Package 'RMSTpowerBoost'

April 28, 2026

Type Package

Title Power and Sample Size for Restricted Mean Survival Time Based Clinical Trials

Version 1.0.3

Description Tools for Restricted Mean Survival Time based study design and analysis planning. Provides power and sample size calculations for two-arm studies using direct modeling approaches from the literature, including semiparametric additive models, linear Inverse Probability Weighting based models from Wei (2014) <doi:10.1093/biostatistics/kxt050>, multiplicative stratified models from Wang (2019) <doi:10.1002/sim.8356>, and covariate-dependent censoring methods from Wang (2018) <doi:10.1007/s10985-017-9391-6>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports survival, mgcv, ggplot2, stats, utils, dplyr, magrittr, future, future.apply, knitr

Suggests roxygen2, shiny, shinyjs, bslib, DT, plotly, kableExtra, survminer, rmarkdown, mice, tidyr, purrr, tibble, htmltools, tinytex, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 4.1.0)

URL https://uthsc-zhang.github.io/RMSTpowerBoost-Package/,

BugReports https://github.com/UTHSC-Zhang/RMSTpowerBoost-Package/issues

NeedsCompilation no

Author Arnab Aich [aut, cre] (ORCID: <https://orcid.org/0009-0005-7801-6701>), Yuan Zhang [aut] (ORCID: <https://orcid.org/0000-0001-7626-9514>)

Maintainer Arnab Aich <aaich@fsu.edu>

Repository CRAN

Date/Publication 2026-04-28 19:50:02 UTC

Contents

`additive.power.analytical` *Analyze Power for a Stratified Additive RMST Model (Analytic)*

Description

Performs power analysis for a stratified, additive RMST model using the analytic variance estimator based on the method of Zhang & Schaubel (2024).

Usage

```
additive.power.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var,
  sample_sizes,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

`pilot_data`	A `data.frame` containing pilot study data.
`time_var`	A character string for the time-to-event variable.
`status_var`	A character string for the event status variable (1=event, 0=censored).
`arm_var`	A character string for the treatment arm variable (1=treatment, 0=control).
`strata_var`	A character string for the stratification variable.
`sample_sizes`	A numeric vector of sample sizes *per stratum* to calculate power for.
`linear_terms`	An optional character vector of other covariate names.
`L`	The numeric value for the RMST truncation time.
`alpha`	The significance level (Type I error rate).
`verbose`	Logical; if `TRUE`, emit progress messages. Default `FALSE`.

Details

This function computes power for the semiparametric additive RMST model $\mu_{ij} = \mu_{0j} + \beta' Z_i$, where `i` indexes subjects and `j` indexes strata.

The method uses Inverse Probability of Censoring Weighting (IPCW), with weights derived from a stratified Cox model for the censoring times. The regression coefficient $\hat{\beta}$ is estimated by centering

the covariates and RMST values within each stratum and then solving the resulting estimating equations in closed form.

Power is obtained from the asymptotic sandwich variance of $\hat{\beta}$. This implementation uses the robust variance estimator $A_n^{-1}B_n(A_n^{-1})'$, where A_n and B_n are empirical estimates of the variance components.

Value

A `list` containing:

`results_data`	A `data.frame` with specified sample sizes and corresponding powers.
`results_plot`	A `ggplot` object visualizing the power curve.

Examples

```
set.seed(123)
pilot_df_strat <- data.frame(
 time = rexp(150, 0.1),
 status = rbinom(150, 1, 0.8),
 arm = rep(0:1, each = 75),
 region = factor(rep(c("A", "B", "C"), each = 50)),
 age = rnorm(150, 60, 10)
)
# Introduce an additive treatment effect
pilot_df_strat$time[pilot_df_strat$arm == 1] <-
  pilot_df_strat$time[pilot_df_strat$arm == 1] + 1.5

power_results <- additive.power.analytical(
  pilot_data = pilot_df_strat,
  time_var = "time", status_var = "status", arm_var = "arm", strata_var = "region",
  sample_sizes = c(100, 150, 200),
  linear_terms = "age",
  L = 12
)
print(power_results$results_data)
print(power_results$results_plot)
```

---

additive.ss.analytical

Find Sample Size for a Stratified Additive RMST Model (Analytic)

Description

Calculates the required sample size for a target power using the analytic method for a stratified, additive RMST model.

Usage

```
additive.ss.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var,
  target_power,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame containing pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable (1=event, 0=censored).
arm_var	A character string for the treatment arm variable (1=treatment, 0=control).
strata_var	A character string for the stratification variable.
target_power	A single numeric value for the desired power.
linear_terms	An optional character vector of other covariate names.
L	The numeric value for the RMST truncation time.
alpha	The significance level (Type I error rate).
n_start	The starting sample size *per stratum* for the search.
n_step	The increment in sample size at each step of the search.
max_n_per_arm	The maximum sample size *per stratum* to search up to.
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function estimates the additive treatment effect and its asymptotic variance once from the pilot data, then increases the per-stratum sample size until the target power is reached or the search limit is hit. It uses the same stratum-centering framework as additive.power.analytical.

Value

A list containing:

results_data	A data.frame with the target power and required sample size.
results_plot	A ggplot object visualizing the search path.
results_summary	A data.frame summarizing the estimated treatment effect.

Examples

```
set.seed(123)
pilot_df_strat <- data.frame(
 time = rexp(150, 0.1),
 status = rbinom(150, 1, 0.8),
 arm = rep(0:1, each = 75),
 region = factor(rep(c("A", "B", "C"), each = 50)),
 age = rnorm(150, 60, 10)
)
# Introduce an additive treatment effect
pilot_df_strat$time[pilot_df_strat$arm == 1] <-
  pilot_df_strat$time[pilot_df_strat$arm == 1] + 1.5

  # Find the required sample size per stratum for 80% power
  ss_results <- additive.ss.analytical(
    pilot_data = pilot_df_strat,
    time_var = "time", status_var = "status",
    arm_var = "arm", strata_var = "region",
    target_power = 0.50,
    L = 18, #
    n_start = 200,
    n_step = 50,
    max_n_per_arm = 1000
  )
  print(ss_results$results_data)
  print(ss_results$results_plot)
```

---

aft_lognormal_L12_n150

Simulated dataset: AFT log-normal, L = 12, n = 150

Description

Columns:

- time: observed time
- status: event indicator (1 = event, 0 = censored)
- arm: treatment arm (0 = control, 1 = treatment)
- age: baseline age
- gender: factor with two levels

Format

A data frame with 150 rows and 5 variables.

Examples

```
data(aft_lognormal_L12_n150)
table(aft_lognormal_L12_n150$arm)
```

aft_weibull_L24_n200 *Simulated dataset: AFT Weibull, L = 24, n = 200*

Description

Columns: time, status, arm, age, gender.

Format

A data frame with 200 rows and 5 variables.

Examples

```
data(aft_weibull_L24_n200)
mean(aft_weibull_L24_n200$status == 0)
```

covar_binary *Binary covariate definition*

Description

Builds a binary (Bernoulli) covariate definition list.

Usage

```
covar_binary(name, p = 0.5)
```

Arguments

name	Column name in the generated data.
p	Probability of value 1. Default 0.5.

