Package ‘beastt’

July 22, 2025

Title Bayesian Evaluation, Analysis, and Simulation Software Tools for Trials

Version 0.0.3

Description Bayesian dynamic borrowing with covariate adjustment via inverse probability weighting for simulations and data analyses in clinical trials. This makes it easy to use propensity score methods to balance covariate distributions between external and internal data. This methodology based on Psioda et al (2025) <doi:10.1080/10543406.2025.2489285>.

License GPL (>= 3)

URL `https://gsk-biostatistics.github.io/beastt/`, `https://github.com/GSK-Biostatistics/beastt`

BugReports `https://github.com/GSK-Biostatistics/beastt/issues`

Suggests knitr, mvtnorm, rmarkdown, spelling, testthat (>= 3.0.0), tibble, vdiffr, survival

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Imports cli, cobalt, distributional, dplyr, generics, ggdist, ggplot2, methods, mixtools, purrr, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.18.1), rstantools (>= 2.4.0), stringr, tidyr

VignetteBuilder knitr

Depends R (>= 4.1.0)

LazyData true

Language en-US

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

NeedsCompilation yes

Author Christina Fillmore [aut, cre] (ORCID: <https://orcid.org/0000-0003-0595-2302>),
Nate Bean [aut] (ORCID: <https://orcid.org/0000-0001-9946-0119>),
Abi Terry [aut],
Ben Arancibia [aut],
GlaxoSmithKline Research & Development Limited [cph, fnd],
Trustees of Columbia University [cph] (R/stanmodels.R, configure, configure.win)

Maintainer Christina Fillmore <christina.e.fillmore@gsk.com>

Repository CRAN

Date/Publication 2025-05-15 11:40:06 UTC

Contents

`beastt-package` *The 'beastt' package.*

Description

Inverse Probability Weighted Bayesian Dynamic Borrowing

References

Stan Development Team (NA). RStan: the R interface to Stan. R package version 2.32.3. https://mc-stan.org

---

`approx_mvn_at_time` *Approximate Multivariate Normal Distribution as Beta at a Specific Time*

Description

Converts a multivariate normal distribution for Weibull parameters (or a mixture of these distributions) into an approximate beta distribution for the survival probability at a specific time point. This is particularly useful for visualizing survival probabilities in sweet spot plots

Usage

```
approx_mvn_at_time(x, time)
```

Arguments

`x`	A vector of distributional objects that must be either multivariate normal distributions or mixtures of multivariate normal distributions. For Weibull models, these represent distributions of the log(shape) and log(scale) parameters.
`time`	A numeric value specifying survival time at which to calculate the survival probability.

Details

The function performs the following steps:

- For each multivariate normal distribution, it generates 10,000 samples of the Weibull parameters
- Calculates the corresponding survival probabilities at the specified time using the Weibull survival function
- Fits a beta distribution to match the mean and variance of these survival probabilities
- For mixture distributions, it performs this approximation for each component and creates a new mixture with the same weights

The conversion uses the relationship between Weibull parameters and survival probability: S(t) = exp(-(t*exp(log_scale))^exp(log_shape)).

Value

A vector of beta distributional (or mixture of beta distributional) objects approximating the survival probabilities at the specified time point. If the input is a mixture distribution, the output will be a mixture of beta distributions with the same weights.

See Also

`sweet_spot_plot()`

Examples

```
library(distributional)

# Create a multivariate normal distribution for Weibull parameters
# (log(shape), log(scale))
mvn_dist <- dist_multivariate_normal(
  mu = list(c(0, -1)),  # log(shape) = 0, log(scale) = -1
  sigma = list(matrix(c(0.1, 0, 0, 0.1), nrow = 2))
)

# Approximate as beta distribution for survival at time t=12
beta_approx <- approx_mvn_at_time(mvn_dist, time = 12)
```

`avg_dist` *Calculate Average Distribution from Multiple Distributional Objects*

Description

Compute a single "average" distribution from a vector of distributional objects. This function calculates the mean of each hyperparameter across all input distributions and returns a new distributional object of the same family with these averaged hyperparameters.

Usage

```
avg_dist(x)
```

Arguments

`x`	A vector of distributional objects of the same family (beta, normal, multivariate normal, or mixture).

Details

The function supports four distribution families:

- Beta distributions: Averages the shape1 and shape2 hyperparameters
- Normal distributions: Averages the mean and standard deviation hyperparameters
- Multivariate normal distributions: Averages the location vectors and covariance matrices
- Mixture distributions: Same as above for each distribution type, where averaging is done by component. Also averages the mixture weight.

For multivariate normal distributions, both the location vector and covariance matrix are averaged element-wise.

Value

A single distributional object of the same family as the input, with hyperparameters set equal to the average of all input distribution hyperparameters.

Examples

```
library(distributional)

# Beta distributions
beta_dists <- c(
  dist_beta(shape1 = 2, shape2 = 5),
  dist_beta(shape1 = 3, shape2 = 3),
  dist_beta(shape1 = 4, shape2 = 2)
)
avg_dist(beta_dists) |> parameters()

# Normal distributions
norm_dists <- c(
  dist_normal(mu = 0, sigma = 1),
  dist_normal(mu = 2, sigma = 2),
  dist_normal(mu = 4, sigma = 3)
)
avg_dist(norm_dists) |> parameters()

# Multivariate normal distributions
mvn_dists <- c(
  dist_multivariate_normal(mu = list(c(0, 0)), sigma = list(matrix(c(1, 0, 0, 1), nrow = 2))),
  dist_multivariate_normal(mu = list(c(1, 1)), sigma = list(matrix(c(2, 0, 0, 2), nrow = 2)))
)
```

```
avg_dist(mvn_dists) |> parameters()
```

`binary_sim_df` *Binary Simulation Data*

Description

This is an example of output from a simulation study that investigates the operating characteristics of inverse probability weighted Bayesian dynamic borrowing for the case with a binary outcome. This output was generated based on the binary simulation template. For this simulation study, only the degree of covariate imbalance (as indicated by `population`) and the marginal treatment effect were varied.

Usage

```
binary_sim_df
```

Format

`binary_sim_df` **A data frame with 255 rows and 6 columns::**

population Populations defined by different covariate imbalances

marg_trt_eff Marginal treatment effect

true_control_RR True control response rate on the marginal scale

reject_H0_yes Probability of rejecting the null hypothesis in the case with borrowing

no_borrowing_reject_H0_yes Probability of rejecting the null hypothesis without borrowing

pwr_prior Vector of power priors (or some other informative prior distribution for the control marginal parameter of interest based on the external data) as distributional objects

---

`bootstrap_cov` *Bootstrap Covariate Data*

Description

Bootstrap Covariate Data

Usage

```
bootstrap_cov(
  external_dat,
  n,
  imbal_var = NULL,
  imbal_prop = NULL,
  ref_val = 0
)
```

Arguments

`external_dat`	Data frame of the external data from which to bootstrap covariate vectors
`n`	Number of rows in the output dataset
`imbal_var`	Optional variable indicating which covariate's distribution should be altered to incorporate imbalance compared to the external data. If left `NULL`, the distributions of all covariates in the output dataset will match the distributions in the external dataset. The imbalance variable must be binary.
`imbal_prop`	Optional imbalance proportion, required if an imbalance variable is specified. This defines the proportion of individuals with the reference value of the imbalance variable in the returned dataset. This can either be a single proportion or a vector of proportions, in which case a list of datasets is returned.
`ref_val`	Optional value corresponding to the reference level of the binary imbalance variable, if specified

Details

Covariate data can be generated for `n` individuals enrolled in the internal trial by bootstrap sampling entire covariate vectors from the external data, thus preserving the correlation between the covariates. If both `imbal_var` = `NULL` and `imbal_prop` = `NULL`, the function returns a single data frame in which the distributions of each covariate align with the covariate distributions from the external data (i.e., balanced covariate distributions across the two trials). Alternatively, covariate imbalance can be incorporated into the generated sample with respect to a binary covariate (`imbal_var`) such that a specified proportion (`imbal_prop`) of individuals in the resulting sample will have the reference level (`ref_val`) of this imbalance covariate. In this case, stratified bootstrap sampling is employed with the imbalance covariate as the stratification factor.

Multiple samples with varying degrees of imbalance can be generated simultaneously by defining `imbal_prop` to be a vector of values. The function then returns a list of data frames with a length equal to the number of specified imbalance proportions.

