Package 'psborrow2'

April 30, 2024

Type Package

Title Bayesian Dynamic Borrowing Analysis and Simulation

Version 0.0.3.4

Description Bayesian dynamic borrowing is an approach to incorporating external data to supplement a randomized, controlled trial analysis in which external data are incorporated in a dynamic way (e.g., based on similarity of outcomes); see Viele 2013 <doi:10.1002/pst.1589> for an overview. This package implements the hierarchical commensurate prior approach to dynamic borrowing as described in Hobbes 2011 <doi:10.1111/j.1541-0420.2011.01564.x>. There are three main functionalities. First, 'psborrow2' provides a user-friendly interface for applying dynamic borrowing on the study results handles the Markov Chain Monte Carlo sampling on behalf of the user. Second, 'psborrow2' provides a simulation framework to compare different borrowing parameters (e.g. full borrowing, no borrowing, dynamic borrowing) and other trial and borrowing characteristics (e.g. sample size, covariates) in a unified way. Third, 'psborrow2' provides a set of functions to generate data for simulation studies, and also allows the user to specify their own data generation process. This package is designed to use the sampling functions from 'cmdstanr' which can be installed from <https://mc-stan.org/r-packages/>.

URL https://github.com/Genentech/psborrow2, https://genentech.github.io/psborrow2/index.html

BugReports https://github.com/Genentech/psborrow2/issues

License Apache License 2.0

Encoding UTF-8

Depends R (>= 4.1.0)

RoxygenNote 7.3.1

Config/testthat/edition 3

Imports checkmate, glue, methods, graphics, posterior, generics, Matrix, mvtnorm, future, simsurv

Suggests cmdstanr, survival, flexsurv, testthat (>= 3.0), usethis (>= 2.1.5), vdiffr, tibble, xml2, knitr, rmarkdown, bayesplot, matrixcalc, WeightIt, MatchIt, BayesPPD, ggsurvfit, gbm, ggplot2, cobalt, table1, gt, gtsummary

Additional_repositories https://mc-stan.org/r-packages/

Language en-US

SystemRequirements cmdstan

Collate 'generics.R' 'prior_class.R' 'covariate_class.R'
'add_covariates.R' 'prior_normal.R' 'treatment_class.R'
'borrowing_class.R' 'outcome_class.R' 'analysis_class.R'
'borrowing_details.R' 'borrowing_full.R'
'borrowing_hierarchical_commensurate.R' 'borrowing_none.R'
'check_data_matrix_has_columns.R' 'cmdstan.R'
'create_analysis_obj.R' 'create_data_matrix.R'
'treatment_details.R' 'sim_treatment_list.R' 'helpers.R'
'outcome_bin_logistic.R' 'prior_exponential.R'
'outcome_surv_weibull_ph.R' 'outcome_surv_exponential.R'
'sim_outcome_list.R' 'sim_borrowing_list.R'
'sim_covariate_list.R' 'sim_data_list.R' 'simulation_class.R'
'create_simulation_obj.R' 'data.R'
'make_analysis_object_list.R' 'make_model_string_data.R'
'make_model_string_functions.R' 'make_model_string_model.R'
'make_model_string_parameters.R'
'make_model_string_transf_params.R' 'mcmc_sample.R'
'mcmc_simulation_result.R' 'prior_half_cauchy.R'
'outcome_cont_normal.R' 'package.R'
'prepare_stan_data_inputs.R' 'prior_bernoulli.R' 'prior_beta.R'
'prior_cauchy.R' 'prior_gamma.R' 'prior_half_normal.R'
'prior_poisson.R' 'sim_covariates.R' 'sim_estimate_bias.R'
'sim_estimate_effect_variance.R' 'sim_estimate_mse.R'
'sim_is_null_effect_covered.R' 'sim_is_true_effect_covered.R'
'sim_samplesize.R' 'simulate_data_baseline.R' 'simulate_data.R'
'simvar_class.R' 'trim_data_matrix.R' 'uniform_prior.R' 'zzz.R'

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Matt Secrest [aut, cre] (<https://orcid.org/0000-0002-0939-4902>),
Isaac Gravestock [aut],
Craig Gower-Page [ctb],
Manoj Khanal [ctb],
Mingyang Shan [ctb],
Kexin Jin [ctb],
Zhi Yang [ctb],
Genentech, Inc. [cph, fnd]

Maintainer Matt Secrest <secrestm@gene.com>

Repository CRAN

Date/Publication 2024-04-30 21:30:02 UTC

R topics documented:

add_covariates *Add Covariates for Model Adjustment*

Description

Specify column names for adjustment variables in model matrix and prior distributions for the model parameters for these covariates (i.e., betas)

Usage

```
add_covariates(covariates, priors)
```

Arguments

covariates	character. Names of columns in the data matrix containing covariates to be adjusted for in the outcome model. Note: the external and treatment flags should not go here.
priors	Either a single object of class Prior specifying the prior distribution to apply to all covariates or a named list of distributions of class Prior, one for each covariate

Value

Object of class Covariates.

Examples

```
add_covariates(
  covariates = c("a", "b"),
  priors = list(
    "a" = prior_normal(0, 1),
    "b" = prior_normal(0, 2)
  )
)
```

Analysis-class Analysis *Class*

Description

A class for defining Analysis details. Objects of class Analysis should not be created directly but by the constructor create_analysis_obj().

Slots

`data_matrix` matrix. The data matrix, including all covariates to be adjusted for, all relevant outcome variables, and treatment arm and external control arm flags.

`covariates` `Covariate`. Object of class `Covariate` as output by the function `covariate_details()`.

`outcome` `Outcome`. Object of class `Outcome` as output by `outcome_surv_exponential()`, `outcome_surv_weibull_ph()`, or `outcome_bin_logistic()`.

