the output censoring/event column. Default="Censored" (and by default it is a censoring column unless censoringOne=FALSE)
censoringOne	Boolean specifying whether censoring is denoted in the censoring/event column by a one (TRUE) or zero (FALSE). Default=TRUE (censorings denoted by 1, events by 0)
Trt	String specifying the name of the output treatment column. Default="Trt"
Iter	String specifying the name of the output iterations column. Default="Iter"

Value

Returns a table with one row per patient per simulation. Table contains the following columns:

- "Time" Simulated actually observed (patient) time of event or censoring: This is the main column of interest for analysis. Named by Time argument.
- "Event" Simulated event indicator denoting censorings/events as 1/0 if censoringOne=TRUE and 0/1 if censoringOne=FALSE. Named by Event argument.
- "Trt" Treatment group number - 1 is active, 2 is control. Named by Trt argument.
- "Iter" Simulation number. Named by Iter argument.
- "Stratum" Stratum number. Column name will be the value of the stratum_name argument.)

- "ETime" Simulated actual event (patient) time (may or may not be observed). Only produced when detailed_output=TRUE.
- "CTime" Simulated actual censoring/dropout (patient) time (may or may not be observed). Only produced when detailed_output=TRUE.
- "Rec_Time" Simulated (trial) time of recruitment. Only produced when detailed_output=TRUE.
- "Assess" Prespecified (trial) time of assessment. Only produced when detailed_output=TRUE.
- "Max_F" Prespecified maximum patient follow-up time. Only produced when detailed_output=TRUE.
- "RCTime" Simulated actual administrative censoring (patient) time (may or may not be observed). Only produced when detailed_output=TRUE.)

Author(s)

James Bell

Examples

```
example_strat_sim <- simulate_trials_strata(stratum_probs=c(0.5,0.5),
  active_ecurve=c(Weibull(250,0.8),Weibull(100,1)), control_ecurve=Weibull(100,1),
  rcurve=LinearR(12,100,100), assess=20, iterations=5, seed=12345)
```

summarise_analysis	<i>Summarise analyses of simulations of time-to-event data using arbitrary event, censoring and recruitment distributions.</i>
--------------------	--

Description

Function for summarising the analyses of simulated time-to-event trial data produced by analyse_sim().

Automatically reads in format from analyse_sim(); no other input format is supported.

It automatically detects types of analysis performed and provides relevant summaries (log-rank, Cox, RMST, landmark).

Usage

```
summarise_analysis(analysed_results, alpha1 = 0.025)
```

Arguments

analysed_results

Output file from analyse_sim(). Only analyse_sim() output is supported.

alpha1

1-sided alpha to be used for analysis. Default=0.025.

Value

Returns a table with one row. Table contains the following columns:

- "Simulations" Number of simulations conducted
- "HR" Exponent of Mean Cox Log Hazard Ratio (LR/Cox analysis only)
- "LogHR" Mean Cox Log Hazard Ratio (LR/Cox analysis only)
- "LogHR_SE" Root mean square of the Cox Standard Errors for Log Hazard Ratio (LR/Cox analysis only)
- "HR_Z" Mean Cox Z-Score (LR/Cox analysis only)
- "HR_P" p-value of Mean Cox Z-Score (LR/Cox analysis only)
- "HR_Power" Simulated power of Cox-regression (LR/Cox analysis only)
- "HR_Failed" Proportion of simulations failing to calculate a Cox HR (LR/Cox analysis only)
- "LR_Z" Mean Log-Rank Test Z-Score (LR/Cox analysis only)
- "LR_P" p-value of Mean Log-Rank Test Z-Score (LR/Cox analysis only)
- "LR_Power" Simulated power of the log-rank test (LR/Cox analysis only)
- "LR_Failed" Proportion of simulations failing to calculate a log-rank test statistic (LR/Cox analysis only)
- "Events_Active" Mean events in active arm (LR/Cox analysis only)
- "Events_Control" Mean events in control arm (LR/Cox analysis only)
- "Events_Total" Mean total events (LR/Cox analysis only)
- "RMST_Time" Restriction time for RMST analysis (RMST analysis only)
- "RMST_Control" Mean RMST for active arm (RMST analysis only)
- "RMST_C_SE" Root mean square of RMST Standard Errors for active arm (RMST analysis only)
- "RMST_Active" Mean RMST for control arm (RMST analysis only)
- "RMST_A_SE" Root mean square of RMST Standard Errors for control arm (RMST analysis only)
- "RMST_Delta" Mean RMST difference between arms active-control (RMST analysis only)
- "RMST_D_SE" Root mean square of RMST difference Standard Errors (RMST analysis only)
- "RMST_Power" Simulated power of RMST (RMST analysis only)
- "RMST_Failed" Proportion of simulations failing to calculate the RMST (RMST analysis only)
- "LM_Time" Landmark analysis time, i.e. assessment time of Survival function (Landmark analysis only)
- "LM_Control" Mean survival function for active arm at landmark time (Landmark analysis only)
- "LM_C_SE" Root mean square of Greenwood standard errors for active arm at landmark time (Landmark analysis only)
- "LM_Active" Mean survival function for control arm at landmark time (Landmark analysis only)

- "LM_A_SE" Root mean square of Greenwood standard errors for control arm at landmark time (Landmark analysis only)
- "LM_Delta" Mean survival function difference active-control at landmark time (Landmark analysis only)
- "LM_D_SE" Root mean square of Greenwood standard errors for survival differences at landmark time (Landmark analysis only)
- "LM_Power" Power of landmark analysis (Landmark analysis only)
- "LM_Failed" Proportion of simulations failing to calculate the survival difference at landmark time (Landmark analysis only)

Author(s)

James Bell

Examples

```
example_sim <- simulate_trials(active_ecurve=Weibull(250,0.8),control_ecurve=Weibull(100,1),
rcurve=LinearR(12,100,100), assess=40, iterations=5,seed=12345,detailed_output=TRUE)

example_analysis1 <- analyse_sim(example_sim)
example_analysis2 <- analyse_sim(data=example_sim,RMST=30,landmark=30)

example_summary1 <- summarise_analysis(example_analysis1)
example_summary2 <- summarise_analysis(example_analysis2)
```

survfit_fast

Create lifetable quickly

Description

Simple, fast function to generate basic lifetables. No error checking is performed.

