

Package ‘psm3mkv’

July 11, 2026

Title Evaluate Partitioned Survival and State Transition Models

Version 0.3.3

Description Fits and evaluates three-state partitioned survival analyses (PartSAs) and Markov models (clock forward or clock reset) to progression and overall survival data typically collected in oncology clinical trials. These model structures are typically considered in cost-effectiveness modeling in advanced/metastatic cancer indications. Muston (2024). ``Informing structural assumptions for three state oncology cost-effectiveness models through model efficiency and fit". Applied Health Economics and Health Policy.

License GPL (>= 3)

URL <https://merck.github.io/psm3mkv/>, <https://github.com/Merck/psm3mkv>

BugReports <https://github.com/Merck/psm3mkv/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports admiral, dplyr, flexsurv, ggplot2, pharmaverseadam, purrr, rlang, SimplicialCubature, stats, survival, stringr, tibble, tidyr, admiralonco

Suggests boot, covr, ggsci, HMDHFDplus, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Dominic Muston [aut, cre] (ORCID: <https://orcid.org/0000-0003-4876-7940>), Merck & Co., Inc., Rahway, NJ, USA and its affiliates [cph, fnd]

Maintainer Dominic Muston <dom.muston@gmail.com>

Repository CRAN

Date/Publication 2026-07-10 23:10:09 UTC

Contents

calc_allrmds	2
calc_haz_psm	4
calc_likes	5
calc_rmd	7
calc_surv_psmpps	8
check_consistent_pfs	9
compare_psm_likes	10
constrain_survprob	11
create_dummydata	12
create_extrafields	13
find_bestfit	13
fit_ends_mods_par	14
fit_ends_mods_spl	16
graph_psm_hazards	17
graph_psm_survs	18
graph_survs	19
prob_os_psm	21
prob_os_stm_cf	22
prob_os_stm_cr	23
prob_pd_psm	24
prob_pd_stm_cf	25
prob_pd_stm_cr	26
prob_pf_psm	27
prob_pf_stm	28
prob_pps_cf	29
prob_pps_cr	30
vlookup	31
Index	32

calc_allrmds	<i>Calculate restricted mean durations for each health state and all three models</i>
--------------	---

Description

Calculate restricted mean durations for each health state (progression free and progressed disease) for all three models (partitioned survival, clock forward state transition model, clock reset state transition model).

Usage

```
calc_allrmds(
  ptdata,
  inclset = 0,
  dpam,
  psmttype = "simple",
  cuttime = 0,
  Ty = 10,
  lifetable = NA,
  discrate = 0,
  rmdmethod = "int",
  timestep = 1,
  boot = FALSE
)
```

Arguments

ptdata	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring).
inclset	Vector to indicate which patients to include in analysis
dpam	List of statistical fits to each endpoint required in PSM, STM-CF and STM-CR models.
psmttype	Either "simple" or "complex" PSM formulation
cuttime	Time cutoff - this is nonzero for two-piece models.
Ty	Time duration over which to calculate. Assumes input is in years, and patient-level data is recorded in weeks.
lifetable	Optional, a life table. Columns must include ltime (time in years, or 52.18 times shorter than the time index elsewhere, starting from zero) and lx
discrate	Discount rate (% per year)
rmdmethod	can be "int" (default for full integral calculations) or "disc" for approximate discretized calculations
timestep	required if method=="int", default being 1
boot	logical flag to indicate whether abbreviated output is required (default = FALSE), for example for bootstrapping

Value

List of detailed numeric results

- cutadj indicates the survival function and area under the curves for PFS and OS up to the cutpoint
- results provides results of the restricted means calculations, by model and state.

Examples

```
# Create dataset and fit survival models (splines)
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_par(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
# RMD using default "int" method, no lifetable constraint
calc_allrmds(bosonc, dpam=params)
# RMD using discretized ("disc") method, no lifetable constraint
calc_allrmds(bosonc, dpam=params, rmdmethod="disc", timestep=1, boot=TRUE)
```

calc_haz_psm

Derive pre and post-progression hazards of death under PSM

Description

Derive the hazards of death pre- and post-progression under either simple or complex PSM formulations.