`borrowing` `Borrowing`. Object of class `Borrowing` as output by `borrowing_full()`, `borrowing_none()`, or `borrowing_hierarchical_commensurate()`.

`treatment` `Treatment`. Object of class `Treatment` as output by `treatment_details()`.

`model_string` character. The string that contains the full Stan model code to be compiled.

`model` `CmdStanModel`. The compiled Stan model as output by `cmdstanr::cmdstan_model()`

`ready_to_sample` logical. Is the object ready to sample?

---

as_data_frame *Coerce a* psborrow2 *object to a data frame*

Description

Creates `data.frame` objects from various classes in `psborrow2`

Usage

```
## S3 method for class 'BaselineDataList'
as.data.frame(x, ...)
```

Arguments

`x`	object of type: BaselineDataList
`...`	Optional arguments for passed to data.frame

Value

A `data.frame`

BaselineDataFrame-class

Baseline Data Frame Object

Description

Contains a generated baseline dataset for a single arm.

Value

A `BaselineDataFrame`

Slots

`cov_names` `character` contains the names of covariates generated from the multivariate normal distribution

`means` `numeric` contains the means of generating distribution for the covariates in `cov_names`

`variances` `numeric` contains the marginal variances of generating distribution for the covariates in `cov_names`.

BaselineDataList-class

Baseline Data Frame List

Description

A named `list` of `BaselineDataFrames` with generated data for `internal_treated`/`internal_control`/ `external_control` groups

Value

A `BaselineDataList`

Slots

`baseline_object` Simulated covariates definitions as `BaselineObject`. See `create_baseline_object()`

BaselineObject-class BaselineObject *class for data simulation*

Description

BaselineObject class for data simulation

Slots

n_trt_int integer. Number of internal treated patients

n_ctrl_int integer. Number of internal control patients

n_ctrl_ext integer. Number of external control patients

covariates list. List of correlated covariates objects, see baseline_covariates()

transformations list. List of named transformation functions.

baseline_covariates *Specify Correlated Baseline Covariates*

Description

Set parameters to generate correlated multivariate normal data for internal and external patients.

Usage

```
baseline_covariates(
  names,
  means_int,
  means_ext = means_int,
  covariance_int,
  covariance_ext = covariance_int
)
```

Arguments

names	character vector of variable names.
means_int	numeric vector of means for internal patients. Must have same length as names
means_ext	numeric vector of means for external patients. Must have same length as names
covariance_int	variance-covariance matrix for generating multivariate normal for internal patients. Must be square matrix with same number of rows and length(names)
covariance_ext	variance-covariance matrix for generating multivariate normal data for external patients. Must be square matrix with same number of rows and length(names)

Value

BaselineObject to build simulated dataset

Examples

```
corr_covs <- baseline_covariates(
  names = c("b1", "b2"),
  means_int = c(5, 25),
  covariance_int = covariance_matrix(diag = c(1, 1), upper_tri = 0.4)
)
```

---

bernoulli_prior *Legacy function for the bernoulli prior*

Description

Please use `prior_bernoulli()` instead.

Usage

```
bernoulli_prior(...)
```

Arguments

`...`	Deprecated arguments to `bernoulli_prior()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `bernoulli_prior()` is deprecated and that `prior_bernoulli()` should be used instead.

---

beta_prior *Legacy function for the beta prior*

Description

Please use `prior_beta()` instead.

Usage

```
beta_prior(...)
```

Arguments

`...`	Deprecated arguments to `beta_prior()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `beta_prior()` is deprecated and that `prior_beta()` should be used instead.

`BinaryOutcome-class` `BinaryOutcome` *class*

Description

`BinaryOutcome` class

Slots

`function_stan_code` character. Code to include in the Stan functions program block.

`param_stan_code` character. Code to include in the Stan parameters program block.

`likelihood_stan_code` character. Code defining the likelihood to include in the Stan model program block.

`data_stan_code` character. Code to include in the Stan data program block.

`n_param` integer. Number of ancillary parameters for the model to estimate.

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.

`binary_var` character. Variable used for outcome in `BinaryOutcome` objects.

`baseline_prior` `Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome.

`name_beta_trt`. Named vector for beta_trt.

`name_exp_trt`. Named vector for exponentiated beta_trt

`alpha_type`. How to interpret alpha.

`name_addnl_params`. Named vector for additional parameters.

See Also

Other outcome: `ContinuousOutcome-class`, `Outcome-class`, `OutcomeBinaryLogistic-class`, `OutcomeContinuousNormal-class`, `OutcomeSurvExponential-class`, `OutcomeSurvWeibullPH-class`, `TimeToEvent-class`

binary_cutoff *Binary Cut-Off Transformation*

Description

Binary Cut-Off Transformation

Usage

```
binary_cutoff(name, int_cutoff, ext_cutoff)
```

Arguments

name	variable to transform
int_cutoff	cut-off for internal patients, numeric between 0 and 1
ext_cutoff	cut-off for external patients, numeric between 0 and 1

Value

Transformation function to be used in create_baseline_object(). Sets quantile values larger than cut-off value to TRUE otherwise FALSE.

Examples

```
# Creates a simple function, where `data` is a `BaselineDataFrame`:
function(data) {
  ext <- data$ext == 0
  q <- get_quantiles(data, name)
  ifelse(ext, q > int_cutoff, q > ext_cutoff)
}
```

bin_var *Create binary covariate*

Description

Create an object of class SimVarBin to hold proportions of binary variables specified in a simulation study.

Usage

```
bin_var(
  prob_internal,
  prob_external,
  mu_internal_before_bin = 0,
  mu_external_before_bin = 0
)
```

Arguments

prob_internal	numeric. Proportion for the internal arms.
prob_external	numeric. Proportion for the external arm.
mu_internal_before_bin	numeric. Mean value of the covariate before binarization for the internal arms. The default is 0. See details for more information.
mu_external_before_bin	numeric. Mean value of the covariate before binarization for the external arm. The default is 0. See details for more information.