Value

A named list suitable as one element of the covariates argument.

See Also

covar_continuous, covar_categorical, rmst.sim

Examples

```
covar_binary("female", p = 0.52)
```

covar_categorical *Categorical covariate definition*

Description

Builds a multi-level categorical covariate definition list.

Usage

```
covar_categorical(name, probs, labels = NULL)
```

Arguments

name	Column name in the generated data.
probs	Numeric vector of category probabilities (must sum to 1).
labels	Optional character vector of level labels. Length must equal length(probs). Defaults to "1", "2", ...

Value

A named list suitable as one element of the covariates argument.

See Also

covar_continuous, covar_binary, rmst.sim

Examples

```
covar_categorical("stage", probs = c(0.4, 0.35, 0.25),
                  labels = c("I", "II", "III"))
```

covar_continuous *Continuous covariate definition*

Description

Builds a covariate definition list for use in rmst.sim, recipe_quick_aft, or recipe_quick_ph.

Usage

```
covar_continuous(name, dist = "normal", ...)
```

Arguments

name	Column name in the generated data.
dist	Distribution name. One of "normal" (default), "lognormal", "gamma", "weibull", "uniform", "beta", "t".
...	Named distribution parameters passed as the params list (e.g., mean = 0, sd = 1 for dist = "normal").

Value

A named list suitable as one element of the covariates argument.

See Also

covar_binary, covar_categorical, rmst.sim

Examples

```
covar_continuous("age", dist = "normal", mean = 50, sd = 10)
covar_continuous("bmi", dist = "lognormal", meanlog = 3.2, sdlog = 0.2)
```

DC.power.analytical *Analyze Power for RMST Model with Covariate-Dependent Censoring (Analytic)*

Description

Performs power analysis for an RMST model when the censoring mechanism depends on observed covariates.

Usage

```
DC.power.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  sample_sizes,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame with pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable (1=event, 0=censored).
arm_var	A character string for the treatment arm variable (1=treatment, 0=control).
sample_sizes	A numeric vector of sample sizes *per arm* to calculate power for.
linear_terms	Optional character vector of additional covariate names.
L	The numeric value for the RMST truncation time.
alpha	The significance level (Type I error rate).
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function assumes a single censoring process whose hazard depends on covariates (but not on treatment by default). It fits a Cox model for censoring

$$\Pr(\text{censoring by } t \mid X) = 1 - G(t \mid X)$$

using Surv(time, status==0) ~ linear_terms, then forms inverse-probability-of-censoring weights (IPCW) $w_i = 1/\hat{G}(Y_i \mid X_i)$ evaluated at $Y_i = \min(T_i, L)$. The RMST regression $E[Y_i \mid A_i, X_i]$ is then fit by weighted least squares, and power is derived from a sandwich variance that ignores uncertainty from estimating $\hat{G}$.

Note: This implementation models a single censoring process and does not handle competing risks.

Value

A list with:

results_data	A data.frame with N_per_Arm and Power.
results_plot	A ggplot object of the power curve.
results_summary	A data.frame summarizing the pilot arm effect.

DC.ss.analytical *Find Sample Size for RMST with Covariate-Dependent Censoring (Analytic)*

Description

Iteratively finds required per-arm sample size for a target power, using the same IPCW-based analytic variance as DC.power.analytical.

Usage

```
DC.ss.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  target_power,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame containing pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable (1=event, 0=censored).
arm_var	A character string for the treatment arm variable (1=treatment, 0=control).
target_power	A single numeric value for the desired power.
linear_terms	Optional character vector of additional covariate names.
L	The numeric value for the RMST truncation time.
alpha	The significance level (Type I error rate).
n_start	The starting sample size *per arm* for the search.
n_step	The increment in sample size at each step of the search.
max_n_per_arm	The maximum sample size *per arm* to search.
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

Uses a single censoring Cox model Surv(time, status==0) ~ linear_terms to form IPCW and fits a weighted RMST regression. Treatment is excluded from the censoring model by default. Competing risks are not modeled. Variance ignores uncertainty in $\hat{G}$.

Note: This implementation models a single censoring process and does not handle competing risks.

Value

A list with:

results_data	data.frame with Target_Power and Required_N_per_Arm.
results_plot	ggplot showing the search path.
results_summary	data.frame summarizing the pilot arm effect.

describe_generation *Summarize a generated dataset and its simulation mechanism*

Description

Given one element returned by load_recipe_sets() (a list with data and a rich meta list), this builds tidy tables describing the data-generating process: model family, baseline parameters, linear predictor coefficients (intercept / treatment / covariates), treatment assignment, and censoring.

Usage

```
describe_generation(set)
```

Arguments

set	A single element from load_recipe_sets(), i.e. list(data=<df>, meta=<list>).

Value

A list of data frames:

- header: n, L (analysis horizon; stored as attr "tau"), model, event_rate, achieved_censoring.
- baseline: flattened baseline parameters.
- effects: coefficients for intercept/treatment and covariates (and formula terms if used).
- treatment: assignment mechanism and knobs.
- censoring: censoring mode/target/admin time.
- covariates: each generated covariate with its distribution and parameters.
- files: paths to on-disk files (csv/rds/rdata) when available.

Examples

```
covs <- list(covar_continuous("x", mean = 0, sd = 1))
rec <- recipe_quick_aft(
  n = 20, model = "aft_lognormal",
  baseline = list(mu = 2.7, sigma = 0.6),
  treat_effect = -0.2, covariates = covs,
  target_censoring = 0.2, seed = 123
)
out <- file.path(tempdir(), "rmst_describe_generation")
generate_recipe_sets(rec, out_dir = out, formats = "rds", n_reps = 1)
sets <- load_recipe_sets(file.path(out, "manifest.rds"))
spec <- describe_generation(sets[[1]])
spec$header
spec$baseline
spec$effects
spec$treatment
spec$censoring
spec$covariates
spec$files
```

GAM.power.boot *Calculate Power for a Semiparametric Additive RMST Model via Simulation*

Description

Performs a power analysis for given sample sizes using a flexible, semiparametric additive model for the RMST based on pseudo-observations.

Usage

```
GAM.power.boot(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var = NULL,
  sample_sizes,
  linear_terms = NULL,
  smooth_terms = NULL,
  L,
  n_sim = 1000,
  alpha = 0.05,
  parallel.cores = 1,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame with pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable.
arm_var	A character string for the treatment arm variable.
strata_var	An optional character string for a stratification variable.
sample_sizes	A numeric vector of sample sizes per arm/stratum.
linear_terms	Optional character vector of covariates with a linear effect.
smooth_terms	Optional character vector of covariates with a non-linear effect.
L	The numeric truncation time for RMST.
n_sim	Number of bootstrap simulations.
alpha	The significance level.
parallel.cores	Number of cores for parallel processing.
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function estimates power by bootstrap resampling the pilot data, computing RMST pseudo-observations, fitting a semiparametric additive model, and recording whether the treatment effect is significant. If `strata_var` is supplied, resampling is carried out within strata. The model formula can include linear terms, non-linear smooth terms (`s()`), and interactions.

Power is estimated as the proportion of simulations with a treatment-effect p-value below `alpha`. This pseudo-observation approach models RMST directly without using the IPCW estimating equations used elsewhere in the package.

Value

A `list` containing:

`results_data` A `data.frame` of sample sizes and corresponding estimated power.

`results_plot` A `ggplot` object visualizing the power curve.

`results_summary` A `data.frame` with a summary of the estimated treatment effect.

Note

`status_var` should be `1` for an event, `0` for censored. `arm_var` should be `1` for treatment, `0` for control.

Examples