Usage

```
calc_haz_psm(timevar, ptdata, dpam, psmttype)
```

Arguments

- | | |
|---------|--|
| timevar | Vector of times at which to calculate the hazards |
| ptdata | Dataset of patient level data. Must be a tibble with columns named: <ul style="list-style-type: none"> • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) |

	• os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring). Survival data for all other endpoints (time to progression, pre-progression death, post-progression survival) are derived from PFS and OS.
dpam	List of survival regressions for each endpoint: • pre-progression death (PPD) • time to progression (TTP) • progression-free survival (PFS) • overall survival (OS) • post-progression survival clock forward (PPS-CF) and • post-progression survival clock reset (PPS-CR).
psmtype	Either "simple" or "complex" PSM formulation

Value

List of pre, the pre-progression hazard, and post, the post-progression hazard

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
calc_haz_psm(0:10, ptdata=bosonc, dpam=params, psmtype="simple")
calc_haz_psm(0:10, ptdata=bosonc, dpam=params, psmtype="complex")
```

Description

Calculate likelihood values and other summary output for the following three state models structures: partitioned survival, clock forward state transition, and clock reset state transition. The function requires appropriately formatted patient-level data, a set of fitted survival regressions, and the time cut-off (if two-piece modeling is used).

Usage

```
calc_likes(ptdata, dpam, cuttime = 0)
```

Arguments

ptdata	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring). Survival data for all other endpoints (time to progression, pre-progression death, post-progression survival) are derived from PFS and OS.
dpam	List of survival regressions for each endpoint: • pre-progression death (PPD) • time to progression (TTP) • progression-free survival (PFS) • overall survival (OS) • post-progression survival clock forward (PPS-CF) and • post-progression survival clock reset (PPS-CR).
cuttime	Time cutoff - this is nonzero for two-piece models.

Value

A list of three tibbles: all is a tibble of results for all patients:

- methname: the model structure or method.
- npar: is the number of parameters used by that method.
- npts_1 to npts_4 are the number of patients experiencing outcomes 1-4 respectively (see below), and npts_tot the total.
- ll_1 to ll_4 are the log-likelihood values for patients experiencing outcomes 1-4 respectively (see below), and ll_tot the total. valid is a tibble of the same design as all but only in patients with valid likelihoods for all 4 methods sum is a tibble in respect of patients with valid likelihoods for all 4 methods providing:
- npts: number of patients contributing results for this method.
- npar: number of parameters used by that method.
- ll: total log-likelihood
- AIC: Akaike Information Criterion value for this model
- BIC: Bayesian Information Criterion value for this model

The four outcomes are as follows:

- (1) refers to patients who remain alive and progression-free during the follow-up;
- (2) refers to patients who die without prior progression during the follow-up;
- (3) refers to patients who progress and then remain alive for the remaining follow-up, and
- (4) refers to patients who progress and die within the follow-up.

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
calc_likes(bosonc, dpam=params)
```

calc_rmd

Calculate restricted mean durations

Description

Calculates the restricted mean duration, given the form of a parametric distribution of Royston-Parmar splines

Usage

```
calc_rmd(Tw, type = NA, spec = NA, survobj = NULL)
```

Arguments

- | | |
|------|---|
| Tw | is the time horizon (weeks) over which the mean should be calculated. |
| type | is either "par" for regular parametric form (exponential, weibull etc) or "spl" for Royston-Parmar splines. |
| spec | <p>is a list comprising: If type=="par": dist is the statistical distribution (named per flexsurv::flexsurvreg) and pars is a vector of the parameters for that distribution.</p> <ul style="list-style-type: none"> • Exponential distribution (exp) requires the rate parameter. • Weibull distribution (both weibullPH and weibull formulations) requires the shape and scale parameters. • Log-logistic distribution (llogis) requires the shape and scale parameters. • Log-normal distribution (lnorm) requires the meanlog and sdlog parameters. |

- Gamma and Gompertz distributions (gamma and gompertz) require the shape and rate parameters.
- Generalized Gamma requires the mu, sigma and Q parameters if using the standard parameterization (gengamma) or shape, scale and k parameters if using the original parameterization (gengamma.orig). If type=="spl":
- gamma - Vector of parameters describing the baseline spline function, as described in [flexsurv::flexsurvspline](#). This may be supplied as a vector with number of elements equal to the length of knots, in which case the parameters are common to all times. Alternatively a matrix may be supplied, with rows corresponding to different times, and columns corresponding to knots.
- knots - Vector of locations of knots on the axis of log time, supplied in increasing order. Unlike in [flexsurv::flexsurvspline](#), these include the two boundary knots.
- scale - Either "hazard", "odds", or "normal", as described in [flexsurv::flexsurvspline](#). With the default of no knots in addition to the boundaries, this model reduces to the Weibull, log-logistic and log-normal respectively. The scale must be common to all times.

survobj is a survival fit object from [flexsurv::flexsurvspline](#) or [flexsurv::flexsurvreg](#)

Value

the restricted mean duration, a numeric value.