Details

This function contains information necessary to create binary covariates as part of a simulation study. The binary covariates are created by binarizing multivariate normal distributions to achieve the probabilities specified in prob_internal and prob_external. The user may choose to change the default mean value of each variable prior to binarization by specifying mu_internal_before_bin or mu_external_before_bin to ensure the correct scales are used in the covariance matrix, though the ultimate proportions will depend on prob_internal and prob_external. The default values for mu_internal_before_bin and mu_external_before_bin are 0, and it is not recommended to change these without good reason.

Value

Object of class SimVarBin.

See Also

Other simvar: cont_var()

Examples

```
cv1 <- bin_var(0.50, 0.80)
cv2 <- bin_var(.95, .92)
```

Borrowing-class Borrowing *Class*

Description

A class for defining borrowing details. Objects of class Borrowing should not be created directly but by the constructors borrowing_hierarchical_commensurate(), borrowing_none(), borrowing_full().

Slots

data_stan_code	string. Code to include in the Stan data program block.
method_name	string. The name of the method.
ext_flag_col	character. Name of the external flag column in the matrix.

See Also

Prior constructor functions: `borrowing_full()`, `borrowing_hierarchical_commensurate()`, `borrowing_none()`

Other borrowing classes: `BorrowingFull-class`, `BorrowingHierarchicalCommensurate-class`, `BorrowingNone-class`

BorrowingFull-class BorrowingFull *class*

Description

A class for defining details for "Full Borrowing" methods. Objects of class `BorrowingFull` should not be created directly but by the constructor `borrowing_full()`.

Slots

`data_stan_code` string. Code to include in the Stan data program block.

`method_name` string. The name of the method.

`ext_flag_col` character. Name of the external flag column in the matrix.

`name_tau` named vector for hierarchical commensurability parameter hyperprior.

See Also

Other borrowing classes: `Borrowing-class`, `BorrowingHierarchicalCommensurate-class`, `BorrowingNone-class`

BorrowingHierarchicalCommensurate-class BorrowingHierarchicalCommensurate *class*

Description

A class for defining details of dynamic borrowing using the hierarchical Bayesian model with a commensurability parameter. Objects of class `BorrowingHierarchicalCommensurate` should not be created directly but by the constructor `borrowing_hierarchical_commensurate()`.

Slots

`data_stan_code` string. Code to include in the Stan data program block.

`method_name` string. The name of the method.

`ext_flag_col` character. Name of the external flag column in the matrix.

`tau_prior` Prior. Prior for the commensurability parameter.

See Also

Other borrowing classes: `Borrowing-class`, `BorrowingFull-class`, `BorrowingNone-class`

BorrowingNone-class BorrowingNone *class*

Description

A class for defining details for "No borrowing" methods. Objects of class BorrowingNone should not be created directly but by the constructor borrowing_none().

Slots

data_stan_code string. Code to include in the Stan data program block.

method_name string. The name of the method.

ext_flag_col character. Name of the external flag column in the matrix.

See Also

Other borrowing classes: Borrowing-class, BorrowingFull-class, BorrowingHierarchicalCommensurate-class

borrowing_details *Legacy function for specifying borrowing details*

Description

Please use one of borrowing_hierarchical_commensurate(), borrowing_none(), or borrowing_full() instead.

Usage

```
borrowing_details(...)
```

Arguments

...	Deprecated arguments to borrowing_details.

Value

This function does not return a value. When called, it triggers an error message indicating that borrowing_details() is deprecated and that one of borrowing_hierarchical_commensurate(), borrowing_none(), or borrowing_full() should be used instead.

`borrowing_full` *Full borrowing*

Description

Full borrowing

Usage

```
borrowing_full(ext_flag_col)
```

Arguments

`ext_flag_col`	character. Name of the external flag column in the matrix.

Details

Method:

This method does not distinguish between internal and external arms, effectively pooling patients.

External Control:

The `ext_flag_col` argument refers to the column in the data matrix that contains the flag indicating a patient is from the external control cohort.

Value

Object of class `BorrowingFull`.

See Also

Other borrowing: `borrowing_none()`

Examples

```
fb <- borrowing_full("ext")
```

`borrowing_hierarchical_commensurate` *Hierarchical commensurate borrowing*

Description

Hierarchical commensurate borrowing

Usage

```
borrowing_hierarchical_commensurate(ext_flag_col, tau_prior)
```

Arguments

`ext_flag_col`	character. Name of the external flag column in the matrix.
`tau_prior`	Prior. Prior for the commensurability parameter.

Details

Method:

In Bayesian dynamic borrowing using the hierarchical commensurate prior approach, external control information is borrowed to the extent that the outcomes (i.e., log hazard rates or log odds) are similar between external and internal control populations. See Viele 2014 doi:10.1002/pst.1589 and Hobbs 2011 doi:10.1111/j.15410420.2011.01564.x for details.

External Control:

The `ext_flag_col` argument refers to the column in the data matrix that contains the flag indicating a patient is from the external control cohort.

Tau Prior:

The `tau_prior` argument specifies the hyperprior on the precision parameter commonly referred to as the commensurability parameter. See Viele 2014 doi:10.1002/pst.1589 for more details. This hyperprior determines (along with the comparability of the outcomes between internal and external controls) how much borrowing of the external control group will be performed. Example hyperpriors include largely uninformative inverse gamma distributions [e.g., `prior_gamma(alpha = .001, beta = .001)`] as well as more informative distributions [e.g., `prior_gamma(alpha = 1, beta = .001)`], though any distribution $x \in (0, \infty)$ can be used. Distributions with more density at higher values of x (i.e., higher precision) will lead to more borrowing.