```
pilot_df <- data.frame(
  time = rexp(100, 0.08),
  status = rbinom(100, 1, 0.7),
  arm = rep(0:1, each = 50),
  age = rnorm(100, 60, 10)
)
# Add a treatment effect
pilot_df$time[pilot_df$arm == 1] <- pilot_df$time[pilot_df$arm == 1] * 1.3

power_results <- GAM.power.boot(
  pilot_data = pilot_df,
  time_var = "time",
  status_var = "status",
  arm_var = "arm",
  sample_sizes = c(100, 150),
  linear_terms = "age",
  L = 15,
  n_sim = 100, # Use more sims in practice, e.g., 1000
  parallel.cores = 2
)
print(power_results$results_data)
print(power_results$results_plot)
```

GAM.ss.boot *Find Sample Size for a Semiparametric Additive RMST Model via Simulation*

Description

Performs an iterative sample size search to achieve a target power using a flexible, semiparametric additive model for the RMST.

Usage

```
GAM.ss.boot(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var = NULL,
  target_power,
  linear_terms = NULL,
  smooth_terms = NULL,
  L,
  n_sim = 1000,
  alpha = 0.05,
  parallel.cores = 1,
  patience = 5,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame with pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable.
arm_var	A character string for the treatment arm variable.
strata_var	An optional string for a stratification variable.
target_power	A single numeric value for the target power.
linear_terms	Optional character vector of covariates with a linear effect.
smooth_terms	Optional character vector of covariates with a non-linear effect.
L	The numeric truncation time for RMST.
n_sim	Number of bootstrap simulations per search step.
alpha	The significance level.

parallel.cores	Number of cores for parallel processing.
patience	Number of consecutive non-improving steps in the search before terminating.
n_start	The starting sample size per arm/stratum for the search.
n_step	The increment in sample size at each step of the search.
max_n_per_arm	The maximum sample size per arm/stratum to search up to.
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function increases the sample size step by step until the estimated power reaches target_power or a stopping rule is met. At each step it runs the same bootstrap procedure described in GAM.power.boot.

Value

A list containing:

results_data	A data.frame with the target power and required sample size.
results_plot	A ggplot object of the search path.
results_summary	A data.frame summarizing the estimated treatment effect.

Note

This function's methodology is bootstrap-based.

Examples

```
pilot_df_effect <- data.frame(
  time = c(stats::rexp(50, 0.1), stats::rexp(50, 0.04)), # Effect
  status = stats::rbinom(100, 1, 0.9),
  arm = rep(0:1, each = 50)
)

ss_results <- GAM.ss.boot(
  pilot_data = pilot_df_effect,
  time_var = "time",
  status_var = "status",
  arm_var = "arm",
  target_power = 0.80,
  L = 15,
  n_sim = 100,      # Low n_sim for example
  n_start = 100,
  n_step = 50,
  patience = 2,
  parallel.cores = 2
)
print(ss_results$results_data)
print(ss_results$results_plot)
```

`generate_recipe_sets` *Generate simulated datasets across scenario combinations (TXT/CSV/RDS/RData)*

Description

Builds a grid of scenarios from a base recipe and a named list of variations, simulates one or more replicates per scenario, and writes the datasets to files in a target folder as `.txt` (tab), `.csv`, `.rds`, and/or `.RData`. A manifest is written as `manifest.rds`. The recipe specification does not include an analysis horizon; specify `L` later in downstream analysis functions.

Usage

```
generate_recipe_sets(
  base_recipe,
  vary = list(),
  out_dir,
  formats = c("txt", "csv", "rds", "rdata"),
  n_reps = 1L,
  seed_base = NULL,
  filename_template = "sc{scenario_id}_r{rep}"
)
```

Arguments

`base_recipe`	A recipe list (use `validate_recipe()` if needed).
`vary`	Named list; keys are dotted paths inside the recipe (e.g., `"n"`, `"censoring.target"`, `"event_time.effects.treatment"`).
`out_dir`	Directory to write datasets and `manifest.rds` (created if missing).
`formats`	Character vector subset of `c("txt","csv","rds","rdata")`.
`n_reps`	Integer; number of replicates per scenario.
`seed_base`	Optional integer retained in the manifest as per-replicate provenance metadata computed as `seed_base + scenario_id*1000 + rep`. It does not set the random-number generator.
`filename_template`	Base filename (no extension) with placeholders: `"{scenario_id}"`, `"{rep}"` and any dotted path used in `vary`.

Value

Invisibly returns the manifest `data.frame` and writes `manifest.rds`.

Examples

```
covs <- list(list(name="x", type="continuous", dist="normal", params=list(mean=0, sd=1)))
rec <- recipe_quick_aft(60, "aft_lognormal",
         baseline=list(mu=2.7, sigma=0.6), treat_effect=-0.2,
         covariates=covs, target_censoring=0.2)
out <- file.path(tempdir(), "rmst_checks")
dir.create(out, showWarnings = FALSE, recursive = TRUE)
man <- generate_recipe_sets(rec, vary=list(n=c(60,80)), out_dir=out,
         formats=c("csv","rds"), n_reps=1, seed_base=123)
```

---

gen_covariates *Generate covariate matrix/data frame from a recipe*

Description

Generate covariate matrix/data frame from a recipe

Usage

```
gen_covariates(n, covariates)
```

Arguments

n	sample size
covariates	list(defs = list(...))

Value

data.frame of covariates

Examples

```
defs <- list(
  list(name="x", type="continuous", dist="normal", params=list(mean=0, sd=1)),
  list(name="z", type="categorical", dist="categorical",
       params=list(prob=c(0.3,0.7), labels=c("A","B")))
)
X <- gen_covariates(10, list(defs = defs))
```

`linear.power.analytical` *Analyze Power for a Linear RMST Model (Analytic)*

Description

Performs power analysis using a direct formula based on the asymptotic variance estimator for the linear RMST model.