Examples

```
calc_rmd(Tw=200,
  type="spl",
  spec=list(gamma=c(0.1,0.2,0.1), knots=c(-5,2,4), scale="normal")
)
calc_rmd(Tw=250,
  type="par",
  spec=list(dist="lnorm", pars=c(3,1))
)
```

calc_surv_psmpps	<i>Derive PPS survival function under a PSM</i>
------------------	---

Description

Derive the post-progression survival (PPS) function under the simple or complex PSM formulation.

Usage

```
calc_surv_psmpps(totime, fromtime = 0, ptdata, dpam, psmttype = "simple")
```

Arguments

totime	Vector of times to which the survival function is calculated
fromtime	Vector of times from which the survival function is calculated
ptdata	Patient-level dataset
dpam	List of fitted survival models for each endpoint
psmtype	Either "simple" or "complex" PSM formulation

Value

Vector of PPS survival function values

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
calc_surv_psmpps(totime=1:10,
  fromtime=rep(1,10),
  ptdata=bosonc,
  dpam=params,
  psmtype="simple")
```

check_consistent_pfs	<i>Check consistency of PFS definition Check that PFS is defined consistently with TTP and OS in a dataset. This convenience function compares pfs.durn with the lower of ttp.durn and os.durn, and checks that the event field pfs.flag is consistent with ttp.flag and os.flag (is 1 when either ttp.flag or os.flag is one).</i>
----------------------	---

Description

Check consistency of PFS definition Check that PFS is defined consistently with TTP and OS in a dataset. This convenience function compares pfs.durn with the lower of ttp.durn and os.durn, and checks that the event field pfs.flag is consistent with ttp.flag and os.flag (is 1 when either ttp.flag or os.flag is one).

Usage

```
check_consistent_pfs(ds)
```

Arguments

- ds** Tibble of complete patient-level dataset
- `ttp.durn`, `pfs.durn`, and `os.durn` are the durations of TTP (time to progression), PFS (progression-free survival), and OS (overall survival).
 - `ttp.flag`, `pfs.flag`, and `os.flag`, and `pps.flag` are event flag indicators for TTP, PFS, and OS respectively (1=event, 0=censoring).

Value

List containing:

- `durn`: Logical vector comparing expected and actual PFS durations
- `flag`: Logical vector comparing expected and actual PFS event flags
- `all`: Single logical value of TRUE if all durations and flags match as expected, FALSE otherwise

Examples

```
ponc <- create_dummydata("pharmaonc")
check_consistent_pfs(ponc)
```

<code>compare_psm_likes</code>	<i>Compare likelihoods of PSMs</i>
--------------------------------	------------------------------------

Description

Compare the total log-likelihood values for the patient-level dataset after fitting PSM-simple and PSM-complex models to each combination of endpoint distributions

Usage

```
compare_psm_likes(ptdata, fitslist, cuttime = 0)
```

Arguments

- ptdata** Dataset of patient level data. Must be a tibble with columns named:
- `ptid`: patient identifier
 - `pfs.durn`: duration of PFS from baseline
 - `pfs.flag`: event flag for PFS (=1 if progression or death occurred, 0 for censoring)
 - `os.durn`: duration of OS from baseline
 - `os.flag`: event flag for OS (=1 if death occurred, 0 for censoring)
 - `ttp.durn`: duration of TTP from baseline (usually should be equal to `pfs.durn`)
 - `ttp.flag`: event flag for TTP (=1 if progression occurred, 0 for censoring).
- fitslist** List of distribution fits to relevant endpoints, after calling `fit_ends_mods_par()` or `fit_ends_mods_spl()`
- cuttime** Time cutoff - this is nonzero for two-piece models.

Value

List containing

- **results**: Dataset of calculation results for each model
- **bests**: Tibble indicating which is the best fitting model individually or jointly, to each end-point, according to AIC or BIC

Examples