Value

Data frame with the same number of columns as the external data frame and n number of rows (if the length of `imbal_prop` is 0 or 1); otherwise, a list of data frames with a length equal to that of `imbal_prop`

Examples

```
# Return one data frame with covariate distributions similar to external data
samp_balance <- bootstrap_cov(ex_binary_df, n = 1000)

# Return a list of two data frames that incorporate imbalance w.r.t. covariate 2
samp_imbalance <- bootstrap_cov(ex_binary_df, n = 1000, imbal_var = cov2,
                                imbal_prop = c(0.25, 0.5), ref_val = 0)
```

calc_cond_binary *Calculate Conditional Drift and Treatment Effect for Binary Outcome Models*

Description

In order to properly generate binary response data for the internal trial as part of a simulation study that investigates inverse probability weighting, we need to translate the desired marginal drift and treatment effect to the corresponding conditional drift and treatment effect that can then be added into a binary outcome model (e.g., logistic regression model) used to simulate response data.

Usage

```
calc_cond_binary(population, glm, marg_drift, marg_trt_eff)
```

Arguments

`population`	A very large data frame (e.g., number of rows $\geq$ 100,000) where the columns correspond to the covariates defined in the logistic regression model object. This data frame should be constructed to represent the population of the internal trial according to the assumed covariate distributions (possibly imbalanced from the external data).
`glm`	Logistic regression model object fit using the external data
`marg_drift`	Vector of marginal drift values
`marg_trt_eff`	Vector of marginal treatment effect values

Details

In simulation studies that investigate the properties of inverse probability weighted Bayesian dynamic borrowing, scenarios should be considered in which the underlying response rates for the internal and external control populations differ by varying amounts due to unmeasured confounding (i.e., drift, where positive values indicate a higher response rate for the internal population). While values of drift and treatment effect (i.e., risk difference) can be defined on the marginal scale for simulation studies, we must first convert these values to the conditional scale and then include these terms, along with covariates, in a logistic regression outcome model when generating response data for the internal arms. Doing so allows us to assume a relationship between the covariates and the response variable while properly accounting for drift and treatment effect.

To identify the conditional drift and treatment effect that correspond to specified values of marginal drift and treatment effect, we first bootstrap covariate vectors from the external data (e.g., $N \geq 100,000$) to construct a "population" that represents both the internal trial (possibly incorporating intentional covariate imbalance) and the external trial *after* standardizing it to match the covariate distributions of the internal trial (allowing us to control for measured confounding from potential imbalance in the covariate distributions). Measured confounding can be incorporated into the data generation by bootstrapping a very large data frame (`population`) in which the distribution of at least one covariate is intentionally varied from that of the external data; additional *unmeasured* drift can be incorporated through the translation of specified marginal values (`marg_drift`) to conditional values.

Let Δ and δ denote the marginal and conditional drift, respectively. For a specified value of Δ, we can identify the corresponding δ as the value that, when added as an additional term in the logistic regression model (i.e., change in the intercept) for each individual in the population, increases/decreases the population-averaged conditional probabilities of response by an amount approximately equal to Δ. That is, the optimal δ minimizes

$$\left|\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta\right)}{1+\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta\right)}-\frac{1}{N}\sum_{i=1}^{N}\frac{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}\right)}{1+\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}\right)}\right)-\Delta\right|,$$

where $\boldsymbol{\beta}_{EC}$ is the vector of regression coefficients from the logistic regression model (`glm`) fit to the external control data (assumed here to be the "true" covariate effects when generating response data) and $\boldsymbol{x}_i$ is a vector of covariates (including an intercept term) from the bootstrapped population of size N. In the formula above, the first and second terms correspond to the population-averaged conditional probabilities (i.e., the marginal response rates) of the internal control population with drift and the external control population (with covariate distributions standardized to match the internal trial), respectively.

If we now denote the marginal and conditional treatment effect by Γ and γ, respectively, we can use a similar process to identify the optimal γ that approximately corresponds to the specified value of Γ, which is done by minimizing the following:

$$\left|\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta+\gamma\right)}{1+\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta+\gamma\right)}-\frac{1}{N}\sum_{i=1}^{N}\frac{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta\right)}{1+\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta\right)}\right)-\Gamma\right|,$$

where the first term is the average of the conditional probabilities of response (i.e., the marginal response rate) of the internal treated population.

See here for a simulation example with a binary outcome.

Value

tibble of all combinations of the marginal drift and treatment effect. For each row the conditional drift and treatment effect has been calculated as well as the true response rates for the control and treated populations.

Examples

```
library(dplyr)
# Model "true" regression coefficients using the external data
logit_mod <- glm(y ~ cov1 + cov2 + cov3 + cov4, data = ex_binary_df, family = binomial)

# Bootstrap internal control "population" with imbalance w.r.t. covariate 2
pop_int_ctrl <- bootstrap_cov(ex_binary_df, n = 100000, imbal_var = cov2,
                              imbal_prop = 0.25, ref_val = 0) |>
                              select(-subjid, -y)  # keep only covariate columns

# Convert the marginal drift and treatment effects to conditional
calc_cond_binary(population = pop_int_ctrl, glm = logit_mod,
                 marg_drift = c(-.1, 0, .1), marg_trt_eff = c(0, .15))
```

calc_cond_weibull *Calculate Conditional Drift and Treatment Effect for Time-to-Event Outcome Models*

Description

In order to properly generate time-to-event (TTE) outcome data for the internal trial as part of a simulation study that investigates inverse probability weighting, we need to translate the desired marginal drift and treatment effect to the corresponding conditional drift and treatment effect that can then be added into a TTE outcome model (e.g., Weibull proportional hazards regression model) used to simulate response data.

Usage

```
calc_cond_weibull(
  population,
  weibull_ph_mod,
  marg_drift,
  marg_trt_eff,
  analysis_time
)
```

Arguments

`population`	A very large data frame (e.g., number of rows $\geq$ 100,000) where the columns correspond to the covariates defined in the `survreg` object for the Weibull proportional hazards model. This data frame should be constructed to represent the population of the internal trial according to the assumed covariate distributions (possibly imbalanced from the external data).
`weibull_ph_mod`	`survreg` object corresponding to a Weibull proportional hazards model fit using the external data
`marg_drift`	Vector of marginal drift values
`marg_trt_eff`	Vector of marginal treatment effect values
`analysis_time`	A single time point when survival probabilities will be calculated

Details

In simulation studies that investigate the properties of inverse probability weighted Bayesian dynamic borrowing, scenarios should be considered in which the underlying survival probabilities at some prespecified time t (`analysis_time`) for the internal and external control populations differ by varying amounts due to unmeasured confounding (i.e., drift, where positive values indicate a higher survival probability for the internal population). While values of drift and treatment effect (i.e., difference between the survival probabilities at time t for the treated and control populations) can be defined on the marginal scale for simulation studies, we must first convert these values to the conditional scale and then include these terms, along with covariates, in a Weibull proportional hazards (PH) regression outcome model when generating time-to-event (TTE) data for the internal arms. Doing so allows us to assume a relationship between the covariates and the response variable while properly accounting for drift and treatment effect.

To identify the conditional drift and treatment effect that correspond to specified values of marginal drift and treatment effect, we first bootstrap covariate vectors from the external data (e.g., $N \geq 100,000$) to construct a "population" that represents both the internal trial (possibly incorporating intentional covariate imbalance) and the external trial *after* standardizing it to match the covariate distributions of the internal trial (allowing us to control for measured confounding from potential imbalance in the covariate distributions). Measured confounding can be incorporated into the data generation by bootstrapping a very large data frame (`population`) in which the distribution of at least one covariate is intentionally varied from that of the external data; additional *unmeasured* drift can be incorporated through the translation of specified marginal values (`marg_drift`) to conditional values.

Let Δ and δ denote the marginal and conditional drift, respectively. For a specified value of Δ, we can identify the corresponding δ as the value that, when added as an additional term in the Weibull PH model survival function (i.e., additive change in the intercept) for each individual in the population, increases/decreases the population-averaged conditional probabilities of survival at time t by an amount approximately equal to Δ. That is, the optimal δ minimizes

$$\left|\left(\frac{1}{N}\sum_{i=1}^{N}\exp\left(-\left\{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta\right)\times t\right\}^{\alpha_{EC}}\right)-\frac{1}{N}\sum_{i=1}^{N}\exp\left(-\left\{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}\right)\times t\right\}^{\alpha_{EC}}\right)\right)-\Delta\right|,$$

where α_{EC} is the Weibull shape parameter, $\boldsymbol{\beta}_{EC}$ is a vector of regression coefficients, and $\boldsymbol{x}_i$ is a vector of covariates (including an intercept term) from the bootstrapped population of size N. We note that $\alpha_{EC} = 1/\sigma_{EC}$ and $\boldsymbol{\beta}_{EC} = -\boldsymbol{\xi}_{EC}$ are calculated as functions of the scale parameter (σ_{EC}) and coefficients ($\boldsymbol{\xi}_{EC}$) estimated by the `survreg` object that was fit to the external data, and we assume here that these estimates are the "true" shape and covariate effects when generating response data. In the formula above, the first and second terms correspond to the population-averaged conditional survival functions (i.e., the marginal survival probabilities) at time t for the internal control population with drift and the external control population (with covariate distributions standardized to match the internal trial), respectively.