Value

Object of class `BorrowingHierarchicalCommensurate`.

References

Viele, K., Berry, S., Neuenschwander, B., Amzal, B., Chen, F., Enas, N., Hobbs, B., Ibrahim, J.G., Kinnersley, N., Lindborg, S., Micallef, S., Roychoudhury, S. and Thompson, L. (2014), Use of historical control data for assessing treatment effects in clinical trials. **Pharmaceut. Statist., 13: 41–54**. doi:10.1002/pst.1589

Hobbes, B.P., Carlin, B.P., Mandrekar, S.J. and Sargent, D.J. (2011), Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. **Biometrics, 67: 1047–1056**. doi:10.1111/j.15410420.2011.01564.x

Examples

```
db <- borrowing_hierarchical_commensurate(
  ext_flag_col = "ext",
  tau_prior = prior_gamma(0.0001, 0.0001)
)
```

---

borrowing_none *No borrowing*

Description

No borrowing

Usage

```
borrowing_none(ext_flag_col)
```

Arguments

`ext_flag_col`	character. Name of the external flag column in the matrix.

Details

Method:

This method evaluates only the internal comparison, ignoring historical controls. Note that this method will filter the model matrix based on values in `ext_flag_col`.

External Control:

The `ext_flag_col` argument refers to the column in the data matrix that contains the flag indicating a patient is from the external control cohort.

Value

Object of class `BorrowingNone`.

See Also

Other borrowing: `borrowing_full()`

Examples

```
db <- borrowing_none(
  ext_flag_col = "ext"
)
```

---

c *Combine objects in* psborrow2

Description

Creates data.frame objects from various classes in psborrow2

Usage

```
## S4 method for signature 'SimDataList'
c(x, ...)
```

Arguments

x	object of type: SimDataList
...	additional objects to combine

Value

A combined object

list of SimDataList objects.

---

cauchy_prior *Legacy function for the cauchy prior*

Description

Please use prior_cauchy() instead.

Usage

```
cauchy_prior(...)
```

Arguments

...	Deprecated arguments to cauchy_prior().

Value

This function does not return a value. When called, it triggers an error message indicating that cauchy_prior() is deprecated and that prior_cauchy() should be used instead.

check_cmdstanr *Check Stan*

Description

Check whether cmdstanr is available and prints version and logistic example.

Usage

```
check_cmdstanr(check_sampling = FALSE)

check_cmdstan()
```

Arguments

check_sampling	Compile and sample from the "logistic" example model.

Value

check_cmdstanr() prints results from checks.

check_cmdstan() returns TRUE if CmdStan seems to be installed, otherwise FALSE

Functions

- check_cmdstan(): Check if the CmdStan command line tools are available.

Examples

```
check_cmdstanr()
```

check_data_matrix_has_columns *Check Data Matrix for Required Columns*

Description

Check that an Analysis object's data_matrix has all the required variables.

Usage

```
check_data_matrix_has_columns(object)
```

Arguments

object	Analysis. Object to check.

Value

`stop()` if some columns are missing.

Examples

```
anls <- create_analysis_obj(
  data_matrix = example_matrix,
  covariates = add_covariates(
    covariates = c("cov1", "cov2"),
    priors = prior_normal(0, 1000)
  ),
  outcome = outcome_surv_exponential(
    "time",
    "cnsr",
    baseline_prior = prior_normal(0, 1000)
  ),
  borrowing = borrowing_hierarchical_commensurate(
    "ext",
    prior_exponential(.001)
  ),
  treatment = treatment_details(
    "trt",
    prior_normal(0, 1000)
  )
)

check_data_matrix_has_columns(anls)
```

---

check_fixed_external_data

Create a Fixed External Data Object

Description

Create a Fixed External Data Object

Usage

```
check_fixed_external_data(data, req_cols)
```

Arguments

`data`	A `data.frame` containing external control data
`req_cols`	A `character` vector of required covariate columns

Value

A `DataSimObject` with updated `enrollment_internal` and `enrollment_external` slots.

ContinuousOutcome-class ContinuousOutcome *class*

Description

ContinuousOutcome class

Slots

function_stan_code character. Code to include in the Stan functions program block.

param_stan_code character. Code to include in the Stan parameters program block.

likelihood_stan_code character. Code defining the likelihood to include in the Stan model program block.

data_stan_code character. Code to include in the Stan data program block.

n_param integer. Number of ancillary parameters for the model to estimate.

param_priors list. Named list of prior distributions on the ancillary parameters in the model.

continuous_var character. Variable used for outcome in ContinuousOutcome objects.

baseline_prior Prior. Object of class Prior specifying prior distribution for the baseline outcome.

name_beta_trt. Named vector for beta_trt.

name_exp_trt. Named vector for exponentiated beta_trt

alpha_type. How to interpret alpha.

name_addnl_params. Named vector for additional parameters.

See Also

Other outcome: BinaryOutcome-class, Outcome-class, OutcomeBinaryLogistic-class, OutcomeContinuousNormal-c
OutcomeSurvExponential-class, OutcomeSurvWeibullPH-class, TimeToEvent-class

cont_var *Create continuous covariate*

Description

Create an object of class SimVarCont to hold mean values of of continuous variables specified in a simulation study.

Usage

```
cont_var(mu_internal, mu_external)
```

Arguments

mu_internal	numeric. Mean covariate value for the internal arms.
mu_external	numeric. Mean covariate value for the external arm.

Value

Object of class SimVarCont.

See Also

Other simvar: bin_var()

Examples

```
cv1 <- cont_var(0.5, 1)
cv2 <- cont_var(10, 10)
```

---

covariance_matrix *Create Covariance Matrix*

Description

Create Covariance Matrix

Usage

```
covariance_matrix(diag, upper_tri)
```

Arguments

diag	Diagonal entries of the covariance matrix
upper_tri	Upper triangle entries of the matrix, specified column wise.