```
# Fit parametric distributions to a dataset
bosonc <- create_dummydata("flexbosms")
parfits <- fit_ends_mods_par(bosonc)

splfits <- fit_ends_mods_spl(bosonc)
# Present comparison of likelihood calculations
compare_psm_likes(bosonc, parfits)
compare_psm_likes(bosonc, splfits)
```

constrain_survprob	<i>Constrain survival probabilities according to hazards in a lifetable Recalculated constrained survival probabilities (by week) as the lower of the original unadjusted survival probability and the survival implied by the given lifetable (assumed indexed as years).</i>
--------------------	--

Description

Constrain survival probabilities according to hazards in a lifetable Recalculated constrained survival probabilities (by week) as the lower of the original unadjusted survival probability and the survival implied by the given lifetable (assumed indexed as years).

Usage

```
constrain_survprob(
  survprob1,
  survprob2 = NA,
  lifetable = NA,
  timevec = 0:(length(survprob1) - 1)
)
```

Arguments

survprob1	(Unconstrained) survival probability value or vector
survprob2	Optional survival probability value or vector to constrain on (default = NA)
lifetable	Lifetable (default = NA)
timevec	Vector of times corresponding with survival probabilities above

Value

Vector of constrained survival probabilities

Examples

```
ltable <- tibble::tibble(ltime=0:20, lx=c(1,0.08,0.05,0.03,0.01,rep(0,16)))
survprob <- c(1,0.5,0.4,0.2,0)
constrain_survprob(survprob, lifetable=ltable)
timevec <- 100*(0:4)
constrain_survprob(survprob, lifetable=ltable, timevec=timevec)
survprob2 <- c(1,0.45,0.35,0.15,0)
constrain_survprob(survprob, survprob2)
```

create_dummydata

Create dummy dataset for illustration

Description

Create dummy dataset to illustrate [psm3mkv](#)

Usage

```
create_dummydata(dsname)
```

Arguments

- | | |
|--------|---|
| dsname | <p>Dataset name, as follows:</p> <ul style="list-style-type: none"> • flexbosms provides a dataset based on flexsurv::bosms3(). This contains all the fields necessary for psm3mkv. Durations have been converted from months in the original dataset to weeks. • pharmaonc provides a dataset based on pharmaverseadam::adsl and pharmaverseadam::adrs_onco to demonstrate how this package can be used with ADaM ADTTE datasets. • survcan provides a dataset based on survival::cancer(). This contains the necessary ID and overall survival fields only. Durations have been converted from days in the original dataset to weeks. You will additionally need to supply PFS and TTP data (fields pfs.durn, pfs.flag, ttp.durn and ttp.flag) to use psm3mkv. |
|--------|---|

Value

Tibble dataset, for use with [psm3mkv](#) functions

Examples

```
create_dummydata("survcan") |> head()
create_dummydata("flexbosms") |> head()
create_dummydata("pharmaonc") |> head()
```

create_extrafields	<i>Create the additional time-to-event endpoints, adjusting for cutpoint</i>
--------------------	--

Description

Create the additional time-to-event endpoints, adjusting for cutpoint

Usage

```
create_extrafields(ds, cuttime = 0)
```

Arguments

ds	Patient-level dataset
cuttime	Time cutpoint

Value

Tibble of complete patient-level dataset, adjusted for cutpoint ttp.durn, pfs.durn, pps.durn and os.durn are the durations of TTP (time to progression), PFS (progression-free survival), PPD (pre-progression death) and OS (overall survival) respectively beyond the cutpoint. pps.durn is the duration of survival beyond progression, irrespective of the cutpoint. pps.odurn is the difference between ttp.durn and os.durn (which may be different to pps.durn). ttp.flag, pfs.flag, pps.flag, os.flag, and pps.flag are event flag indicators for TTP, PFS, PPD, OS and PPS respectively (1=event, 0=censoring).

Examples

```
bosonc <- create_dummydata("flexbosms")
create_extrafields(bosonc, cuttime=10)
```

find_bestfit	<i>Find the "best" survival regression from a list of model fits</i>
--------------	--

Description

When there are multiple survival regressions fitted to the same endpoint and dataset, it is necessary to identify the preferred model. This function reviews the fitted regressions and selects that with the minimum Akaike or Bayesian Information Criterion (AIC, BIC), depending on user choice. Model fits must be all parametric or all splines.

Usage

```
find_bestfit(reglist, crit)
```

Arguments

reglist	List of fitted survival regressions to an endpoint and dataset.
crit	Criterion to be used in selection of best fit, either "aic" (Akaike Information Criterion) or "bic" (Bayesian Information Criterion).

Value

List of the single survival regression with the best fit.

Examples