If we now denote the marginal and conditional treatment effect by Γ and γ, respectively, we can use a similar process to identify the optimal γ that approximately corresponds to the specified value of Γ, which is done by minimizing the following:

$$\left|\left(\frac{1}{N}\sum_{i=1}^{N}\exp\left(-\left\{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta+\gamma\right)\times t\right\}^{\alpha_{EC}}\right)-\frac{1}{N}\sum_{i=1}^{N}\exp\left(-\left\{\exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}_{EC}+\delta\right)\times t\right\}^{\alpha_{EC}}\right)\right)-\Gamma\right|,$$

where the first term is the average of the conditional survival functions (i.e., the marginal survival probabilities) at time t for the internal treated population.

See here for a simulation example with a time-to-event outcome.

Value

tibble of all combinations of the marginal drift and treatment effect. For each row the conditional drift and treatment effect has been calculated as well as the true marginal survival probabilities at time `t` for the control and treatment populations.

Examples

```
library(dplyr)
library(survival)
# Model "true" regression coefficients using the external data
weibull_ph_mod <- survreg(Surv(y, event) ~ cov1 + cov2 + cov3 + cov4, data = ex_tte_df,
                          dist = "weibull")

# Bootstrap internal control "population" with imbalance w.r.t. covariate 2
pop_int_ctrl <- bootstrap_cov(ex_tte_df, n = 100000, imbal_var = cov2,
                              imbal_prop = 0.25, ref_val = 0) |>
  select(c(cov1, cov2, cov3, cov4))     # keep only covariate columns

# Convert the marginal drift and treatment effects to conditional
calc_cond_weibull(population = pop_int_ctrl, weibull_ph_mod,
                  marg_drift = c(-.1, 0, .1), marg_trt_eff = c(0, .10),
                  analysis_time = 12)
```

`calc_post_beta` *Calculate Posterior Beta*

Description

Calculate a posterior distribution that is beta (or a mixture of beta components). Only the relevant treatment arms from the internal dataset should be read in (e.g., only the control arm if constructing a posterior distribution for the control response rate).

Usage

```
calc_post_beta(internal_data, response, prior)
```

Arguments

`internal_data`	A tibble of the internal data.
`response`	Name of response variable
`prior`	A distributional object corresponding to a beta distribution or a mixture distribution of beta components

Details

For a given arm of an internal trial (e.g., the control arm or an active treatment arm) of size N_I, suppose the response data are binary such that $Y_i \sim$ Bernoulli(θ), $i = 1, \ldots, N_I$. The posterior distribution for θ is written as

$$\pi(\theta \mid \boldsymbol{y}) \propto \mathcal{L}(\theta \mid \boldsymbol{y})\, \pi(\theta),$$

where $\mathcal{L}(\theta \mid \boldsymbol{y})$ is the likelihood of the response data from the internal arm and $\pi(\theta)$ is a prior distribution on θ (either a beta distribution or a mixture distribution with an arbitrary number of beta components). The posterior distribution for θ is either a beta distribution or a mixture of beta components depending on whether the prior is a single beta distribution or a mixture distribution.

Value

distributional object

Examples

```
library(dplyr)
library(distributional)
calc_post_beta(internal_data = filter(int_binary_df, trt == 1),
               response = y,
               prior = dist_beta(0.5, 0.5))
```

`calc_post_norm` *Calculate Posterior Normal*

Description

Calculate a posterior distribution that is normal (or a mixture of normal components). Only the relevant treatment arms from the internal dataset should be read in (e.g., only the control arm if constructing a posterior distribution for the control mean).

Usage

```
calc_post_norm(internal_data, response, prior, internal_sd = NULL)
```

Arguments

`internal_data`	A tibble of the internal data.
`response`	Name of response variable
`prior`	A distributional object corresponding to a normal distribution, a t distribution, or a mixture distribution of normal and/or t components
`internal_sd`	Standard deviation of internal response data if assumed known. It can be left as `NULL` if assumed unknown

Details

For a given arm of an internal trial (e.g., the control arm or an active treatment arm) of size N_I, suppose the response data are normally distributed such that $Y_i \sim N(\theta, \sigma_I^2)$, $i = 1, \ldots, N_I$. If σ_I^2 is assumed known, the posterior distribution for θ is written as

$$\pi(\theta \mid \boldsymbol{y}, \sigma_I^2) \propto \mathcal{L}(\theta \mid \boldsymbol{y}, \sigma_I^2)\; \pi(\theta),$$

where $\mathcal{L}(\theta \mid \boldsymbol{y}, \sigma_I^2)$ is the likelihood of the response data from the internal arm and $\pi(\theta)$ is a prior distribution on θ (either a normal distribution, a t distribution, or a mixture distribution with an arbitrary number of normal and/or t components). Any t components of the prior for θ are approximated with a mixture of two normal distributions.

If σ_I^2 is unknown, the marginal posterior distribution for θ is instead written as

$$\pi(\theta \mid \boldsymbol{y}) \propto \left\{ \int_0^\infty \mathcal{L}(\theta, \sigma_I^2 \mid \boldsymbol{y})\; \pi(\sigma_I^2)\; d\sigma_I^2 \right\} \times \pi(\theta).$$

In this case, the prior for σ_I^2 is chosen to be $\pi(\sigma_I^2) = (\sigma_I^2)^{-1}$ such that $\left\{ \int_0^\infty \mathcal{L}(\theta, \sigma_I^2 \mid \boldsymbol{y})\; \pi(\sigma_I^2)\; d\sigma_I^2 \right\}$ becomes a non-standardized t distribution. This integrated likelihood is then approximated with a mixture of two normal distributions.

If `internal_sd` is supplied a positive value and `prior` corresponds to a single normal distribution, then the posterior distribution for θ is a normal distribution. If `internal_sd = NULL` or if other types of prior distributions are specified (e.g., mixture or t distribution), then the posterior distribution is a mixture of normal distributions.

Value

distributional object

Examples

```
library(distributional)
library(dplyr)
post_treated <- calc_post_norm(internal_data = filter(int_norm_df, trt == 1),
                               response = y,
                               prior = dist_normal(0.5, 10),
                               internal_sd = 0.15)
```

`calc_post_weibull` *Calculate Posterior Weibull*

Description

Calculate a posterior distribution for the probability of surviving past a given analysis time(s) for time-to-event data with a Weibull likelihood. Only the relevant treatment arms from the internal dataset should be read in (e.g., only the control arm if constructing a posterior distribution for the control survival probability).

Usage

```
calc_post_weibull(internal_data, response, event, prior, analysis_time, ...)
```

Arguments

`internal_data`	This can either be a propensity score object or a tibble of the internal data.
`response`	Name of response variable
`event`	Name of event indicator variable (1: event; 0: censored)
`prior`	A distributional object corresponding to a multivariate normal distribution or a mixture of 2 multivariate normals. The first element of the mean vector and the first row/column of covariance matrix correspond to the log-shape parameter, and the second element corresponds to the intercept (i.e., log-inverse-scale) parameter.
`analysis_time`	Vector of time(s) when survival probabilities will be calculated
`...`	rstan sampling option. This overrides any default options. For more information, see `rstan::sampling()`

Details

For a given arm of an internal trial (e.g., the control arm or an active treatment arm) of size N_I, suppose the response data are time to event such that $Y_i \sim \text{Weibull}(\alpha, \sigma)$, where

$$f(y_i \mid \alpha, \sigma) = \left(\frac{\alpha}{\sigma}\right)\left(\frac{y_i}{\sigma}\right)^{\alpha-1} \exp\left(-\left(\frac{y_i}{\sigma}\right)^{\alpha}\right),$$

$i = 1, \ldots, N_I$. Define $\boldsymbol{\theta} = \{\log(\alpha), \beta\}$ where $\beta = -\log(\sigma)$ is the intercept parameter of a Weibull proportional hazards regression model. The posterior distribution for $\boldsymbol{\theta}$ is written as

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{\nu}) \propto \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{\nu})\; \pi(\boldsymbol{\theta}),$$

where $\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{\nu}) = \prod_{i=1}^{N_I} f(y_i \mid \boldsymbol{\theta})^{\nu_i} S(y_i \mid \boldsymbol{\theta})^{1-\nu_i}$ is the likelihood of the response data from the internal arm with event indicator $\boldsymbol{\nu}$ and survival function $S(y_i \mid \boldsymbol{\theta}) = 1 - F(y_i \mid \boldsymbol{\theta})$, and $\pi(\boldsymbol{\theta})$ is a prior distribution on $\boldsymbol{\theta}$ (either a multivariate normal distribution or a mixture of two multivariate normal distributions). Note that the posterior distribution for $\boldsymbol{\theta}$ does not have a closed form, and thus MCMC samples for $\log(\alpha)$ and β are drawn from the posterior. These MCMC samples are used to construct samples from the posterior distribution for the probability of surviving past the specified analysis time(s) for the given arm.