Usage

```
linear.power.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  sample_sizes,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

`pilot_data`	A `data.frame` containing pilot study data.
`time_var`	A character string specifying the name of the time-to-event variable.
`status_var`	A character string specifying the name of the event status variable (1=event, 0=censored).
`arm_var`	A character string specifying the name of the treatment arm variable (1=treatment, 0=control).
`sample_sizes`	A numeric vector of sample sizes *per arm* to calculate power for.
`linear_terms`	An optional character vector of other covariate names to include in the model.
`L`	The numeric value for the RMST truncation time.
`alpha`	The significance level for the power calculation (Type I error rate).
`verbose`	Logical; if `TRUE`, emit progress messages. Default `FALSE`.

Details

This function implements the analytic power calculation for the direct linear regression model of the Restricted Mean Survival Time (RMST) proposed by Tian et al. (2014). The core of the method is a weighted linear model of the form

$$E[Y_i \mid A_i, \mathbf{Z}_i] = \alpha + \tau A_i + \mathbf{Z}_i^\top \boldsymbol{\gamma}$$

where $Y_i = \min(T_i, L)$ is the event time truncated at L, A_i is the treatment indicator, and τ is the treatment effect of interest.

To handle right-censoring, the method uses Inverse Probability of Censoring Weighting (IPCW). The weight for an uncensored individual `i` is the inverse of the probability of remaining uncensored until their event time, $w_i = \delta_i / \hat{G}(Y_i)$, where $\hat{G}(t) = P(C > t)$ is the Kaplan-Meier estimate of the censoring distribution.

Power is calculated analytically based on the asymptotic properties of the coefficient estimators. The variance of the treatment effect estimator, $\hat{\tau}$, is derived from a robust sandwich variance estimator of the form $A^{-1}B(A^{-1})'$. In this implementation, `A` is the scaled information matrix $(X'WX)/n$, and `B` is the empirical second moment of the influence functions, $(\sum \epsilon_i \epsilon_i')/n$, where ϵ_i is the influence curve for observation `i`. The resulting variance is used to calculate the standard error for a given sample size, which in turn is used in the power formula.

Value

A `list` containing:

`results_data` A `data.frame` with the specified sample sizes and their corresponding calculated power.

`results_plot` A `ggplot` object visualizing the power curve.

`results_summary` A `data.frame` summarizing the treatment effect from the pilot data used for the calculation.

Examples

```
pilot_df <- data.frame(
  time = rexp(100, 0.1),
  status = rbinom(100, 1, 0.7),
  arm = rep(0:1, each = 50),
  age = rnorm(100, 55, 10)
)
power_results <- linear.power.analytical(
  pilot_data = pilot_df,
  time_var = "time",
  status_var = "status",
  arm_var = "arm",
  linear_terms = "age",
  sample_sizes = c(100, 200, 300),
  L = 10
)
print(power_results$results_data)
print(power_results$results_plot)
```

`linear.power.boot` *Analyze Power for a Linear RMST Model via Simulation*

Description

Performs a power analysis for given sample sizes based on the direct linear regression model for RMST, using a bootstrap simulation approach.

Usage

```
linear.power.boot(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  sample_sizes,
  linear_terms = NULL,
  L,
  n_sim = 1000,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

`pilot_data`	A `data.frame` with pilot study data.
`time_var`	A character string for the time-to-event variable.
`status_var`	A character string for the event status variable.
`arm_var`	A character string for the treatment arm variable.
`sample_sizes`	A numeric vector of sample sizes *per arm* to calculate power for.
`linear_terms`	Optional character vector of other covariates for the linear model.
`L`	The numeric truncation time for RMST.
`n_sim`	The number of bootstrap simulations to run for each sample size.
`alpha`	The significance level (Type I error rate).
`verbose`	Logical; if `TRUE`, emit progress messages. Default `FALSE`.

Details

This function estimates power by generating a number of bootstrap samples (`n_sim`) from the provided pilot data by resampling with replacement. For each bootstrap sample, it performs the following steps:

1. Estimates the censoring distribution using the Kaplan-Meier method (`survival::survfit`).
2. Calculates Inverse Probability of Censoring Weights (IPCW) for each observation.

3. Fits a weighted linear model (`stats::lm`) to the RMST of the uncensored subjects.
4. Extracts the p-value for the treatment `arm_var` coefficient.

The final power for a given sample size is the proportion of the `n_sim` simulations where this p-value is less than the significance level `alpha`. This simulation-based approach can be useful when analytic approximations are less reliable, but it can be computationally intensive.

Value

A `list` containing:

`results_data` A `data.frame` of sample sizes and corresponding estimated power.

`results_plot` A `ggplot` object visualizing the power curve.

`results_summary` A `data.frame` with summary statistics for the estimated treatment effect from the largest sample size simulation.

Note

`status_var` should be `1` for an event, `0` for censored. `arm_var` should be `1` for treatment, `0` for control.

Examples

```
pilot_df <- data.frame(
  time = rexp(100, 0.1),
  status = rbinom(100, 1, 0.7),
  arm = rep(0:1, each = 50),
  age = rnorm(100, 60, 8)
)
# Introduce a treatment effect for a more interesting example
pilot_df$time[pilot_df$arm == 1] <- pilot_df$time[pilot_df$arm == 1] * 1.5

power_results <- linear.power.boot(
  pilot_data = pilot_df,
  time_var = "time",
  status_var = "status",
  arm_var = "arm",
  linear_terms = "age",
  sample_sizes = c(100, 150, 200),
  L = 10,
  n_sim = 200 # Use more simulations in practice (e.g., 1000)
)
print(power_results$results_data)
print(power_results$results_plot)
```

`linear.ss.analytical` *Find Sample Size for a Linear RMST Model (Analytic)*

Description

Calculates the required sample size for a target power using an analytic formula based on the methods of Tian et al. (2014).

Usage

```
linear.ss.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  target_power,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

`pilot_data`	A `data.frame` containing pilot study data.
`time_var`	A character string specifying the name of the time-to-event variable.
`status_var`	A character string specifying the name of the event status variable (1=event, 0=censored).
`arm_var`	A character string specifying the name of the treatment arm variable (1=treatment, 0=control).
`target_power`	A single numeric value for the desired power (e.g., 0.80 or 0.90).
`linear_terms`	An optional character vector of other covariate names to include in the model.
`L`	The numeric value for the RMST truncation time.
`alpha`	The significance level (Type I error rate).
`n_start`	The starting sample size *per arm* for the search.
`n_step`	The increment in sample size at each step of the search.
`max_n_per_arm`	The maximum sample size *per arm* to search up to.
`verbose`	Logical; if `TRUE`, emit progress messages. Default `FALSE`.

Details

This function performs an iterative search to find the sample size needed to achieve a specified `target_power`. It uses the same underlying theory as `linear.power.analytical`. First, it estimates the treatment effect size and its asymptotic variance from the pilot data. Then, it iteratively calculates the power for increasing sample sizes using the analytic formula until the target power is achieved.

Value

A `list` containing:

`results_data` A `data.frame` with the target power and the required sample size per arm.

`results_plot` A `ggplot` object visualizing the sample size search path.

`results_summary` A `data.frame` summarizing the treatment effect from the pilot data used for the calculation.

Examples

```
pilot_df <- data.frame(
  time = c(rexp(50, 0.1), rexp(50, 0.07)), # Introduce an effect
  status = rbinom(100, 1, 0.8),
  arm = rep(0:1, each = 50),
  age = rnorm(100, 55, 10)
)
ss_results <- linear.ss.analytical(
  pilot_data = pilot_df,
  time_var = "time",
  status_var = "status",
  arm_var = "arm",
  target_power = 0.80,
  L = 10
)
print(ss_results$results_data)
print(ss_results$results_plot)
```

---

linear.ss.boot *Find Sample Size for a Linear RMST Model via Simulation*

Description

Performs an iterative sample size search to achieve a target power based on the direct linear regression model for RMST, using bootstrap simulation.

Usage

```
linear.ss.boot(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  target_power,
  linear_terms = NULL,
  L,
  n_sim = 1000,
  alpha = 0.05,
  patience = 5,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame with pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable.
arm_var	A character string for the treatment arm variable.
target_power	A single numeric value for the target power (e.g., 0.80).
linear_terms	Optional character vector of other covariates for the linear model.
L	The numeric truncation time for RMST.
n_sim	The number of bootstrap simulations per search step.
alpha	The significance level.
patience	The number of consecutive non-improving steps in the search before terminating.
n_start	The starting sample size *per arm* for the search.
n_step	The increment in sample size at each step of the search.
max_n_per_arm	The maximum sample size *per arm* to search up to.
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function iteratively searches for the required sample size to achieve a specified target_power. At each step of the search, it runs a full bootstrap simulation (n_sim iterations), as described in linear.power.boot, to estimate the power for the current sample size. The search stops when the target power is achieved or other stopping criteria (e.g., patience) are met. Due to the nested simulation structure, this function can be very time-consuming.