Value

A symmetric matrix with diag values on the main diagonal and upper_tri values in the lower and upper triangles.

Examples

```
m1 <- covariance_matrix(c(1, 1, 1, 1), c(.8, .3, .8, 0, 0, 0))
m1
mvtnorm::rmvnorm(5, mean = c(0, 0, 0, 0), sigma = m1)

# No correlation
covariance_matrix(c(1, 2, 3))
```

Covariates-class Covariate *Class*

Description

A class for defining covariate details. Objects of class `Covariate` should not be created directly but by the constructor `add_covariates()`.

Slots

`covariates` character. Names of columns in the data matrix containing covariates to be adjusted for in the outcome model. Note: the external and treatment flags should not go here.

`priors.` Either a single object of class `Prior` specifying the prior distribution to apply to all covariates or a named list of distributions of class `Prior`, one for each covariate

`name_betas.` Names for the beta parameters in the STAN model.

create_alpha_string *Create alpha string*

Description

Create alpha string

Usage

```
create_alpha_string(borrowing_object, outcome_object)

## S4 method for signature 'Borrowing'
create_alpha_string(borrowing_object, outcome_object)

## S4 method for signature 'BorrowingHierarchicalCommensurate'
create_alpha_string(borrowing_object, outcome_object)
```

Arguments

`borrowing_object` borrowing object

`outcome_object` outcome object

`create_analysis_obj` *Compile MCMC sampler using STAN and create analysis object*

Description

Compile MCMC sampler using STAN and create analysis object

Usage

```
create_analysis_obj(
  data_matrix,
  outcome,
  borrowing,
  treatment,
  covariates = NULL,
  quiet = FALSE
)
```

Arguments

`data_matrix`	matrix. The data matrix, including all covariates to be adjusted for, all relevant outcome variables, and treatment arm and external control arm flags.
`outcome`	`Outcome`. Object of class `Outcome` as output by `outcome_surv_exponential()`, `outcome_surv_weibull_ph()`, or `outcome_bin_logistic()`.
`borrowing`	`Borrowing`. Object of class `Borrowing` as output by `borrowing_full()`, `borrowing_none()`, and `borrowing_hierarchical_commensurate()`.
`treatment`	`Treatment`. Object of class `Treatment` as output by `treatment_details()`.
`covariates`	`Covariates`. Object of class `Covariates` as output by the function `add_covariates()`.
`quiet`	logical. Whether to suppress messages (`TRUE`) or not (`FALSE`, the default)

Value

Object of class `Analysis`.

Examples

```
if (check_cmdstan()) {
  anls <- create_analysis_obj(
    data_matrix = example_matrix,
    outcome = outcome_surv_exponential(
      "time",
      "cnsr",
      baseline_prior = prior_normal(0, 1000)
    ),
    borrowing = borrowing_hierarchical_commensurate(
      "ext",
      prior_exponential(.001)
```

```
    ),
    treatment = treatment_details(
      "trt",
      prior_normal(0, 1000)
    ),
    covariates = add_covariates(
      covariates = c("cov1", "cov2"),
      priors = prior_normal(0, 1000)
    )
  )
}
```

`create_baseline_object` *Create Baseline Data Simulation Object*

Description

Create Baseline Data Simulation Object

Usage

```
create_baseline_object(
  n_trt_int,
  n_ctrl_int,
  n_ctrl_ext,
  covariates,
  transformations
)
```

Arguments

`n_trt_int`	Number of internal treated patients
`n_ctrl_int`	Number of internal control patients
`n_ctrl_ext`	Number of external control patients
`covariates`	List of correlated covariates objects, see `baseline_covariates()`
`transformations`	List of named transformation functions.

Details

Transformation functions are evaluated in order and create or overwrite a column in the data.frame with that name. The function should take a `data.frame` (specifically a `BaselineDataFrame` object from `generate(BaselineObject)`) and return a vector with length identical to the total number of patients. The `@BaselineObject` slot may be accessed directly or with `get_quantiles()` to create transformations. See `binary_cutoff()`

Value

A BaselineObject

Examples

```
bl_no_covs <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 100
)


bl_biomarkers <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 100,
  covariates = baseline_covariates(
    c("b1", "b2", "b3"),
    means_int = c(0, 0, 0),
    covariance_int = covariance_matrix(c(1, 1, 1), c(.8, .3, .8))
  ),
  transformations = list(
    exp_b1 = function(data) exp(data$b1),
    b2 = binary_cutoff("b2", int_cutoff = 0.7, ext_cutoff = 0.5)
  )
)
```

---

create_data_matrix *Create Data Matrix*

Description

Creates a matrix suitable for create_analysis_obj(). Creates dummy variables for factors and allows transformations of covariates specified with a formula.

Usage

```
create_data_matrix(
  data,
  outcome,
  trt_flag_col,
  ext_flag_col,
  covariates = NULL,
  weight_var = NULL
)
```

Arguments

data	data.frame. Data containing all variables
outcome	character. The outcome variable for binary outcomes or the time and censoring variables.
trt_flag_col	character. The treatment indicator variable.
ext_flag_col	character. The external cohort indicator.
covariates	character or formula. The covariates for model adjustment.
weight_var	character. An optional weight variable.

Value

Invisibly returns a matrix containing all variables to pass to create_analysis_obj(). Prints names of covariates columns to use with add_covariates().