To construct a posterior distribution for the treatment difference (i.e., the difference in survival probabilities at the specified analysis time), obtain MCMC samples from the posterior distributions for the survival probabilities under each arm and then subtract the control-arm samples from the treatment-arm samples.

Value

stan posterior object

Examples

```
library(distributional)
library(dplyr)
library(rstan)
mvn_prior <- dist_multivariate_normal(
   mu = list(c(0.3, -2.6)),
   sigma = list(matrix(c(1.5, 0.3, 0.3, 1.1), nrow = 2)))
post_treated <- calc_post_weibull(filter(int_tte_df, trt == 1),
                                  response = y,
                                  event = event,
                                  prior = mvn_prior,
                                  analysis_time = 12,
                                  warmup = 5000,
                                  iter = 15000)
# Extract MCMC samples of survival probabilities at time t=12
surv_prob_treated <- as.data.frame(extract(post_treated,
                                   pars = c("survProb")))[,1]
```

`calc_power_prior_beta` *Calculate Power Prior Beta*

Description

Calculate a (potentially inverse probability weighted) beta power prior for the control response rate using external control data.

Usage

```
calc_power_prior_beta(external_data, response, prior)
```

Arguments

`external_data`	This can either be a `prop_scr_obj` created by calling `create_prop_scr()` or a tibble of the external data. If it is just a tibble the weights will be assumed to be 1.
`response`	Name of response variable
`prior`	A beta distributional object that is the initial prior for the control response rate before the external control data are observed

Details

Weighted participant-level response data from the control arm of an external study are incorporated into an inverse probability weighted (IPW) power prior for the control response rate θ_C. When borrowing information from an external control arm of size N_{EC}, the components of the IPW power prior for θ_C are defined as follows:

Initial prior:

$$\theta_C \sim \text{Beta}(\nu_0, \phi_0)$$

IPW likelihood of the external response data $\boldsymbol{y}_E$ with weights $\hat{\boldsymbol{a}}_0$:

$$\mathcal{L}_E(\theta_C \mid \boldsymbol{y}_E, \hat{\boldsymbol{a}}_0) \propto \exp\left(\sum_{i=1}^{N_{EC}} \hat{a}_{0i} \left[y_i \log(\theta_C) + (1 - y_i)\log(1 - \theta_C)\right]\right)$$

IPW power prior:

$$\theta_C \mid \boldsymbol{y}_E, \hat{\boldsymbol{a}}_0 \sim \text{Beta}\left(\sum_{i=1}^{N_{EC}} \hat{a}_{0i} y_i + \nu_0, \sum_{i=1}^{N_{EC}} \hat{a}_{0i}(1 - y_i) + \phi_0\right)$$

Defining the weights $\hat{\boldsymbol{a}}_0$ to equal 1 results in a conventional beta power prior.

Value

Beta power prior object

See Also

Other power prior: `calc_power_prior_norm()`, `calc_power_prior_weibull()`

Examples

```
library(distributional)
library(dplyr)
# This function can be used directly on the data
calc_power_prior_beta(external_data = ex_binary_df,
                      response = y,
                      prior = dist_beta(0.5, 0.5))

# Or this function can be used with a propensity score object
ps_obj <- calc_prop_scr(internal_df = filter(int_binary_df, trt == 0),
                        external_df = ex_binary_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)

calc_power_prior_beta(ps_obj,
                      response = y,
                      prior = dist_beta(0.5, 0.5))
```

`calc_power_prior_norm` *Calculate Power Prior Normal*

Description

Calculate a (potentially inverse probability weighted) normal power prior using external data.

Usage

```
calc_power_prior_norm(
  external_data,
  response,
  prior = NULL,
  external_sd = NULL
)
```

Arguments

`external_data`	This can either be a `prop_scr_obj` created by calling `create_prop_scr()` or a tibble of the external data. If it is just a tibble the weights will be assumed to be 1. Only the external data for the arm(s) of interest should be included in this object (e.g., external control data if creating a power prior for the control mean)
`response`	Name of response variable
`prior`	Either `NULL` or a normal distributional object that is the initial prior for the parameter of interest (e.g., control mean) before the external data are observed
`external_sd`	Standard deviation of external response data if assumed known. It can be left as `NULL` if assumed unknown

Details

Weighted participant-level response data from an external study are incorporated into an inverse probability weighted (IPW) power prior for the parameter of interest θ (e.g., the control mean if borrowing from an external control arm). When borrowing information from an external dataset of size N_E, the IPW likelihood of the external response data $\boldsymbol{y}_E$ with weights $\hat{\boldsymbol{a}}_0$ is defined as

$$\mathcal{L}_E(\theta \mid \boldsymbol{y}_E, \hat{\boldsymbol{a}}_0, \sigma_E^2) \propto \exp\left(-\frac{1}{2\sigma_E^2}\sum_{i=1}^{N_E}\hat{a}_{0i}(y_i-\theta)^2\right).$$

The `prior` argument should be either a distributional object with a family type of `normal` or `NULL`, corresponding to the use of a normal initial prior or an improper uniform initial prior (i.e., $\pi(\theta) \propto 1$), respectively.

The `external_sd` argument can be a positive value if the external standard deviation is assumed known or left as `NULL` otherwise. If `external_sd = NULL`, then `prior` must be `NULL` to indicate the use of an improper uniform initial prior for θ, and an improper prior is defined for the unknown external standard deviation such that $\pi(\sigma_E^2) \propto (\sigma_E^2)^{-1}$. The details of the IPW power prior for each case are as follows:

`external_sd = positive value` **(σ_E^2 known):** With either a proper normal or an improper uniform initial prior, the IPW weighted power prior for θ is a normal distribution.

`external_sd = NULL` **(σ_E^2 unknown):** With improper priors for both θ and σ_E^2, the marginal IPW weighted power prior for θ after integrating over σ_E^2 is a non-standardized t distribution.

Defining the weights $\hat{\boldsymbol{a}}_0$ to equal 1 results in a conventional normal (or t) power prior if the external standard deviation is known (unknown).

Value

Normal power prior object

See Also

Other power prior: `calc_power_prior_beta()`, `calc_power_prior_weibull()`

Examples

```
library(distributional)
library(dplyr)
# This function can be used directly on the data
calc_power_prior_norm(ex_norm_df,
                      response = y,
                      prior = dist_normal(0.5, 10),
                      external_sd = 0.15)

# Or this function can be used with a propensity score object
ps_obj <- calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                        external_df = ex_norm_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
calc_power_prior_norm(ps_obj,
                      response = y,
                      prior = dist_normal(0.5, 10),
                      external_sd = 0.15)
```

`calc_power_prior_weibull` *Calculate Power Prior Weibull*

Description

Calculate an approximate (potentially inverse probability weighted) multivariate normal power prior for the log-shape and log-inverse-scale parameters of a Weibull likelihood for external time-to-event control data.

Usage

```
calc_power_prior_weibull(
  external_data,
  response,
  event,
  intercept,
  shape,
  approximation = c("Laplace", "MCMC"),
  ...
)
```

Arguments

`external_data`	This can either be a `prop_scr_obj` created by calling `create_prop_scr()` or a tibble of the external data. If it is just a tibble the weights will be assumed to be 1. Only the external data for the arm(s) of interest should be included in this object (e.g., external control data if creating a power prior for the control Weibull shape and intercept parameters)
`response`	Name of response variable
`event`	Name of event indicator variable (1: event; 0: censored)
`intercept`	Normal distributional object that is the initial prior for the intercept (i.e., log-inverse-scale) parameter
`shape`	Integer value that is the scale of the half-normal prior for the shape parameter
`approximation`	Type of approximation to be used. Either `Laplace` or `MCMC`. `Laplace` is used by default because it is faster than `MCMC`.
`...`	Arguments passed to `rstan::sampling` (e.g. iter, chains).