Value

A list containing:

- `results_data` A data.frame with the target power and the final required sample size per arm.
- `results_plot` A ggplot object showing the search path.
- `results_summary` A data.frame with summary statistics for the estimated treatment effect from the final simulation.

Note

status_var should be 1 for an event, 0 for censored. arm_var should be 1 for treatment, 0 for control.

Examples

```
pilot_df_effect <- data.frame(
  time = c(rexp(50, 0.1), rexp(50, 0.05)), # Effect present
  status = rbinom(100, 1, 0.8),
  arm = rep(0:1, each = 50)
)
ss_results <- linear.ss.boot(
  pilot_data = pilot_df_effect,
  time_var = "time",
  status_var = "status",
  arm_var = "arm",
  target_power = 0.80,
  L = 10,
  n_sim = 200, # Low n_sim for example
  patience = 2,
  n_start = 100,
  n_step = 50,
  max_n_per_arm = 500
)
print(ss_results$results_data)
print(ss_results$results_plot)
```

---

load_recipe_sets *Load datasets from a recipe-sets manifest*

Description

Reads a manifest.rds created by generate_recipe_sets(), loads one dataset per row (preferring rds to rdata to csv to txt), restores attribute "achieved_censoring", and returns a named list of list(data = <data.frame>, meta = <list>).

Usage

```
load_recipe_sets(manifest_path)
```

Arguments

`manifest_path` Path to `manifest.rds`.

Value

A named list where each element is `list(data=..., meta=...)`.

Examples

```
covs <- list(covar_continuous("x", mean = 0, sd = 1))
rec <- recipe_quick_aft(
  n = 20, model = "aft_lognormal",
  baseline = list(mu = 2.7, sigma = 0.6),
  treat_effect = -0.2, covariates = covs,
  target_censoring = 0.2, seed = 123
)
out <- file.path(tempdir(), "rmst_load_recipe_sets")
generate_recipe_sets(rec, out_dir = out, formats = "rds", n_reps = 1)
sets <- load_recipe_sets(file.path(out, "manifest.rds"))
names(sets)
str(sets[[1]]$meta)
```

`MS.power.analytical` *Analyze Power for a Multiplicative Stratified RMST Model (Analytic)*

Description

Performs power analysis for a multiplicative, stratified RMST model using an analytic method based on the work of Wang et al. (2019).

Usage

```
MS.power.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var,
  sample_sizes,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame with pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable (1=event, 0=censored).
arm_var	A character string for the treatment arm variable (1=treatment, 0=control).
strata_var	A character string for the stratification variable.
sample_sizes	A numeric vector of sample sizes *per stratum* to calculate power for.
linear_terms	An optional character vector of other covariate names.
L	The numeric value for the RMST truncation time.
alpha	The significance level (Type I error rate).
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function is based on the method for evaluating center (stratum) effects using a multiplicative model for RMST: $\mu_{ij} = \mu_{0j} \exp\{\beta' Z_i\}$. The method uses IPCW with a stratified Cox model for the censoring distribution.

Formal estimation of β requires an iterative solver for the estimating equation given in Equation (8) of Wang et al. (2019). Because this is computationally intensive, this implementation uses a log-linear working model fitted by weighted least squares to pseudo-observations (`lm(log(Y_rmst) ~ ...)`) as a tractable approximation. The approximation is consistent when the log-linear mean structure is well-specified but may differ from the formal estimator under strong misspecification.

The power calculation relies on the asymptotic variance of the log-RMST ratio estimator, $\hat{\tau}$. The variance is derived from the robust variance-covariance matrix of the `lm` fit, which serves as a proxy for the formal sandwich estimator $A^{-1}B(A^{-1})'$ described in Theorem 1 of Wang et al. (2019).

Value

A `list` containing:

results_data	A data.frame with sample sizes and corresponding powers.
results_plot	A ggplot object visualizing the power curve.

Examples

```
set.seed(123)
pilot_df_strat <- data.frame(
 time = rexp(120, 0.15),
 status = rbinom(120, 1, 0.6),
 arm = rep(0:1, each = 60),
 region = factor(rep(c("A", "B", "C"), each = 40))
)
pilot_df_strat$time[pilot_df_strat$arm == 1] <- pilot_df_strat$time[pilot_df_strat$arm == 1] * 1.5

power_results <- MS.power.analytical(
 pilot_data = pilot_df_strat,
```

```
 time_var = "time", status_var = "status", arm_var = "arm", strata_var = "region",
 sample_sizes = c(50, 75, 100),
 L = 10, alpha = 0.05
)
print(power_results$results_data)
```

`MS.power.boot` *Analyze Power for a Multiplicative Stratified RMST Model via Simulation*

Description

Performs power analysis based on a multiplicative model for RMST for stratified trials, using a bootstrap simulation approach.

Usage

```
MS.power.boot(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var,
  sample_sizes,
  linear_terms = NULL,
  L,
  n_sim = 1000,
  alpha = 0.05,
  parallel.cores = 1,
  verbose = FALSE
)
```

Arguments

`pilot_data`	A `data.frame` with pilot study data.
`time_var`	A character string for the time-to-event variable.
`status_var`	A character string for the event status variable.
`arm_var`	A character string for the treatment arm variable.
`strata_var`	A character string for the stratification variable.
`sample_sizes`	A numeric vector of sample sizes *per stratum* to calculate power for.
`linear_terms`	Optional character vector of covariates for the model.
`L`	The numeric truncation time for RMST.
`n_sim`	Number of bootstrap simulations.
`alpha`	The significance level.
`parallel.cores`	Number of cores for parallel processing.
`verbose`	Logical; if `TRUE`, emit progress messages. Default `FALSE`.

Details

This function estimates power through bootstrap simulation, resampling within each stratum defined by `strata_var`. In each of the `n_sim` iterations:

1. Jackknife pseudo-observations for the RMST are calculated.
2. A log-linear model (`stats::lm`) is fitted to the `log(pseudo_obs)`. This models the multiplicative relationship on the original RMST scale, i.e., $\mu_{ij} = \mu_{0j} \exp\{\beta' Z_i\}$. The model formula includes stratum-specific intercepts and interactions with the treatment arm.
3. The p-value for the treatment effect is extracted from the model summary.

Power is determined as the proportion of simulations where the p-value is less than `alpha`.

Value

A `list` containing:

`results_data` A `data.frame` of sample sizes and corresponding powers.

`results_plot` A `ggplot` object visualizing the power curve.

`results_summary` A `data.frame` with the estimated treatment effect (RMST Ratio).

Note

`status_var` should be `1`/`0`. `arm_var` should be `1`/`0`. `strata_var` is a mandatory argument.

Examples