Examples

```
dat <- survival::diabetic
dat$ext <- dat$trt == 0 & dat$id > 1000
data_mat <- create_data_matrix(
  dat,
  outcome = c("time", "status"),
  trt_flag_col = "trt",
  ext_flag_col = "ext",
  covariates = ~ age + laser + log(risk)
)
data_mat
```

---

create_data_simulation

Data Simulation

Description

Data Simulation

Usage

```
create_data_simulation(
  baseline,
  coefficients = numeric(),
  treatment_hr = 1,
  drift_hr = 1,
  event_dist,
  fixed_external_data
)
```

Arguments

baseline	BaselineObject from create_baseline_object()
coefficients	Named vector of coefficients for linear predictor. Must correspond to variables in baseline object
treatment_hr	Default treatment hazard ratio for simulations. Alternative simulation settings can be specified in generate.
drift_hr	Default drift hazard ratio between internal and external arms. Alternative simulation settings can be specified in generate.
event_dist	Specify time to event distribution with SimDataEvent object from create_event_dist()
fixed_external_data	A data.frame containing external control data. It must contain columns eventtime, status and all of the variables named in coefficients. If present, trt must be 0 and ext must be 1 for all rows.

Value

DataSimObject

Examples

```
baseline_obj <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 10,
  covariates = baseline_covariates(
    names = c("age", "score"),
    means_int = c(55, 5),
    means_ext = c(60, 5),
    covariance_int = covariance_matrix(c(5, 1))
  )
)
sim_obj <- create_data_simulation(
  baseline_obj,
  coefficients = c(age = 0.001, score = 1.5),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
data_sim_list <- generate(sim_obj, treatment_hr = c(0.5, 1), drift_hr = 0.5)
```

---

create_event_dist *Specify a Time to Event Distribution*

Description

Uses simsurv::simsurv to generate time to event data. See simsurv help for more details.

Usage

```
create_event_dist(
  dist = NULL,
  lambdas = NULL,
  gammas = NULL,
  mixture = FALSE,
  pmix = 0.5,
  hazard = NULL,
  loghazard = NULL,
  cumhazard = NULL,
  logcumhazard = NULL,
  ...
)

null_event_dist()
```

Arguments

dist	Specify the distribution "exponential"
lambdas	Scale parameter
gammas	Second parameter needed for Weibull or Gompertz distributions
mixture	Use mixture model?
pmix	Proportion of mixtures
hazard	A user defined hazard function
loghazard	Alternatively, a user defined log hazard function
cumhazard	Alternatively, a user defined cumulative hazard function
logcumhazard	Alternatively, a user defined log cumulative hazard function
...	Other simsurv parameters

Value

A SimDataEvent object

null_event_dist returns an object with no parameters specified that does not simulate event times.

Examples

```
weibull_surv <- create_event_dist(dist = "weibull", lambdas = 1 / 200, gammas = 1)
exp_event_dist <- create_event_dist(dist = "exponential", lambdas = 1 / 36)
null_event_dist()
```

`create_simulation_obj` *Compile MCMC sampler using STAN and create simulation object*

Description

Compile MCMC sampler using STAN and create simulation object

Usage

```
create_simulation_obj(
  data_matrix_list,
  covariate = NULL,
  outcome,
  borrowing,
  treatment,
  quiet = TRUE
)
```

Arguments

`data_matrix_list`	`SimDataList`. The list of lists of data matrices created with `sim_data_list()`.
`covariate`	`SimCovariateList` or `Covariate` or `NULL`. List of `Covariate` objects created with `sim_covariate()`, a single `Covariate` object created by `add_covariates()`, or `NULL` (no covariate adjustment).
`outcome`	`SimOutcomeList` or `Outcome`. List of `Outcome` objects created with `sim_outcome()`, or single `Outcome` object (e.g., created by `outcome_surv_exponential()`).
`borrowing`	`SimBorrowingList` or `Borrowing`. List of `Borrowing` objects created with `sim_borrowing()`, or a single `Borrowing` object created by `borrowing_full()`, `borrowing_none()`, or `borrowing_hierarchical_commensurate()`.
`treatment`	`SimTreatmentList` or `Treatment`. List of `Treatment` objects created with `sim_treatment()` or a single `Treatment` object created by `treatment_details()`.
`quiet`	logical. Whether to print messages (`quiet = FALSE`) or not (`quiet = TRUE`, the default)

Value

Object of class `Simulation`.

Examples

```
base_mat <- matrix(
  c(
    rep(0, 200), rep(0, 200), rep(1, 200),
    rep(1, 200), rep(0, 200), rep(0, 200),
    rep(0, 600)
```

```
  ),
  ncol = 3,
  dimnames = list(NULL, c("ext", "trt", "driftOR"))
)

add_binary_endpoint <- function(odds_ratio,
                                base_matrix = base_mat) {
  linear_predictor <- base_matrix[, "trt"] * log(odds_ratio)
  prob <- 1 / (1 + exp(-linear_predictor))

  bin_endpoint <- rbinom(
    NROW(base_matrix),
    1,
    prob
  )

  cbind(base_matrix, matrix(bin_endpoint, ncol = 1, dimnames = list(NULL, "ep")))
}

data_list <- list(
  list(add_binary_endpoint(1.5), add_binary_endpoint(1.5)),
  list(add_binary_endpoint(2.5), add_binary_endpoint(2.5))
)

guide <- data.frame(
  trueOR = c(1.5, 2.5),
  driftOR = c(1.0, 1.0),
  index = 1:2
)

sdl <- sim_data_list(
  data_list = data_list,
  guide = guide,
  effect = "trueOR",
  drift = "driftOR",
  index = "index"
)

if (check_cmdstan()) {
  sim_object <- create_simulation_obj(
    data_matrix_list = sdl,
    outcome = outcome_bin_logistic("ep", prior_normal(0, 1000)),
    borrowing = sim_borrowing_list(list(
      full_borrowing = borrowing_full("ext"),
      bdb = borrowing_hierarchical_commensurate("ext", prior_exponential(0.0001))
    )),
    treatment = treatment_details("trt", prior_normal(0, 1000))
  )
}
```