Details

Weighted participant-level response data from the control arm of an external study are incorporated into an approximated inverse probability weighted (IPW) power prior for the parameter vector $\boldsymbol{\theta}_C = \{\log(\alpha), \beta\}$, where $\beta = -\log(\sigma)$ is the intercept parameter of a Weibull proportional hazards regression model and α and σ are the Weibull shape and scale parameters, respectively. When borrowing information from an external dataset of size N_E, the IPW likelihood of the external response data $\boldsymbol{y}_E$ with event indicators $\boldsymbol{\nu}_E$ and weights $\hat{\boldsymbol{a}}_0$ is defined as

$$\mathcal{L}_E(\alpha, \sigma \mid \boldsymbol{y}_E, \boldsymbol{\nu}_E, \hat{\boldsymbol{a}}_0) \propto \prod_{i=1}^{N_E} \left\{ \left(\frac{\alpha}{\sigma}\right) \left(\frac{y_i}{\sigma}\right)^{\alpha-1} \exp\left(-\left(\frac{y_i}{\sigma}\right)^{\alpha}\right) \right\}^{\hat{a}_{0i}\nu_i} \left\{ \exp\left(-\left(\frac{y_i}{\sigma}\right)^{\alpha}\right) \right\}^{\hat{a}_{0i}(1-\nu_i)}.$$

The initial priors for the intercept parameter β and the shape parameter α are assumed to be normal and half-normal, respectively, which can be defined using the `intercept` and `shape` arguments.

The power prior for $\boldsymbol{\theta}_C$ does not have a closed form, and thus we approximate it via a bivariate normal distribution; i.e.,

$$\boldsymbol{\theta}_C \mid \boldsymbol{y}_E, \boldsymbol{\nu}_E, \hat{\boldsymbol{a}}_0 \mathrel{\dot\sim} \text{MVN}\left(\tilde{\boldsymbol{\mu}}_0, \tilde{\boldsymbol{\Sigma}}_0\right).$$

If `approximation = Laplace`, then $\tilde{\boldsymbol{\mu}}_0$ is the mode vector of the IPW power prior and $\tilde{\boldsymbol{\Sigma}}_0$ is the negative inverse of the Hessian of the log IPW power prior evaluated at the mode. If `approximation = MCMC`, then MCMC samples are obtained from the IPW power prior, and $\tilde{\boldsymbol{\mu}}_0$ and $\tilde{\boldsymbol{\Sigma}}_0$ are the estimated mean vector and covariance matrix of these MCMC samples. Note that the Laplace approximation method is faster due to its use of optimization instead of MCMC sampling.

The first element of the mean vector and the first row/column of covariance matrix correspond to the log-shape parameter ($\log(\alpha)$), and the second element corresponds to the intercept (β, the log-inverse-scale) parameter.

Value

Multivariate Normal Distributional Object

See Also

Other power prior: `calc_power_prior_beta()`, `calc_power_prior_norm()`

Examples

```
library(distributional)
library(dplyr)
# This function can be used directly on the data
calc_power_prior_weibull(ex_tte_df,
                         response = y,
                         event = event,
                         intercept = dist_normal(0, 10),
                         shape = 50,
                         approximation = "Laplace")

# Or this function can be used with a propensity score object
ps_obj <- calc_prop_scr(internal_df = filter(int_tte_df, trt == 0),
                        external_df = ex_tte_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
calc_power_prior_weibull(ps_obj,
                         response = y,
                         event = event,
                         intercept = dist_normal(0, 10),
                         shape = 50,
                         approximation = "Laplace")
```

`calc_prop_scr` *Create a Propensity Score Object*

Description

Calculate the propensity scores and ATT inverse probability weights for participants from internal and external datasets. Only the relevant treatment arms from each dataset should be read in (e.g., only the control arm from each dataset if creating a hybrid control arm).

Usage

```
calc_prop_scr(internal_df, external_df, id_col, model, ...)
```

Arguments

`internal_df`	Internal dataset with one row per subject and all the variables needed to run the model
`external_df`	External dataset with one row per subject and all the variables needed to run the model

`id_col`	Name of the column in both datasets used to identify each subject. It must be the same across datasets
`model`	Model used to calculate propensity scores
`...`	Optional arguments

Details

For the subset of participants in both the external and internal studies for which we want to balance the covariate distributions (e.g., external control and internal control participants if constructing a hybrid control arm), we define a study-inclusion propensity score for each participant as

$$e(x_i) = P(S_i = 1 \mid x_i),$$

where x_i denotes a vector of baseline covariates for the ith participant and S_i denotes the indicator that the participant is enrolled in the internal trial ($S_i = 1$ if internal, $S_i = 0$ if external). The estimated propensity score $\hat{e}(x_i)$ is obtained using logistic regression.

An ATT inverse probability weight is calculated for each individual as

$$\hat{a}_{0i} = \frac{\hat{e}(x_i)}{\hat{P}(S_i = s_i|x_i)} = s_i + (1 - s_i)\frac{\hat{e}(x_i)}{1 - \hat{e}(x_i)}.$$

In a weighted estimator, data from participants in the external study are given a weight of $\hat{e}(x_i)/(1-\hat{e}(x_i))$ whereas data from participants in the internal trial are given a weight of 1.

Value

`prop_scr_obj` object, with the internal and the external data and the propensity score and inverse probability weight calculated for each subject.

Examples

```
# This can be used for both continuous and binary data
library(dplyr)
# Continuous
calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                       external_df = ex_norm_df,
                       id_col = subjid,
                       model = ~ cov1 + cov2 + cov3 + cov4)
# Binary
calc_prop_scr(internal_df = filter(int_binary_df, trt == 0),
                       external_df = ex_binary_df,
                       id_col = subjid,
                       model = ~ cov1 + cov2 + cov3 + cov4)
```

calc_study_duration — *Calculate the Analysis Time Based on a Target Number of Events and/or Target Follow-up Time*

Description

Calculate the Analysis Time Based on a Target Number of Events and/or Target Follow-up Time

Usage

```
calc_study_duration(
  study_time,
  observed_time,
  event_indicator,
  target_events = NULL,
  target_follow_up = NULL
)
```

Arguments

`study_time`	Vector of study times (accrual time + observed time)
`observed_time`	Vector of observed times (event time or censoring time)
`event_indicator`	Vector of boolean values (TRUE/FALSE or 1/0) indicating if the observed time value is an event or censoring time
`target_events`	Target number of events, where the analysis time is determined once this number of events is reached. Default is `NULL`, in which case `target_follow_up` must be specified.
`target_follow_up`	Target follow-up for each participant, where the analysis time is determined once each participant in the risk set is followed up for this amount of time (i.e., minimum follow-up time). Default is `NULL`, in which case `target_events` must be specified.

Details

This function calculates the analysis time for a study with a time-to-event endpoint for which the target number of events (`target_events`) and/or target follow-up time (`target_follow_up`) are specified. If only `target_events` is specified, the analysis will occur at the time when the target number of events has been reached. If only `target_follow_up` is specified, the analysis will occur once the last-enrolled participant who is still in the risk set has been followed up for this amount of time. If both `target_events` and `target_follow_up` are specified, the analysis time will be based on whichever occurs first.

Value

Time of analysis

Examples

```
library(dplyr)

# Determining analysis time by reaching a target number of events
ex_tte_df |> mutate(
  analysis_time = calc_study_duration(study_time = total_time, observed_time = y,
                                      event_indicator = event, target_events = 30)
)

# Determining analysis time by a target follow-up time
ex_tte_df |> mutate(
  analysis_time = calc_study_duration(study_time = total_time, observed_time = y,
                                      event_indicator = event, target_follow_up = 12)
)

# Or use both (whichever happens first)
ex_tte_df |> mutate(
  analysis_time = calc_study_duration(study_time = total_time, observed_time = y,
                                      event_indicator = event,
                                      target_events = 30, target_follow_up = 12)
)
```

`ex_binary_df` *External Binary Control Data for Propensity Score Balancing*

Description

This is a simulated dataset used to illustrate Bayesian dynamic borrowing in the case when borrowing from an external control arm with a binary endpoint, where the baseline covariate distributions of the internal and external data are balanced via inverse probability weighting.

Usage

```
ex_binary_df
```

Format

`ex_binary_df`**:**

A data frame with 150 rows and 6 columns:

subjid Unique subject ID

cov1 Covariate 1, which is normally distributed around 65 with a SD of 10

cov2 Covariate 2, which is binary (0 vs. 1) with about 30% of participants having level 1

cov3 Covariate 3, which is binary (0 vs. 1) with about 40% of participants having level 1

cov4 Covariate 4, which is binary (0 vs. 1) with about 50% of participants having level 1

y Response, which is binary (0 vs. 1) `ex_norm_df` *External Normal Control Data for Propensity Score Balancing*

Description

This is a simulated dataset used to illustrate Bayesian dynamic borrowing in the case when borrowing from an external control arm with a normal endpoint, where the baseline covariate distributions of the internal and external data are balanced via inverse probability weighting.