Usage

```
survfit_fast(time, events)
```

Arguments

time	Vector of event/censoring times
events	Vector of indicators for whether each time is an event (1) or a censoring (0). Must be same length vector as time argument.

Value

Returns a list format lifetable with the following entries:

- "n" Number of patients
- "time" Vector of times
- "n.risk" Vector of numbers of patients at risk at each time
- "n.event" Vector of numbers of patients with an event at each time
- "n.censor" Vector of numbers of patients censored at each time
- "surv" Vector of estimates of Survival function at each time
- "se" Vector of standard errors of estimates of Survival function at each time

Author(s)

James Bell

Examples

```
survfit_fast(c(1,2,3,3,4,5),c(1,0,1,1,0,1))
```

Weibull

Weibull Curve constructor function

Description

This creates a Curve object for a Weibull distribution.

Curve objects contain all necessary information to describe a distribution, including functions and parameters describing it.

Parameterisation follows that used by pweibull etc. See Details for more information on parameterisation.

Usage

```
Weibull(alpha, beta = 1)
```

Arguments

alpha	Scale parameter for Weibull distribution.
beta	Shape parameter for Weibull distribution. Default is 1; an exponential distribution.

Details

The Weibull distribution with shape parameter beta and scale parameter alpha has parameterisation:

$$f(x) = (\beta/\alpha) (x/\alpha)^{(\beta-1)} \exp(- (x/\alpha)^\beta)$$

$$F(x) = 1 - \exp(- (x/\alpha)^\beta)$$

Author(s)

James Bell

Examples

```
Weibull(alpha=100,beta=0.8)
```

Index

analyse_sim, [3](#)

Blank, [6](#)

create_tte_prior, [8](#)
createRFfunction, [7](#)
createRFfunction, Curve-method, [7](#)
Curve-class, [9](#)

evaluateCDFfunction, [10](#)
evaluateCDFfunction, Curve-method, [10](#)
evaluateInvfunction, [11](#)
evaluateInvfunction, Curve-method, [11](#)
evaluatePDFfunction, [12](#)
evaluatePDFfunction, Curve-method, [12](#)
event_prediction, [13](#)
event_prediction_KM, [16](#)
event_prediction_prior, [18](#)
Exponential, [22](#)

fit_KM, [23](#)
fit_tte_data, [25](#)
fit_tte_data_prior, [26](#)
frontierpower, [28](#)

getAssessCDFfunction, [29](#)
getAssessCDFfunction, RCurve-method, [30](#)
getCDFfunction, [30](#)
getCDFfunction, Curve-method, [31](#)
getInvfunction, [31](#)
getInvfunction, Curve-method, [32](#)
getLength, [32](#)
getLength, RCurve-method, [33](#)
getMaxF, [33](#)
getMaxF, RCurve-method, [34](#)
getN, [34](#)
getN, RCurve-method, [35](#)
getNactive, [35](#)
getNactive, RCurve-method, [36](#)
getNames, [36](#)
getNames, Curve-method, [37](#)

getNcontrol, [37](#)
getNcontrol, RCurve-method, [38](#)
getParam, [38](#)
getParam, Curve-method, [39](#)
getParams, [39](#)
getParams, Curve-method, [40](#)
getParamsV, [40](#)
getParamsV, Curve-method, [41](#)
getPatients, [41](#)
getPatients, RCurve-method, [42](#)
getPDFfunction, [42](#)
getPDFfunction, Curve-method, [43](#)
getRatio, [43](#)
getRatio, RCurve-method, [44](#)
getRFfunction, [44](#)
getRFfunction, Curve-method, [45](#)
getType, [45](#)
getType, Curve-method, [46](#)
GGamma, [46](#)
Gompertz, [47](#)

InstantR, [48](#)

LinearR, [48](#)
LogLogistic, [49](#)
Lognormal, [50](#)

MixExp, [51](#)
MixWei, [52](#)

nph_traj, [53](#)

PieceExponential, [56](#)
PieceR, [57](#)
PieceRMaxF, [58](#)
plot_ep, [63](#)
plot_km_fit, [65](#)
plot_npht, [67](#)
plotCDF, [59](#)
plotCDF, Curve-method, [59](#)
plotRecruitment, [60](#)

plotRecruitment, RCurve-method, [61](#)
plotSF, [62](#)
plotSF, Curve-method, [62](#)

random_draw, [69](#)
random_draw, Curve-method, [69](#)
RCurve-class, [70](#)
run_gestate, [71](#)

set_assess_time, [72](#)
set_event_number, [74](#)
setPatients, [71](#)
setPatients, RCurve-method, [72](#)
show, Curve-method, [75](#)
show, RCurve-method, [76](#)
simulate_trials, [76](#)
simulate_trials_strata, [78](#)
summarise_analysis, [81](#)
survfit_fast, [83](#)

Weibull, [84](#)