`create_tau_string` *Create tau string*

Description

Create tau string

Usage

```
create_tau_string(borrowing_object)

## S4 method for signature 'Borrowing'
create_tau_string(borrowing_object)

## S4 method for signature 'BorrowingHierarchicalCommensurate'
create_tau_string(borrowing_object)
```

Arguments

borrowing_object	borrowing object

---

custom_enrollment *Create a* DataSimEnrollment *Object*

Description

Create a DataSimEnrollment Object

Usage

```
custom_enrollment(fun, label)
```

Arguments

fun	A function that takes one argument n the number of enrollment times to observe and returns a vector of times.
label	A user-friendly label

Value

A DataSimEnrollment object

Examples

```
custom_enrollment(
  fun = function(n) rpois(n, lambda = 5),
  label = "Poisson enrollment distribution"
)
```

cut_off_funs *Cut Off Functions*

Description

Cut Off Functions

Usage

```
cut_off_none()

cut_off_after_first(time)

cut_off_after_last(time)

cut_off_after_events(n)
```

Arguments

time	Time to cut off
n	Number of events

Value

A DataSimCutOff object containing a cut-off function

Functions

- cut_off_none(): No cut off is specified
- cut_off_after_first(): Cut off at time after first enrolled patient
- cut_off_after_last(): Cut off at time after last enrolled patient
- cut_off_after_events(): Cut off after the time of the n-th event

Examples

```
cut_off_none()
cut_off_after_first(time = 36)
cut_off_after_last(time = 36)
cut_off_after_events(n = 20)
```

DataSimCutOff-class *Cut Off Object*

Description

Cut Off Object

Slots

`cut_off_fun` A function that takes a `data.frame` with columns of enrollment time, survival time and outcome. The function returns a modified `data.frame` after applied the cut-off rule.

DataSimEnrollment-class *Enrollment Object*

Description

Enrollment Object

Slots

`fun` A function that takes one argument `n` the number of enrollment times to observe and returns a vector of times.

`label` A user-friendly label

DataSimEvent-class *Event Time Distribution Object*

Description

Event Time Distribution Object

Slots

`params` Parameters used for simulating event times with `simsurv::simsurv()`.

`label` Description of the distribution.

DataSimFixedExternalData-class *Fixed External Control Data Object*

Description

Fixed External Control Data Object

Value

A `FixedExternalData`

Slots

`data` `data.frame` containing external control data

`n` Number of observations

DataSimObject-class *Data Simulation Object Class*

Description

Data Simulation Object Class

Value

A `DataSimObject`

Slots

`baseline` `BaselineObject` from create_baseline_object

`coefficients` Named `numeric` vector of `beta` coefficients for survival model. See `beta` at `?simsurv::simsurv`

`treatment_hr` `numeric` treatment effect as a hazard ration. `log(treatment_hr)` is included in `beta` with `coefficients` and `log(drift_hr)`. This default is overridden by generate arguments

`drift_hr` `numeric` hazard ratio between internal and external arms. Included as `log(drift_hr)`.

`fixed_external_data` `data.frame` for external data. Currently unused.

`event_dist` `DataSimEvent` parameters for outcome distribution from `create_event_dist()`

`enrollment` `DataSimEnrollment` object.

`cut_off` `DataSimCutOff`

enrollment_constant *Constant Enrollment Rates*

Description

Constant Enrollment Rates

Usage

```
enrollment_constant(rate, for_time = rep(1, length(rate)))
```

Arguments

rate	Number of patients to enroll per unit time
for_time	Number of time periods for each rate. Must be equal length to rate

Value

An object of class DataSimEnrollment to be passed to create_data_simulation()

Examples

```
# 10 patients/month for 6 months, then 5/month for 6 months
enroll_obj <- enrollment_constant(rate = c(10, 5), for_time = c(6, 6))
enroll_obj@fun(n = 80)
```

eval_constraints *Evaluate constraints*

Description

Evaluate constraints when these are called

Usage

```
eval_constraints(object)

## S4 method for signature 'Prior'
eval_constraints(object)
```

Arguments

object	Prior object

example_matrix *Example data matrix*

Description

A matrix containing data from a clinical trial with a treatment arm and a control arm, as well as data from an external control. In this simulated dataset, the true hazard ratio (HR) for the time-to-event endpoint comparing the experimental treatment to the control treatment is 0.70. The true odds ratio (OR) for the binary response endpoint comparing the experimental treatment to the control treatment is 1.20.

Usage

```
example_matrix
```

Format

A data frame with 500 rows and 11 columns. The distributions of patients is: 50 internal control patients, 100 internal experimental patients, 350 external control patients.

id patient identifier

ext 0/1, flag for external controls

trt 0/1, flag for treatment arm

cov1 0/1, baseline covariate

cov2 0/1, baseline covariate

cov3 0/1, baseline covariate

cov4 0/1, baseline covariate

time numeric >0, survival time

status 0/1, indicator for event status (1 = had event, 0 = did not have event)

cnsr 0/1, censoring indicator (1 = was censored, 0 = was not censored). This value is 1 - status.

resp 0/1, indicator for response outcome (1 = had a response, 0 = did not have a response)

example_surv *Simulated Survival Data*

Description

A data frame containing simulated data from a clinical trial with a treatment arm (n=200) and a control arm (n=158), as well as data from an external control (n=242).

Usage

```
example_surv
```

Format

A data frame with 600 rows and 6 variables:

trt 0/1, flag for treatment arm

ext 0/1, flag for external controls

eventtime numeric >0, survival time

status 0/1, event indicator

censor 0/1, censoring indicator

cov1 0/1, binary baseline covariate 1

cov2 integer in [0, 15], baseline covariate 2

cov3 continuous numeric, baseline covariate 3

---

exponential_prior *Legacy function for the exponential prior*

Description

Please use `prior_exponential()` instead.