Usage

```
ex_norm_df
```

Format

`ex_norm_df`:

A data frame with 150 rows and 6 columns:

subjid Unique subject ID

cov1 Covariate 1, which is normally distributed around 50 with a SD of 10

cov2 Covariate 2, which is binary (0 vs. 1) with about 20% of participants having level 1

cov3 Covariate 3, which is binary (0 vs. 1) with about 60% of participants having level 1

cov4 Covariate 4, which is binary (0 vs. 1) with about 30% of participants having level 1

y Response, which is normally distributed with a SD of 0.15

`ex_tte_df` *External Time-to-Event Control Data for Propensity Score Balancing*

Description

This is a simulated dataset used to illustrate Bayesian dynamic borrowing in the case when borrowing from an external control arm with a time-to-event endpoint, where the baseline covariate distributions of the internal and external data are balanced via inverse probability weighting.

Usage

```
ex_tte_df
```

Format

`ex_tte_df`:

A data frame with 150 rows and 9 columns:

subjid Unique subject ID

y Response (observed time at which the participant either had an event or was censored)

enr_time Enrollment time

total_time Time from study start

event Event indicator (1: event; 0: censored)

cov1 Covariate 1, which is normally distributed around 65 with a SD of 10

cov2 Covariate 2, which is binary (0 vs. 1) with about 30% of participants having level 1

cov3 Covariate 3, which is binary (0 vs. 1) with about 40% of participants having level 1

cov4 Covariate 4, which is binary (0 vs. 1) with about 50% of participants having level 1

`int_binary_df` *Internal Binary Data for Propensity Score Balancing*

Description

This is a simulated dataset used to illustrate Bayesian dynamic borrowing in the case when borrowing from an external control arm with a binary endpoint, where the baseline covariate distributions of the internal and external data are balanced via inverse probability weighting.

Usage

```
int_binary_df
```

Format

`int_binary_df`:

A data frame with 160 rows and 7 columns:

subjid Unique subject ID

cov1 Covariate 1, which is normally distributed around 62 with an sd of 8

cov2 Covariate 2, which is binary (0 vs. 1) with about 40% of participants having level 1

cov3 Covariate 3, which is binary (0 vs. 1) with about 40% of participants having level 1

cov4 Covariate 4, which is binary (0 vs. 1) with about 60% of participants having level 1

trt Treatment indicator, where 0 = control and 1 = active treatment

y Response, which is binary (0 vs. 1)

`int_norm_df` *Internal Normal Data for Propensity Score Balancing*

Description

This is a simulated dataset used to illustrate Bayesian dynamic borrowing in the case when borrowing from an external control arm with a normal endpoint, where the baseline covariate distributions of the internal and external data are balanced via inverse probability weighting.

Usage

```
int_norm_df
```

Format

`int_norm_df`**:**

A data frame with 120 rows and 7 columns:

subjid Unique subject ID

cov1 Covariate 1, which is normally distributed around 55 with a SD of 8

cov2 Covariate 2, which is binary (0 vs. 1) with about 30% of participants having level 1

cov3 Covariate 3, which is binary (0 vs. 1) with about 50% of participants having level 1

cov4 Covariate 4, which is binary (0 vs. 1) with about 30% of participants having level 1

trt Treatment indicator, where 0 = control and 1 = active treatment

y Response, which is normally distributed with a SD of 0.15

`int_tte_df` *Internal Time-to-Event Control Data for Propensity Score Balancing*

Description

This is a simulated dataset used to illustrate Bayesian dynamic borrowing in the case when borrowing from an external control arm with a time-to-event endpoint, where the baseline covariate distributions of the internal and external data are balanced via inverse probability weighting.

Usage

```
int_tte_df
```

Format

`int_tte_df`**:**

A data frame with 160 rows and 10 columns:

subjid Unique subject ID

y Response (observed time at which the participant either had an event or was censored)

enr_time Enrollment time

total_time Time from study start

event Event indicator (1: event; 0: censored)

trt Treatment indicator, where 0 = control and 1 = active treatment

cov1 Covariate 1, which is normally distributed around 62 with a SD of 8

cov2 Covariate 2, which is binary (0 vs. 1) with about 40% of participants having level 1

cov3 Covariate 3, which is binary (0 vs. 1) with about 40% of participants having level 1

cov4 Covariate 4, which is binary (0 vs. 1) with about 60% of participants having level 1

`inv_logit` *Inverse Logit Function*

Description

Inverse Logit Function

Usage

```
inv_logit(x)
```

Arguments

`x`	Real number(s) to take the inverse logit of

Details

This function is a short hand to $\exp(x)/(1+\exp(x))$.

Value

Vector of probabilities between 0 and 1

Examples

```
inv_logit(0.5)
```

`is_prop_scr` *Test If Propensity Score Object*

Description

Test If Propensity Score Object

Usage

```
is_prop_scr(x)
```

Arguments

`x`	Object to test

Value

Boolean

Examples

```
library(dplyr)
x <- calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                   external_df = ex_norm_df,
                   id_col = subjid,
                   model = ~ cov1 + cov2 + cov3 + cov4)
is_prop_scr(x)
```

`mix_means` *Extract Means of Mixture Components*

Description

Extract Means of Mixture Components

Usage

```
mix_means(x)
```

Arguments

`x`	A mixture distributional object

Details

If a distributional object that is a mixture of two or more normal distributions is read in, the function will return a numeric object with the means of each normal component. If the distributional object is a mixture of two or more multivariate normal distributions, the function will return a list with the mean vectors of each multivariate normal component.

Value

numeric or list object

Examples

```
library(distributional)
mix_norm <- dist_mixture(comp1 = dist_normal(1, 10),
                         comp2 = dist_normal(1.5, 12),
                         weights = c(.5, .5))
mix_means(mix_norm)
```

`mix_sigmas` *Extract Standard Deviations of Mixture Components*

Description

Extract Standard Deviations of Mixture Components

Usage

```
mix_sigmas(x)
```

Arguments

`x`	A mixture distributional object

Details

If a distributional object that is a mixture of two or more normal distributions is read in, the function will return a numeric object with the standard deviations of each normal component. If the distributional object is a mixture of two or more multivariate normal distributions, the function will return a list with the covariance matrices of each multivariate normal component.

Value

numeric or list object

Examples

```
library(distributional)
mix_norm <- dist_mixture(comp1 = dist_normal(1, 10),
                         comp2 = dist_normal(1.5, 12),
                         weights = c(.5, .5))
mix_sigmas(mix_norm)
```

---

plot_dist *Plot Distribution*

Description

Plot Distribution

Usage

```
plot_dist(...)
```

Arguments

`...`	Distributional object(s) to plot. When passing multiple objects naming them will change the labels in the plot, else they will use the distributional format

Value

ggplot object that is the density of the provided distribution

Examples

```
library(distributional)
plot_dist(dist_normal(0, 1))
plot_dist(dist_multivariate_normal(mu = list(c(1, 2)), sigma = list(matrix(c(4, 2, 2, 3), ncol=2))))
#Plotting Multiple
plot_dist(dist_normal(0, 1), dist_normal(10, 5))
plot_dist('Prior' = dist_normal(0, 1), 'Posterior' = dist_normal(10, 5))
```

---

prop_scr_cloud *Propensity Score Cloud Plot*

Description

Propensity Score Cloud Plot

Usage

```
prop_scr_cloud(x, trimmed_prop_scr = NULL)
```

Arguments

`x`	A `prop_scr` object
`trimmed_prop_scr`	A trimmed `prop_scr` object

Value

ggplot object

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                        external_df = ex_norm_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
ps_obj_trimmed <- trim_ps(ps_obj, low = 0.1, high = 0.6)
# Plotting the Propensity Scores
prop_scr_cloud(ps_obj, trimmed_prop_scr = ps_obj_trimmed)
```

`prop_scr_dens` *Density of the Propensity Score Object*

Description

Plot overlapping density curves of the propensity scores for both the internal and external participants, or plot external IPWs.

Usage

```
prop_scr_dens(
  x,
  variable = c("propensity score", "ps", "inverse probability weight", "ipw"),
  ...
)
```

Arguments

`x`	Propensity score object
`variable`	Variable to plot. It must be either a propensity score ("ps" or "propensity score") or inverse probability weight ("ipw" or "inverse probability weight")
`...`	Optional arguments for `geom_density`

Value

ggplot object

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                        external_df = ex_norm_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
# Plotting the Propensity Scores
prop_scr_dens(ps_obj)
# Or plotting the inverse probability weights
prop_scr_dens(ps_obj, variable = "ipw")
```

`prop_scr_hist` *Histogram of the Propensity Score Object*

Description

Plot overlapping histograms of the propensity scores for both the internal and external participants, or plot external IPWs.