```
bosonc <- create_dummydata("flexbosms")
# Parametric modeling
fits_par <- fit_ends_mods_par(bosonc)
find_bestfit(fits_par$ttp, "aic")

# Splines modeling
fits_spl <- fit_ends_mods_spl(bosonc)
find_bestfit(fits_spl$ttp, "bic")
```

fit_ends_mods_par	<i>Fit multiple parametric survival regressions to the multiple required endpoints</i>
-------------------	--

Description

Fits multiple parametric survival regressions, according to the distributions stipulated, to the multiple endpoints required in fitting partitioned survival analysis, clock forward and clock reset semi-markov models.

Usage

```
fit_ends_mods_par(
  simdat,
  cuttime = 0,
  ppd.dist = c("exp", "weibullPH", "llogis", "lnorm", "gamma", "gompertz"),
  ttp.dist = c("exp", "weibullPH", "llogis", "lnorm", "gamma", "gompertz"),
  pfs.dist = c("exp", "weibullPH", "llogis", "lnorm", "gamma", "gompertz"),
  os.dist = c("exp", "weibullPH", "llogis", "lnorm", "gamma", "gompertz"),
  pps_cf.dist = c("exp", "weibullPH", "llogis", "lnorm", "gamma", "gompertz"),
  pps_cr.dist = c("exp", "weibullPH", "llogis", "lnorm", "gamma", "gompertz"),
  expvar = NA
)
```

Arguments

simdat	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring). Survival data for all other endpoints (time to progression, pre-progression death, post-progression survival) are derived from PFS and OS.
cuttime	Cut-off time for a two-piece model, equals zero for one-piece models.
ppd.dist	Vector of distributions (named per <code>flexsurv::flexsurvreg()</code>) to be fitted to Pre-Progression Death (PPD).
ttp.dist	Vector of distributions (named per <code>flexsurv::flexsurvreg()</code>) to be fitted to Time To Progression (TTP).
pfs.dist	Vector of distributions (named per <code>flexsurv::flexsurvreg()</code>) to be fitted to Progression-Free Survival (PFS).
os.dist	Vector of distributions (named per <code>flexsurv::flexsurvreg()</code>) to be fitted to Overall Survival (OS).
pps_cf.dist	Vector of distributions (named per <code>flexsurv::flexsurvreg()</code>) to be fitted to Post Progression Survival, where time is from baseline (clock forward).
pps_cr.dist	Vector of distributions (named per <code>flexsurv::flexsurvreg()</code>) to be fitted to Post Progression Survival, where time is from progression (clock reset).
expvar	Explanatory variable for modeling of PPS

Value

A list by endpoint, then distribution, each containing two components:

- result: A list of class *flexsurvreg* containing information about the fitted model.
- error: Any error message returned on fitting the regression (NULL indicates no error).

See Also

Spline modeling is handled by `fit_ends_mods_spl()`

Examples

```
bosonc <- create_dummydata("flexbosms")
fit_ends_mods_par(bosonc, expvar=bosonc$ttp.durn)
```

fit_ends_mods_spl	<i>Fit multiple spline regressions to the multiple required endpoints</i>
-------------------	---

Description

Fits multiple survival regressions, according to the distributions stipulated, to the multiple endpoints required in fitting partitioned survival analysis, clock forward and clock reset semi-markov models.

Usage

```
fit_ends_mods_spl(
  simdat,
  knot_set = 1:3,
  scale_set = c("hazard", "odds", "normal"),
  expvar = NA
)
```

Arguments

simdat	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring). Survival data for all other endpoints (time to progression, pre-progression death, post-progression survival) are derived from PFS and OS.
knot_set	is a vector of the numbers of knots to consider, following flexsurv::flexsurvspline() .
scale_set	is a vector of the spline scales to consider, following flexsurv::flexsurvspline() .
expvar	Explanatory variable for modeling of PPS

Value

A list by endpoint, then distribution, each containing two components:

- result: A list of class [flexsurv::flexsurvspline](#) containing information about the fitted model.
- error: Any error message returned on fitting the regression (NULL indicates no error). Also, the given cuttime.

See Also

Parametric modeling is handled by [fit_ends_mods_par\(\)](#)

Examples