```
pilot_df_strat <- data.frame(
 time = rexp(120, 0.15),
 status = rbinom(120, 1, 0.6),
 arm = rep(0:1, each = 60),
 region = factor(rep(c("A", "B", "C"), each = 40))
)
pilot_df_strat$time[pilot_df_strat$arm == 1] <- pilot_df_strat$time[pilot_df_strat$arm == 1] * 1.4

power_results <- MS.power.boot(
 pilot_data = pilot_df_strat,
 time_var = "time", status_var = "status", arm_var = "arm", strata_var = "region",
 sample_sizes = c(50, 75),
 L = 10,
 n_sim = 100 # Low n_sim for example
)
print(power_results$results_data)
```

`MS.ss.analytical` *Find Sample Size for a Multiplicative Stratified RMST Model (Analytic)*

Description

Calculates the required sample size for a target power using the analytic (approximate) method from Wang et al. (2019).

Usage

```
MS.ss.analytical(
  pilot_data,
  time_var,
  status_var,
  arm_var,
  strata_var,
  target_power,
  linear_terms = NULL,
  L,
  alpha = 0.05,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

`pilot_data`	A `data.frame` containing pilot study data.
`time_var`	A character string for the time-to-event variable.
`status_var`	A character string for the event status variable (1=event, 0=censored).
`arm_var`	A character string for the treatment arm variable (1=treatment, 0=control).
`strata_var`	A character string for the stratification variable.
`target_power`	A single numeric value for the desired power.
`linear_terms`	An optional character vector of other covariate names.
`L`	The numeric value for the RMST truncation time.
`alpha`	The significance level (Type I error rate).
`n_start`	The starting sample size *per stratum* for the search.
`n_step`	The increment in sample size at each step of the search.
`max_n_per_arm`	The maximum sample size *per stratum* to search up to.
`verbose`	Logical; if `TRUE`, emit progress messages. Default `FALSE`.

Details

This function estimates the log-RMST ratio and its asymptotic variance once from the pilot data, then increases the per-stratum sample size until the target power is reached or the search limit is hit. It uses the same log-linear approximation as `MS.power.analytical`.

Value

A `list` containing:

`results_data` A `data.frame` with the target power and required sample size.

`results_plot` A `ggplot` object visualizing the search path.

`results_summary` A `data.frame` summarizing the estimated log(RMST Ratio).

Examples

```
set.seed(456)
pilot_df_strat_effect <- data.frame(
 time = c(rexp(60, 0.15), rexp(60, 0.08)), # Effect
 status = rbinom(120, 1, 0.7),
 arm = rep(0:1, each = 60),
 region = factor(rep(c("A", "B"), each = 60))
)

ss_results <- MS.ss.analytical(
 pilot_data = pilot_df_strat_effect,
 time_var = "time", status_var = "status", arm_var = "arm", strata_var = "region",
 target_power = 0.80, L = 10,
 n_start = 100, n_step = 50
)
print(ss_results$results_data)
```

MS.ss.boot *Estimate Sample Size for a Multiplicative Stratified RMST Model via Simulation*

Description

Performs sample size estimation based on a multiplicative model for RMST for stratified trials, using iterative bootstrap simulations.

Usage

```
MS.ss.boot(
  pilot_data,
  time_var,
  status_var,
  arm_var,
```

```
  strata_var,
  target_power,
  linear_terms = NULL,
  L,
  n_sim = 1000,
  alpha = 0.05,
  parallel.cores = 1,
  patience = 5,
  n_start = 50,
  n_step = 25,
  max_n_per_arm = 2000,
  verbose = FALSE
)
```

Arguments

pilot_data	A data.frame with pilot study data.
time_var	A character string for the time-to-event variable.
status_var	A character string for the event status variable.
arm_var	A character string for the treatment arm variable.
strata_var	A character string for the stratification variable.
target_power	A single numeric value for the target power (e.g., 0.80).
linear_terms	Optional vector of covariates for the model.
L	The numeric truncation time for RMST.
n_sim	Number of bootstrap simulations per search step.
alpha	The significance level.
parallel.cores	Number of cores for parallel processing.
patience	Number of consecutive non-improving steps in the search before terminating.
n_start	Starting sample size per stratum for the search.
n_step	Increment for the sample size search.
max_n_per_arm	Maximum sample size per stratum to try.
verbose	Logical; if TRUE, emit progress messages. Default FALSE.

Details

This function iteratively searches for the sample size required to achieve a target_power. At each step of the search, it runs a full bootstrap simulation (as described in MS.power.boot) to estimate the power for the current sample size. The search proceeds until the target power is met or other stopping criteria are satisfied. This process can be very computationally intensive.

Value

A list containing:

results_data	A data.frame with the target power and required N.

`results_plot` A ggplot object showing the search path.

`results_summary`
A data.frame with the estimated treatment effect.

Note

`status_var` should be 1/0. `arm_var` should be 1/0. `strata_var` is a mandatory argument.

Examples

```
pilot_df_strat_effect <- data.frame(
 time = c(rexp(60, 0.15), rexp(60, 0.08)), # Effect
 status = rbinom(120, 1, 0.7),
 arm = rep(0:1, each = 60),
 region = factor(rep(c("A", "B", "C"), each = 40))
)
ss_results <- MS.ss.boot(
 pilot_data = pilot_df_strat_effect,
 time_var = "time", status_var = "status", arm_var = "arm", strata_var = "region",
 target_power = 0.80, L = 10,
 n_sim = 100, # Low n_sim for example
 n_start = 100,
 n_step = 50, patience = 2
)
print(ss_results$results_data)
```

---

ph_pwexp_L18_n250 *Simulated dataset: PH piecewise-exponential, L = 18, n = 250*

Description

Columns: time, status, arm, age, gender.

Format

A data frame with 250 rows and 5 variables.

Examples

```
data(ph_pwexp_L18_n250)
summary(ph_pwexp_L18_n250$time)
```

ph_weibull_L24_n300 *Simulated dataset: PH Weibull, L = 24, n = 300*

Description

Columns: time, status, arm, age, gender.

Format

A data frame with 300 rows and 5 variables.

Examples

```
data(ph_weibull_L24_n300)
prop.table(table(ph_weibull_L24_n300$status))
```

print.rmst_power *Print an rmst_power result*

Description

Prints the model metadata and power table returned by `rmst.power()`.

Returns a summary object for printing and further inspection.

Prints and returns the stored ggplot2 object.

Usage

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

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

## S3 method for class 'rmst_power'
plot(x, ...)
```

Arguments

x	An object returned by `rmst.power()`.
...	Additional arguments passed to or from other methods.
object	An object returned by `rmst.power()`.

Value

The input object x, invisibly.

An object of class "summary.rmst_power".

A ggplot2 object, invisibly.

---

print.rmst_ss *Print an rmst_ss result*

Description

Prints the model metadata and sample-size table returned by rmst.ss().

Returns a summary object for printing and further inspection.

Prints and returns the stored ggplot2 object.

Usage

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

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

## S3 method for class 'rmst_ss'
plot(x, ...)
```

Arguments

x	An object returned by rmst.ss().
...	Additional arguments passed to or from other methods.
object	An object returned by rmst.ss().