Usage

```
prop_scr_hist(
  x,
  variable = c("propensity score", "ps", "inverse probability weight", "ipw"),
  ...
)
```

Arguments

`x`	Propensity score object
`variable`	Variable to plot. It must be either a propensity score ("ps" or "propensity score") or inverse probability weight ("ipw" or "inverse probability weight")
`...`	Optional arguments for `geom_histogram`

Value

ggplot object

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                        external_df = ex_norm_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
# Plotting the Propensity Scores
prop_scr_hist(ps_obj)
```

```
# Or plotting the inverse probability weights
prop_scr_hist(ps_obj, variable = "ipw")
```

prop_scr_love *Love Plot of the Absolute Standardized Mean Differences*

Description

Plot the unadjusted and IPW-adjusted absolute standardized mean differences for each covariate.

Usage

```
prop_scr_love(x, reference_line = NULL, ...)
```

Arguments

`x`	Propensity score object
`reference_line`	Numeric value of where along the x-axis the vertical reference line should be placed
`...`	Optional options for `geom_point`

Value

ggplot object

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_norm_df, trt == 0),
                       external_df = ex_norm_df,
                       id_col = subjid,
                       model = ~ cov1 + cov2 + cov3 + cov4)
# Plotting the Propensity Scores
prop_scr_love(ps_obj, reference_line = 0.1)
```

`rescale_ps` *Rescale a* `prop_scr` *object*

Description

Rescale a `prop_scr` object

Usage

```
rescale_ps(x, n = NULL, scale_factor = NULL)
```

Arguments

`x`	a `prop_scr` obj
`n`	Desired sample size that the external data should effectively contribute to the analysis of the internal trial data. This will be used to scale the external weights if `scale_factor` is not specified
`scale_factor`	Value to multiple all weights by. This will be used to scale the external weights if `n` is not specified

Value

a `prop_scr` object with rescaled weights

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_binary_df, trt == 0),
                        external_df = ex_binary_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
# weights in a propensity score object can be rescaled to achieve a desired
# effective sample size (i.e., sum of weights)
rescale_ps(ps_obj, n = 75)

# Or by a predetermined factor
rescale_ps(ps_obj, scale_factor = 1.5)
```

`robustify_mvnorm` *Robustify Multivariate Normal Distributions*

Description

Adds vague normal component, where the level of vagueness is controlled by the `n` parameter

Usage

```
robustify_mvnorm(prior, n, weights = c(0.5, 0.5))
```

Arguments

`prior`	Multivariate Normal distributional object
`n`	Number of theoretical participants (or events, for time-to-event data)
`weights`	Vector of weights, where the first number corresponds to the informative component and the second is the vague

Details

In cases with a time-to-event endpoint, a robust mixture prior can be created by adding a vague multivariate normal component to any multivariate normal prior with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The vague component is calculated to have the same mean vector $\boldsymbol{\mu}$ and covariance matrix equal to $\boldsymbol{\Sigma} \times n$, where `n` is the specified number of theoretical events.

Value

mixture distribution

Examples

```
library(distributional)
robustify_mvnorm(
      dist_multivariate_normal(mu = list(c(1, 0)), sigma = list(c(10, 5))),
       n = 15)
```

`robustify_norm` *Robustify Normal Distributions*

Description

Adds vague normal component, where the level of vagueness is controlled by the `n` parameter

Usage

```
robustify_norm(prior, n, weights = c(0.5, 0.5))
```

Arguments

`prior`	Normal or Multivariate Normal distributional object
`n`	Number of theoretical participants (or events, for time-to-event data)
`weights`	Vector of weights, where the first number corresponds to the informative component and the second is the vague

Details

In cases with a normal endpoint, a robust mixture prior can be created by adding a vague normal component to any normal prior with mean θ and variance σ^2.The vague component is calculated to have the same mean θ and variance equal to $\sigma^2 \times n$, where `n` is the specified number of theoretical participants. If robustifying a normal power prior that was calculated from external control data and `n` is defined as the number of external control participants, and the vague component would then correspond to one external control participant's worth of data.

Value

mixture distribution

Examples

```
library(distributional)
robustify_norm(dist_normal(0,1), n = 15)
```

`sim_accrual` *Simulate Participant Accrual Times*

Description

Simulate Participant Accrual Times

Usage

```
sim_accrual(n, accrual_periods, accrual_props)
```

Arguments

`n`	Number of participants
`accrual_periods`	Vector of right endpoints defining the time periods of accrual, e.g., c(6,8) defines 0<=x<6, 6<=x<8.
`accrual_props`	Vector indicating the proportion of participants that are expected to be enrolled during each of the defined accrual periods. Should sum to 1, otherwise these proportions will be normalized.

Details

Simulate the accrual times for each participant, where `accrual_periods` defines the right time points for each accrual period (with the last element corresponding to the end of accrual), and `accrual_props` defines the proportion of study participants who are enrolled during each of these periods. The simulated accrual times for participants within a given accrual period are assumed to be uniformly distributed.

Value

Vector of accrual times corresponding to when each participant enters the study

Examples

```
at <- sim_accrual(n = 100000, accrual_periods = c(6, 8), accrual_props = c(.5, .5))
hist(at, breaks = 100, main = "Histogram of Enrollment Times", xlab = "Enrollment Time")
```

`sim_pw_const_haz` *Simulate Event Times for Each Individual from a Piecewise Constant Hazard Model*

Description

Simulate Event Times for Each Individual from a Piecewise Constant Hazard Model

Usage

```
sim_pw_const_haz(n, hazard_periods = NULL, hazard_values)
```

Arguments

`n`	Number of individuals
`hazard_periods`	Vector of break points between time periods with separate constant hazards, e.g., c(6,8) defines [0,6), [6,8), [8, infinity). Leave as NULL if defining only one hazard period.
`hazard_values`	Vector of constant hazard values associated with the time intervals

Details

Simulate the event or censoring times for each participant using a piecewise constant hazard model where each time period is defined to have a different constant hazard. Here, `hazard_periods` defines the right time points for each time period (with the exception of the last time period which extends to infinity), and `hazard_values` defines the constant hazards for each time period.

Value

Vector of simulated times from the time-to-event distribution

Examples

```
tte_dat <- sim_pw_const_haz(n = 100000, hazard_periods = c(6, 8), hazard_values = c(0.1, 0.1, 0.1))
hist(tte_dat, breaks = 100, main = "Event Time Distribution", xlab = "Event Time")
```

`sim_weib_ph` *Simulate Event Times for Each Participant from a Weibull Proportional Hazards Regression Model*

Description

Simulate Event Times for Each Participant from a Weibull Proportional Hazards Regression Model

Usage

```
sim_weib_ph(weibull_ph_mod, samp_df, cond_drift = 0, cond_trt_effect = 0)
```

Arguments

`weibull_ph_mod`	`survreg` object corresponding to a Weibull proportional hazards model fit using the external data
`samp_df`	Data frame of covariates corresponding to the sample arm (control or treated) for which event times should be simulated. The column names should correspond to the covariate names in the `survreg` object.
`cond_drift`	Optional value of the conditional drift by which the intercept in the Weibull proportional hazards regression model should be increased/decreased to incorporate the impact of unmeasurable sources of drift. Default is 0.
`cond_trt_effect`	Optional value of the conditional treatment effect by which the intercept in the Weibull proportional hazards regression model should be increased/decreased if simulating event data for a treated arm. Default is 0.

Details

Simulate the event times for each participant using a Weibull proportional hazards (PH) regression model. The "true" parameter values for the Weibull shape α and the regression coefficients $\boldsymbol{\beta}$ are assumed to be equal to the parameter estimates from a `survreg` object (`weibull_ph_mod`) fit using external data (note that the Weibull shape parameter α is defined as the inverse of the scale parameter reported by `survreg`).

For participant i, let y_i denote the time-to-event random variable and $\boldsymbol{x}_i = \{x_{i,1}, \ldots, x_{i,p}\}$ the vector of p covariates (row i of `samp_df`) that correspond to the $(p+1)$-dimensional vector of regression coefficients $\boldsymbol{\beta}$. The density function of the Weibull PH regression model is

$$f(y_i \mid \boldsymbol{x}_i, \alpha, \boldsymbol{\beta}, \delta, \gamma) = \left(\frac{\alpha}{\sigma_i}\right)\left(\frac{y_i}{\sigma_i}\right)^{\alpha-1} \exp\left(-\left(\frac{y_i}{\sigma_i}\right)^{\alpha}\right),$$

where $-\log(\sigma_i) = \beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p} + \delta + \gamma$. Here, δ and γ denote the conditional drift (`cond_drift`) and conditional treatment effect (`cond_trt_effect`), respectively, that can be calculated using `calc_cond_weibull()` for desired values of the marginal drift and marginal treatment effect.

Value

Vector of simulated event times from a Weibull proportional hazards regression model

Examples