```
# Create dataset in suitable form using bos dataset from the flexsurv package
bosonc <- create_dummydata("flexbosms")
fit_ends_mods_spl(bosonc, expvar=bosonc$ttp.durn)
```

graph_psm_hazards *Graph the PSM hazard functions*

Description

Graph the PSM hazard functions

Usage

```
graph_psm_hazards(timevar, endpoint, ptdata, dpam, psmttype)
```

Arguments

timevar	Vector of times at which to calculate the hazards
endpoint	Endpoint for which hazard is required (TTP, PPD, PFS, OS or PPS)
ptdata	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring).
dpam	List of survival regressions for each endpoint: • pre-progression death (PPD) • time to progression (TTP) • progression-free survival (PFS) • overall survival (OS) • post-progression survival clock forward (PPS-CF) and • post-progression survival clock reset (PPS-CR).
psmttype	Either "simple" or "complex" PSM formulation

Value

List containing:

- adj is the hazard adjusted for constraints
- unadj is the unadjusted hazard

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_par(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
# Create graphics
# psmh_simple <- graph_psm_hazards(
#   timerange=(0:10)*6,
#   endpoint="OS",
#   dpam=params,
#   psmttype="simple")
# psmh_simple$graph
```

graph_psm_survs

Graph the PSM survival functions

Description

Graph the PSM survival functions

Usage

```
graph_psm_survs(timevar, endpoint, ptdata, dpam, psmttype)
```

Arguments

timevar	Vector of times at which to calculate the hazards
endpoint	Endpoint for which hazard is required (TTP, PPD, PFS, OS or PPS)
ptdata	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring).
dpam	List of survival regressions for each endpoint: • pre-progression death (PPD)

- time to progression (TTP)
- progression-free survival (PFS)
- overall survival (OS)
- post-progression survival clock forward (PPS-CF) and
- post-progression survival clock reset (PPS-CR).

psmtype Either "simple" or "complex" PSM formulation

Value

List containing:

- adj is the hazard adjusted for constraints
- unadj is the unadjusted hazard

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_par(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
# Graphic illustrating effect of constraints on OS model
psms_simple <- graph_psm_survs(
  timevar=6*(0:10),
  endpoint="OS",
  ptdata=bosonc,
  dpam=params,
  psmtype="simple"
)
psms_simple$graph
```

graph_survs

Graph the observed and fitted state membership probabilities

Description

Graph the observed and fitted state membership probabilities for PF, PD, OS and PPS.

Usage

```
graph_survs(ptdata, dpam, cuttime = 0)
```

Arguments

ptdata	Dataset of patient level data. Must be a tibble with columns named: • ptid: patient identifier • pfs.durn: duration of PFS from baseline • pfs.flag: event flag for PFS (=1 if progression or death occurred, 0 for censoring) • os.durn: duration of OS from baseline • os.flag: event flag for OS (=1 if death occurred, 0 for censoring) • ttp.durn: duration of TTP from baseline (usually should be equal to pfs.durn) • ttp.flag: event flag for TTP (=1 if progression occurred, 0 for censoring). Survival data for all other endpoints (time to progression, pre-progression death, post-progression survival) are derived from PFS and OS.
dpam	List of survival regressions for each endpoint: • pre-progression death (PPD) • time to progression (TTP) • progression-free survival (PFS) • overall survival (OS) • post-progression survival clock forward (PPS-CF) and • post-progression survival clock reset (PPS-CR).
cuttime	is the cut-off time for a two-piece model (default 0, indicating a one-piece model)

Value

List of two items as follows. `data` is a tibble containing data derived and used in the derivation of the graphics. `graph` is a list of four graphics as follows:

- `pf`: Membership probability in PF (progression-free) state versus time since baseline, by method
- `pd`: Membership probability in PD (progressive disease) state versus time since baseline, by method
- `os`: Probability alive versus time since baseline, by method
- `pps`: Probability alive versus time since progression, by method

Examples

```

bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_par(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,

```

```

    pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
  )
  # Create graphics
  gs <- graph_survs(ptdata=bosonc, dpam=params)
  gs$graph$pd

```

prob_os_psm

Calculate probability of being alive in a partitioned survival model

Description

Calculates membership probability of being alive at a particular time (vectorized), given either state transition model (clock forward or clock reset) with given statistical distributions and parameters. This is the sum of membership probabilities in the progression free and progressed disease states.

Usage