Value

The input object x, invisibly.

An object of class "summary.rmst_ss".

A ggplot2 object, invisibly.

rebuild_manifest *Rebuild manifest for an existing output directory (no re-simulation)*

Description

Reads datasets already written in `out_dir` and reconstructs a `manifest.rds` with rich metadata (model, baseline, effects, etc.).

Usage

```
rebuild_manifest(
  base_recipe,
  vary,
  out_dir,
  filename_template = "sc{scenario_id}_r{rep}"
)
```

Arguments

`base_recipe`	A validated recipe list (use `validate_recipe()` if needed).
`vary`	Named list used originally in `generate_recipe_sets()` for the grid.
`out_dir`	Directory that already contains the datasets.
`filename_template`	The same template you used when writing files (default `"sc{scenario_id}_r{rep}"`). It may also include tokens for dotted paths from `vary`.

Value

The rebuilt manifest (also writes `manifest.rds` in `out_dir`).

recipe_grid *Expand a recipe over a grid of values (list-only)*

Description

Expand a recipe over a grid of values (list-only)

Usage

```
recipe_grid(base, vary)
```

Arguments

`base`	A recipe list.
`vary`	Named list of vectors; keys are dotted paths.

Value

A list of recipe lists.

Examples

```
r <- recipe_quick_aft(60, "aft_lognormal", baseline=list(mu=2.7, sigma=0.6),
     treat_effect=-0.2, covariates=list(list(name="x", type="continuous", dist="normal",
      params=list(mean=0, sd=1))))
recipe_grid(r, list(n=c(60,80), "event_time.effects.treatment"=c(-0.2,-0.4)))
```

`recipe_quick_aft` *Quick AFT recipe builder for list-based simulation recipes*

Description

Quick AFT recipe builder for list-based simulation recipes

Usage

```
recipe_quick_aft(
  n,
  model = c("aft_lognormal", "aft_weibull"),
  baseline,
  treat_effect,
  covariates,
  target_censoring = 0.25,
  allocation = "1:1",
  seed = NULL
)
```

Arguments

`n`	Sample size.
`model`	One of `"aft_lognormal"` or `"aft_weibull"`.
`baseline`	Baseline parameter list (see model).
`treat_effect`	Numeric treatment coefficient (on log-time scale).
`covariates`	Covariate definitions (list of defs).
`target_censoring`	Target overall censoring fraction (0-1).
`allocation`	Allocation ratio string (e.g., "1:1").
`seed`	Optional seed retained as recipe/provenance metadata. It does not set the random-number generator.

Value

A recipe list suitable for `simulate_from_recipe`.

Examples

```
covs <- list(list(name="x", type="continuous", dist="normal", params=list(mean=0, sd=1)))
r <- recipe_quick_aft(120, "aft_lognormal",
       baseline=list(mu=2.3, sigma=0.5), treat_effect=-0.2,
       covariates=covs, target_censoring=0.25, allocation="1:1")
set.seed(1)
dat <- simulate_from_recipe(r)
```

`recipe_quick_ph` *Quick PH recipe builder*

Description

Convenience wrapper that builds a recipe list for proportional-hazard (PH) models, parallel to `recipe_quick_aft` for AFT models. The returned recipe is validated and ready for `simulate_from_recipe`.

Usage

```
recipe_quick_ph(
  n,
  model = c("ph_exponential", "ph_weibull", "ph_gompertz", "cox_pwexp"),
  baseline,
  treat_effect,
  covariates = list(),
  target_censoring = 0.25,
  allocation = "1:1",
  seed = NULL
)
```

Arguments

`n`	Total sample size (integer).
`model`	PH model. One of `"ph_exponential"`, `"ph_weibull"`, `"ph_gompertz"`, `"cox_pwexp"`.
`baseline`	Named list of baseline hazard parameters: **ph_exponential** `list(rate = ...)` **ph_weibull** `list(shape = ..., scale = ...)` **ph_gompertz** `list(shape = ..., rate = ...)` **cox_pwexp** `list(rates = c(...), cuts = c(...))`
`treat_effect`	Numeric log-hazard ratio for the treatment arm.
`covariates`	List of covariate definitions. Use `covar_continuous`, `covar_binary`, or `covar_categorical`. Default `list()`.
`target_censoring`	Target overall censoring fraction (0-1). Default `0.25`.
`allocation`	Treatment allocation ratio string, e.g. `"1:1"` (default) or `"1:2"`.
`seed`	Optional seed retained as recipe/provenance metadata. It does not set the random-number generator.

Value

A recipe list suitable for simulate_from_recipe.

See Also

recipe_quick_aft, rmst.sim, simulate_from_recipe

Examples

```
r <- recipe_quick_ph(100, "ph_weibull",
       baseline = list(shape = 1.5, scale = 10),
       treat_effect = -0.5,
       covariates = list(covar_continuous("age")),
       target_censoring = 0.30)
set.seed(1)
dat <- simulate_from_recipe(r)
```

---

rmst.power *Power analysis for RMST-based models via formula interface*

Description

Routes a formula-based call to the matching RMST power routine.

Usage

```
rmst.power(
  formula,
  data,
  arm,
  sample_sizes,
  L,
  strata = NULL,
  strata_type = c("additive", "multiplicative"),
  dep_cens = FALSE,
  type = c("analytical", "boot"),
  alpha = 0.05,
  n_sim = 1000L,
  parallel.cores = 1L,
  verbose = FALSE
)
```

Arguments

formula	A formula of the form `Surv(time, status) ~ cov1 + cov2`. Use `s()` wrapping (mgcv style) for smooth terms: `Surv(time, status) ~ s(age)`.
data	A `data.frame` containing the reference (pilot) data.
arm	Character string naming the treatment arm column (binary 0/1).
sample_sizes	Integer vector of per-arm (or per-stratum) sample sizes to evaluate.
L	Numeric truncation time for RMST.
strata	Character column name, one-sided formula (`~col`), or `NULL` (default). Ignored when `dep_cens = TRUE`.
strata_type	One of `"additive"` (default) or `"multiplicative"`. Only used when `strata` is non-`NULL` and `dep_cens = FALSE`.
dep_cens	Logical; use dependent-censoring model? Default `FALSE`.
type	One of `"analytical"` or `"boot"`. Auto-switched with a message when the requested type is unavailable for the chosen model.
alpha	Significance level. Default `0.05`.
n_sim	Number of bootstrap replicates (boot methods only). Default `1000`.
parallel.cores	Number of cores for parallel processing. Default `1`.
verbose	Logical; if `TRUE`, emit progress messages from the underlying calculation. Default `FALSE`.

Value

An object of class `c("rmst_power", "list")` with elements `results_data`, `results_plot`, `results_summary`, `model_output`, and `.meta`.