```
library(dplyr)
library(survival)
# Model "true" regression coefficients and shape parameter using the external data
weibull_ph_mod <- survreg(Surv(y, event) ~ cov1 + cov2 + cov3 + cov4, data = ex_tte_df,
                          dist = "weibull")

# Sample covariates for internal control arm via bootstrap from external data
samp_int_ctrl <- bootstrap_cov(ex_tte_df, n = 100) |>
  select(c(cov1, cov2, cov3, cov4))     # keep only covariate columns
tte_dat <- sim_weib_ph(weibull_ph_mod, samp_df = samp_int_ctrl)
```

`sweet_spot_plot` *Create Sweet Spot Plots for Multiple Simulation Scenarios*

Description

Create visualization plots to help identify the "sweet spot" in borrowing strategies across different simulation scenarios. For each unique scenario defined by the combination of variables in `scenario_vars`, the function produces a plot showing power, type I error, and the distribution of the design prior for the control marginal parameter for approaches with and without borrowing.

Usage

```
sweet_spot_plot(
  .data,
  scenario_vars,
  trt_diff,
  control_marg_param,
  h0_prob,
  h0_prob_no_borrowing,
  design_prior = NULL,
  highlight = TRUE
)
```

Arguments

`.data`	A data frame containing iteration-level simulation results.
`scenario_vars`	A vector of quoted column names corresponding to variables used to define unique simulation scenarios. Each unique combination of values in these columns will generate a separate plot.
`trt_diff`	An unquoted column name representing the treatment difference. Used to identify scenarios with null effect (trt_diff = 0) for type I error calculation.
`control_marg_param`	An unquoted column name to be used as the x-axis in the plots. This is typically the control endpoint of interest on the marginal scale (e.g., control response rate).
`h0_prob`	An unquoted column name containing the probability of rejecting the null hypothesis when when borrowing external data.
`h0_prob_no_borrowing`	An unquoted column name containing the probability of rejecting the null hypothesis when not borrowing external data.
`design_prior`	An unquoted column name containing distributional objects that represent the design prior distribution for the control marginal parameter (e.g., posterior distribution using the external control data). Used to aid visualization of which values of the control marginal parameter are assumed to be plausible. Default is `NULL`, in which case no design prior is plotted. See Details for more information.
`highlight`	Logical value to indicate if you want sweet spot highlighting or not. If `TRUE` the sweet spot (where borrowing increase power and reduces type 1 error) will be highlighted.

Details

The function calculates power and type I error rates for BDB approaches that borrow from external data (e.g., use of a robust mixture prior with positive weight on the informative component) and an approach that does not borrow from external data (e.g., use of a vague prior) under each scenario and visualizes them together as a function of the underlying control marginal parameter of interest (e.g., control response rate for binary outcomes) that may vary as a result of drift. This helps identify the "sweet spot" where borrowing results in higher power and lower type I error rates compared to not borrowing. Type I error is calculated using scenarios where `trt_diff` equals 0, and power is calculated for all scenarios with positive values of `trt_diff`.

If `design_prior` is non-`NULL`, the design prior distribution is included in the plot to provide insight into which values of the control marginal parameter are plausible given this assumed design prior. We note that `design_prior` can represent any informative prior that potentially incorporates the external control data (e.g., the posterior distribution of the control marginal parameter constructed using the external data and a vague prior). Each element of the vector corresponding to `design_prior` must be a distributional object with a family equal to "beta", "normal", or "mixture" (where each component is either "beta" or "normal"). For the time-to-event case in which a multivariate normal prior is assumed for the control log-shape and intercept of a Weibull proportional hazards model, this distribution must first be translated into a univariate beta design prior for the control survival probability at some prespecified time. This approximation can be done using `approx_mvn_at_time()`. If the design priors in the vector indicated by `design_prior` differ across iterations within a given scenario (e.g., using the IPW power prior as the iteration-specific design prior), then the average distribution will be plotted (i.e., a distribution of the same family with the hyperparameters averaged across iterations).

Value

A list of ggplot objects, one for each unique scenario defined by `scenario_vars`. Each plot shows:

- Power curves for the cases with and without borrowing
- Type I error rates for the cases with and without borrowing
- Distribution of the design prior (if `design_prior` is specified)

References

Best, N., Ajimi, M., Neuenschwander, B., Saint-Hilary, G., & Wandel, S. (2024). Beyond the Classical Type I Error: Bayesian Metrics for Bayesian Designs Using Informative Priors. *Statistics in Biopharmaceutical Research*, 17(2), 183–196. doi:10.1080/19466315.2024.2342817

Examples

```
library(dplyr)
# Assuming binary_sim_df is a data frame with simulation results in the shape
# of binary template code
plots <- sweet_spot_plot(
  .data = binary_sim_df,
  scenario_vars = c("population", "marg_trt_eff"),
  trt_diff = marg_trt_eff,
  control_marg_param = true_control_RR,
  h0_prob = reject_H0_yes,
  h0_prob_no_borrowing = no_borrowing_reject_H0_yes,
  design_prior = pwr_prior
)

# Display the first plot
plots[[1]]

tte_plots <- tte_sim_df |>
 mutate(beta_appox = approx_mvn_at_time(mix_prior, time = 12)) |>
 sweet_spot_plot(
   scenario_vars = c("population", "marg_trt_eff"),
   trt_diff = marg_trt_eff,
   control_marg_param = true_control_surv_prob,
   h0_prob = reject_H0_yes,
   h0_prob_no_borrowing = no_borrowing_reject_H0_yes,
   design_prior = beta_appox
 )

tte_plots[[1]]
```

`tidy.prop_scr` *Tidy a(n) prop_scr object*

Description

Tidy a(n) prop_scr object

Usage

```
## S3 method for class 'prop_scr'
tidy(x, ...)
```

Arguments

`x`	a `prop_scr` obj
`...`	Unused, included for generic consistency only.

Value

A tidy `tibble::tibble()` summarizing the results of the propensity score weighting. The tibble will have the id column of the external data, an `internal` column to indicate all the data is external, a `ps` column with the propensity scores and a `weight` column with the inverse probability weights

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_binary_df, trt == 0),
                       external_df = ex_binary_df,
                       id_col = subjid,
                       model = ~ cov1 + cov2 + cov3 + cov4)
tidy(ps_obj)
```

`trim_ps` *Trim a* `prop_scr` *object*

Description

Trim a `prop_scr` object

Usage

```
trim_ps(x, low = NULL, high = NULL, quantile = FALSE)
```

Arguments

x	A prop_scr object
low	Low cut-off such that all participants with propensity scores less than this value (or quantile if quantile = TRUE) are removed. If left NULL no lower bound will be used
high	High cut-off such that all participants with propensity scores greater than this value (or quantile if quantile = TRUE) are removed. If left NULL no upper bound will be used
quantile	True/False value to determine if the cut-off values are based directly on the propensity scores (false) or their quantiles (true). By default this is false.

Details

This function uses R's default method of quantile calculation (type 7)

Value

a prop_scr object with a trimmed propensity score distribution

Examples

```
library(dplyr)
ps_obj <- calc_prop_scr(internal_df = filter(int_binary_df, trt == 0),
                        external_df = ex_binary_df,
                        id_col = subjid,
                        model = ~ cov1 + cov2 + cov3 + cov4)
trim_ps(ps_obj, low = 0.3, high = 0.7)
```

tte_sim_df *Time-to-Event Simulation Data*

Description

This is an example of output from a simulation study that investigates the operating characteristics of inverse probability weighted Bayesian dynamic borrowing for the case with a time-to-event outcome. This output was generated based on the time-to-event simulation template. For this simulation study, only the degree of covariate imbalance (as indicated by population) and the marginal treatment effect were varied.

Usage

```
tte_sim_df
```

Format

`tte_sim_df` **A data frame with 18 rows and 7 columns::**

population Populations defined by different covariate imbalances

marg_trt_eff Marginal treatment effect

true_control_surv_prop True control survival probability at time t=12 months on the marginal scale

reject_H0_yes Probability of rejecting the null hypothesis in the case with borrowing

no_borrowing_reject_H0_yes Probability of rejecting the null hypothesis without borrowing

pwr_prior Vector of IPW power priors as distributional objects

mix_prior Vector of mixture priors (i.e., the robustified IPW power priors) as distributional objects

Index