```
prob_os_psm(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized)
dpam	List of survival regressions for model endpoints. This must include overall survival (OS).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```

bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_os_psm(0:100, params)

```

prob_os_stm_cf	<i>Calculate probability of being alive under the state transition clock forward model</i>
----------------	--

Description

Calculates membership probability of being alive at a given time (vectorized). This probability is from the state transition clock forward model, according to the given statistical distributions and parameters.

Usage

```
prob_os_stm_cf(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized) from baseline.
dpam	List of survival regressions for model endpoints. This must include pre-progression death (PPD), time to progression (TTP) and post progression survival calculated under the clock forward model (PPS-CF).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_os_stm_cf(0:100, params)
```

prob_os_stm_cr	<i>Calculate probability of being alive under the state transition clock reset model</i>
----------------	--

Description

Calculates membership probability of being alive at a given time (vectorized). This probability is from the state transition clock reset model, according to the given statistical distributions and parameters.

Usage

```
prob_os_stm_cr(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized) from baseline.
dpam	List of survival regressions for model endpoints. This must include pre-progression death (PPD), time to progression (TTP) and post progression survival calculated under the clock reset model (PPS-CR).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_os_stm_cr(0:100, params)
```

prob_pd_psm	<i>Calculate membership probability of progressed disease state in a partitioned survival model</i>
-------------	---

Description

Calculates membership probability of having progressed disease at a particular time (vectorized), given the partitioned survival model with certain statistical distributions and parameters.

Usage

```
prob_pd_psm(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized)
dpam	List of survival regressions for model endpoints. This must include progression-free survival (PFS) and overall survival (OS).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pd_psm(0:100, params)
```

prob_pd_stm_cf	<i>Calculate probability of having progressed disease under the state transition clock forward model</i>
----------------	--

Description

Calculates membership probability of the progressed disease state at a given time (vectorized). This probability is from the state transition clock forward model, according to the given statistical distributions and parameters.

Usage

```
prob_pd_stm_cf(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized) from baseline.
dpam	List of survival regressions for model endpoints. This must include pre-progression death (PPD), time to progression (TTP) and post progression survival calculated under the clock forward model (PPS-CF).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pd_stm_cf(0:100, params)
```

prob_pd_stm_cr	<i>Calculate probability of having progressed disease under the state transition clock reset model</i>
----------------	--

Description

Calculates membership probability of the progressed disease state at a given time (vectorized). This probability is from the state transition clock reset model, according to the given statistical distributions and parameters.

Usage

```
prob_pd_stm_cr(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized) from baseline.
dpam	List of survival regressions for model endpoints. This must include pre-progression death (PPD), time to progression (TTP) and post progression survival calculated under the clock reset model (PPS-CR).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pd_stm_cr(0:100, params)
```

prob_pf_psm	<i>Calculate probability of being progression free in partitioned survival model</i>
-------------	--

Description

Calculates membership probability for the progression free state, at a particular time (vectorized), given a partitioned survival model with given statistical distributions and parameters.

Usage

```
prob_pf_psm(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized)
dpam	List of survival regressions for model endpoints. This must include progression-free survival (PFS).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pf_psm(0:100, params)
```

prob_pf_stm	<i>Calculate probability of being progression free in either state transition model (clock forward or clock reset)</i>
-------------	--

Description

Calculates membership probability for the progression free state, at a particular time (vectorized), given either state transition model (clock forward or clock reset) with given statistical distributions and parameters.

Usage

```
prob_pf_stm(time, dpam, starting = c(1, 0, 0))
```

Arguments

time	Time (numeric and vectorized)
dpam	List of survival regressions for model endpoints. This must include pre-progression death (PPD) and time to progression (TTP).
starting	Vector of membership probabilities (PF, PD, death) at time zero.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pf_stm(0:100, params)
```

prob_pps_cf	<i>Calculate probability of post progression survival under the state transition clock forward model</i>
-------------	--

Description

Calculates probability of post progression survival at a given time from progression (vectorized). This probability is from the state transition clock forward model, according to the given statistical distributions and parameters.

Usage

```
prob_pps_cf(ttptimes, ppstimes, dpam)
```

Arguments

ttptimes	Time (numeric and vectorized) from progression - not time from baseline.
ppstimes	Time (numeric and vectorized) of progression
dpam	List of survival regressions for model endpoints. This must include post progression survival calculated under the clock forward state transition model.