See Also

`rmst.ss`, `print.rmst_power`, `summary.rmst_power`, `plot.rmst_power`

Examples

```
data(aft_lognormal_L12_n150, package = "RMSTpowerBoost")
r <- rmst.power(Surv(time, status) ~ age,
                data = aft_lognormal_L12_n150,
                arm  = "arm",
                sample_sizes = c(50, 100, 150),
                L    = 12)
print(r)
s <- summary(r)
plot(r)
```

`rmst.sim` *Simulate survival data for RMST analysis*

Description

A unified, single-call wrapper for generating survival data suitable for use as reference/pilot data in `rmst.power` and `rmst.ss`. Supports all seven built-in event-time models (AFT and PH families).

Usage

```
rmst.sim(
  n,
  model = "aft_lognormal",
  baseline,
  treat_effect = 0,
  covariates = list(),
  target_censoring = 0.25,
  allocation = "1:1",
  L = NULL,
  seed = NULL
)
```

Arguments

`n`	Total sample size (split by `allocation` ratio).
`model`	Event-time model. One of: `"aft_lognormal"` (default), `"aft_weibull"`, `"aft_loglogistic"`, `"ph_exponential"`, `"ph_weibull"`, `"ph_gompertz"`, `"cox_pwexp"`.
`baseline`	Named list of baseline parameters (model-specific; see `recipe_quick_aft` and `recipe_quick_ph`).
`treat_effect`	Numeric treatment coefficient. Log-time scale for AFT models; log-hazard ratio for PH models. Default `0`.
`covariates`	List of covariate definitions. Elements can be created with `covar_continuous`, `covar_binary`, or `covar_categorical`, or supplied as raw named lists in the existing recipe format. Default `list()` (no covariates).
`target_censoring`	Target overall censoring fraction (0-1). Default `0.25`.
`allocation`	Treatment allocation ratio string, e.g. `"1:1"` (default).
`L`	Optional numeric truncation time. Stored as an attribute on the returned object for downstream use by `rmst.power` / `rmst.ss`. Does not affect data generation.
`seed`	Optional seed retained as recipe/provenance metadata. It does not set the random-number generator; call `set.seed()` before `rmst.sim()` for reproducible simulation.

Details

Internally routes to recipe_quick_aft for AFT models and recipe_quick_ph for PH models, then calls simulate_from_recipe.

Value

A data.frame of class c("rmst_simdata", "data.frame") with columns time, status, arm (when treatment is present), and one column per covariate. Attributes:

recipe The validated recipe list used for generation.

L The truncation time if supplied, else NULL.

achieved_censoring Actual censoring fraction achieved.

See Also

rmst.power, rmst.ss, recipe_quick_ph, recipe_quick_aft, covar_continuous

Examples

```
df <- rmst.sim(
  n            = 150,
  model        = "aft_lognormal",
  baseline     = list(mu = 2.2, sigma = 0.5),
  treat_effect = -0.3,
  covariates   = list(covar_continuous("age"), covar_binary("female")),
  L            = 12,
  seed         = 42
)
print(df)
s <- summary(df)
```

---

rmst.ss *Sample size estimation for RMST-based models via formula interface*

Description

Routes a formula-based call to the matching RMST sample-size routine.

Usage

```
rmst.ss(
  formula,
  data,
  arm,
  target_power,
  L,
  strata = NULL,
```

```
  strata_type = c("additive", "multiplicative"),
  dep_cens = FALSE,
  type = c("analytical", "boot"),
  alpha = 0.05,
  n_sim = 1000L,
  parallel.cores = 1L,
  n_start = 50L,
  n_step = 25L,
  max_n = 2000L,
  patience = 5L,
  verbose = FALSE
)
```

Arguments

formula	A formula of the form `Surv(time, status) ~ cov1 + cov2`.
data	A `data.frame` containing the reference (pilot) data.
arm	Character string naming the treatment arm column (binary 0/1).
target_power	Numeric target power (e.g., `0.80`).
L	Numeric truncation time for RMST.
strata	Character column name, one-sided formula (`~col`), or `NULL`.
strata_type	One of `"additive"` (default) or `"multiplicative"`.
dep_cens	Logical; use dependent-censoring model? Default `FALSE`.
type	One of `"analytical"` or `"boot"`.
alpha	Significance level. Default `0.05`.
n_sim	Number of bootstrap replicates. Default `1000`.
parallel.cores	Number of cores for parallel processing. Default `1`.
n_start	Starting sample size for the search. Default `50`.
n_step	Search increment. Default `25`.
max_n	Maximum sample size to try. Default `2000`.
patience	Number of consecutive non-improving steps before stopping. Default `5`.
verbose	Logical; if `TRUE`, emit progress messages from the underlying calculation. Default `FALSE`.

Value

An object of class `c("rmst_ss", "list")` with elements `results_data`, `results_plot`, `results_summary`, `model_output`, and `.meta`.

See Also

`rmst.power`, `print.rmst_ss`, `summary.rmst_ss`, `plot.rmst_ss`

Examples

```
data(aft_lognormal_L12_n150, package = "RMSTpowerBoost")
r <- rmst.ss(Surv(time, status) ~ age,
             data         = aft_lognormal_L12_n150,
             arm          = "arm",
             target_power = 0.80,
             L            = 12)
print(r)
```

---

run_app *Launch the RMSTpowerBoost Shiny Application*

Description

Launches the Shiny application bundled with the package. App-specific dependencies are installed only when they are needed.

Usage

```
run_app(install_missing = TRUE, repos = getOption("repos"))
```

Arguments

install_missing	Logical; if TRUE, prompt to install missing app dependencies.
repos	CRAN mirror(s) passed to utils::install.packages() when installing missing app dependencies.

Value

Invisible return value from shiny::runApp().

Examples

```
RMSTpowerBoost::run_app()
```

simulate_from_recipe *Simulate a dataset from a validated recipe (list-only)*

Description

Simulate a dataset from a validated recipe (list-only)

Usage

```
simulate_from_recipe(recipe, seed = NULL)
```

Arguments

recipe	A validated recipe list (use validate_recipe()).
seed	Optional integer retained as recipe/provenance metadata; it does not set the random-number generator. Call set.seed() before this function for reproducible simulation.

Value

A data.frame with columns time, status, arm (if treatment present), plus covariates. Attribute "achieved_censoring" is attached.

Examples

```
covs <- list(list(name="x", type="continuous", dist="normal", params=list(mean=0, sd=1)))
rec <- recipe_quick_aft(120, "aft_lognormal",
        baseline=list(mu=2.2, sigma=0.5), treat_effect=-0.2,
        covariates=covs, target_censoring=0.25)
set.seed(11)
dat <- simulate_from_recipe(rec)
```

validate_recipe *Validate a simulation recipe (list-only schema)*

Description

Checks/normalizes the simulation recipe and fills reasonable defaults. This function does not use or require an analysis horizon; specify L later in downstream analysis functions.

Usage

```
validate_recipe(recipe)
```

Arguments

recipe	A named list defining n, covariates, treatment (optional), event_time (model, baseline, effects, optional frailty), and censoring.

Value

A validated recipe list.

Examples

```
r <- recipe_quick_aft(
  n = 100,
  model = "aft_lognormal",
  baseline = list(mu = 2.2, sigma = 0.5),
  treat_effect = -0.2,
  covariates = list(list(name="x", type="continuous", dist="normal",
                         params=list(mean=0, sd=1))),
  target_censoring = 0.25, allocation = "1:1"
)
r2 <- validate_recipe(r)
```

Index