Value

Vector of the mean probabilities of post-progression survival at each PPS time, averaged over TTP times.

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pps_cf(0:100, 0:100, params)
```

prob_pps_cr	<i>Calculate probability of post progression survival under the state transition clock reset model</i>
-------------	--

Description

Calculates probability of post progression survival at a given time from progression (vectorized). This probability is from the state transition clock reset model, according to the given statistical distributions and parameters.

Usage

```
prob_pps_cr(time, dpam)
```

Arguments

time	Time (numeric and vectorized) from baseline - not time from progression.
dpam	List of survival regressions for model endpoints. This must include post progression survival calculated under the clock reset state transition model.

Value

Numeric value

Examples

```
bosonc <- create_dummydata("flexbosms")
fits <- fit_ends_mods_spl(bosonc)
# Pick out best distribution according to min AIC
params <- list(
  ppd = find_bestfit(fits$ppd, "aic")$fit,
  ttp = find_bestfit(fits$ttp, "aic")$fit,
  pfs = find_bestfit(fits$pfs, "aic")$fit,
  os = find_bestfit(fits$os, "aic")$fit,
  pps_cf = find_bestfit(fits$pps_cf, "aic")$fit,
  pps_cr = find_bestfit(fits$pps_cr, "aic")$fit
)
prob_pps_cr(0:100, params)
```

vlookup

*VLOOKUP function***Description**

Function to lookup values according to an index. Aims to behave similarly to VLOOKUP in Microsoft Excel, however several lookups can be made at once (indexval can be a vector) and interpolation is available where lookups are inexact (choice of 4 methods).

Usage

```
vlookup(indexval, indexvec, valvec, method = "geom")
```

Arguments

indexval	The index value to be looked-up (may be a vector of multiple values)
indexvec	The vector of indices to look-up within
valvec	The vector of values corresponding to the vector of indices
method	Method may be floor, ceiling, arith or geom (default).

Value

Numeric value or vector, depending on the lookup/interpolation method chosen:

- floor: Floor (minimum) value, where interpolation is required between measured values
- ceiling: Ceiling (maximum) value, where interpolation is required between measured values
- arith: Arithmetic mean, where interpolation is required between measured values
- geom: Geometric mean, where interpolation is required between measured values

See Also

HMDHFDplus::readHMDweb() can be used to obtain lifetables from the Human Mortality Database

Examples

```
# Suppose we have survival probabilities at times 0 to 20
times <- 0:20
survival <- 1-times*0.04
# We would like to look-up the survival probability at time 7
vlookup(7, times, survival)
# In this case, the floor, ceiling, arith and geom values are identical
# because survival time 7 is known, and no interpolation is necessary
vlookup(c(7, 7.5), times, survival)
# The second row of the returned tibble reveal different estimates of the survival at time 7.5.
# The values vary according to the interpolation method between
# observed survival values at times 7 and 8.
```

Index

calc_allrmds, [2](#)
calc_haz_psm, [4](#)
calc_likes, [5](#)
calc_rmd, [7](#)
calc_surv_psmpps, [8](#)
check_consistent_pfs, [9](#)
compare_psm_likes, [10](#)
constrain_survprob, [11](#)
create_dummydata, [12](#)
create_extrafields, [13](#)

find_bestfit, [13](#)
fit_ends_mods_par, [14](#)
fit_ends_mods_par(), [16](#)
fit_ends_mods_spl, [16](#)
fit_ends_mods_spl(), [15](#)
flexsurv::bosms3(), [12](#)
flexsurv::flexsurvreg, [7](#), [8](#)
flexsurv::flexsurvreg(), [15](#)
flexsurv::flexsurvspline, [8](#), [16](#)
flexsurv::flexsurvspline(), [16](#)

graph_psm_hazards, [17](#)
graph_psm_survs, [18](#)
graph_survs, [19](#)

pharmaverseadam::adrs_onco, [12](#)
pharmaverseadam::adsl, [12](#)
prob_os_psm, [21](#)
prob_os_stm_cf, [22](#)
prob_os_stm_cr, [23](#)
prob_pd_psm, [24](#)
prob_pd_stm_cf, [25](#)
prob_pd_stm_cr, [26](#)
prob_pf_psm, [27](#)
prob_pf_stm, [28](#)
prob_pps_cf, [29](#)
prob_pps_cr, [30](#)
psm3mkv, [12](#)

survival::cancer(), [12](#)

vlookup, [31](#)