Usage

```
exponential_prior(...)
```

Arguments

`...`	Deprecated arguments to `exponential_prior()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `exponential_prior()` is deprecated and that `prior_exponential()` should be used instead.

exp_surv_dist *Legacy function for the exponential survival distribution*

Description

Please use `outcome_surv_exponential()` instead.

Usage

```
exp_surv_dist(...)
```

Arguments

`...`	Deprecated arguments to `exp_surv_dist()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `exp_surv_dist()` is deprecated and that `outcome_surv_exponential()` should be used instead.

gamma_prior *Legacy function for the gamma prior*

Description

Please use `prior_gamma()` instead.

Usage

```
gamma_prior(...)
```

Arguments

`...`	Deprecated arguments to `gamma_prior()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `gamma_prior()` is deprecated and that `prior_gamma()` should be used instead.

generate *Generate Data from Object*

Description

Generate Data from Object

Usage

```
generate(x, ...)
```

Arguments

x	object
...	Other arguments passed to methods

Value

Object of class SimDataList.

generate,BaselineObject-method *Generate Data for a* BaselineObject

Description

Generate Data for a BaselineObject

Usage

```
## S4 method for signature 'BaselineObject'
generate(x, ...)
```

Arguments

x	a BaselineObject object created by create_baseline_object
...	additional parameters are ignored

Value

A BaselineDataFrame object

Examples

```
bl_biomarkers <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 100,
  covariates = baseline_covariates(
    c("b1", "b2", "b3"),
    means_int = c(0, 0, 0),
    covariance_int = covariance_matrix(c(1, 1, 1), c(.8, .3, .8))
  ),
  transformations = list(
    exp_b1 = function(data) exp(data$b1),
    b2 = binary_cutoff("b2", int_cutoff = 0.7, ext_cutoff = 0.5)
  )
)
generate(bl_biomarkers)
```

---

generate,DataSimObject-method

Generate Data for a DataSimObject

Description

Generate Data for a DataSimObject

Usage

```
## S4 method for signature 'DataSimObject'
generate(x, n = 1, treatment_hr = NULL, drift_hr = NULL)
```

Arguments

x	a DataSimObject object created by create_data_simulation
n	number of data sets to simulate
treatment_hr	vector of numeric treatment effects
drift_hr	vector of numeric drift effects

Value

A SimDataList object for use with create_simulation_obj().

Examples

```
baseline_obj <- create_baseline_object(
  n_trt_int = 100,
  n_ctrl_int = 50,
  n_ctrl_ext = 10,
  covariates = baseline_covariates(
```

```
    names = c("age", "score"),
    means_int = c(55, 5),
    means_ext = c(60, 5),
    covariance_int = covariance_matrix(c(5, 1))
  )
)
sim_obj <- create_data_simulation(
  baseline_obj,
  coefficients = c(age = 0.001, score = 1.5),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
data_sim_list <- generate(sim_obj, treatment_hr = c(0, 1), drift_hr = 0.5)
```

---

get_cmd_stan_models *Get* CmdStanModel *objects for* MCMCSimulationResults

---

Description

Show the CmdStanModel objects from MCMCSimulationResults objects.

Usage

```
get_cmd_stan_models(object)

## S4 method for signature 'MCMCSimulationResult'
get_cmd_stan_models(object)
```

Arguments

object	MCMCSimulationResults object

Value

List of lists of CmdStanModel objects for each model.

---

get_data *Get Simulated Data from* SimDataList *object*

Description

Retrieves the simulated data from a SimDataList object by index.

Usage

```
get_data(object, index = 1, dataset = 1)

## S4 method for signature 'SimDataList'
get_data(object, index = NULL, dataset = NULL)
```

Arguments

object	SimDataList object
index	the index of the scenario (see guide with print(SimDataList))
dataset	the dataset out of n_datasets_per_param

Value

Simulated data as a data frame if the index is specified, else as a list

---

get_quantiles *Get Quantiles of Random Data*

Description

Helper for use within transformation functions for create_baseline_object().

Usage

```
get_quantiles(object, var)
```

Arguments

object	a BaselineDataFrame
var	character string name of the variable

Value

A numeric vector containing quantiles based on the data generating distribution.

---

get_results *Get results for* MCMCSimulationResults *objects*

Description

Get the results data.frame from MCMCSimulationResults objects.

Usage

```
get_results(object)

## S4 method for signature 'MCMCSimulationResult'
get_results(object)
```

Arguments

object	MCMCSimulationResults object

Value

data.frame with simulation results.

---

get_stan_code *Get method for Stan model*

Description

Get method for Stan model

Usage

```
get_stan_code(object)

## S4 method for signature 'Analysis'
get_stan_code(object)
```

Arguments

object	Analysis object

Value

String containing the Stan model

---

get_vars *Get Variables*

Description

Gets the data variable names from an object.

Usage

```
get_vars(object)

## S4 method for signature 'Covariates'
get_vars(object)

## S4 method for signature 'Treatment'
get_vars(object)

## S4 method for signature 'Borrowing'
get_vars(object)

## S4 method for signature 'TimeToEvent'
get_vars(object)

## S4 method for signature 'BinaryOutcome'
get_vars(object)

## S4 method for signature 'ContinuousOutcome'
get_vars(object)

## S4 method for signature 'Analysis'
get_vars(object)

## S4 method for signature 'NULL'
get_vars(object)

## S4 method for signature 'SimTreatmentList'
get_vars(object)

## S4 method for signature 'SimOutcomeList'
get_vars(object)

## S4 method for signature 'SimBorrowingList'
get_vars(object)

## S4 method for signature 'SimCovariateList'
get_vars(object)

## S4 method for signature 'Simulation'
get_vars(object)

## S4 method for signature 'BaselineObject'
get_vars(object)
```

Arguments

object	Object

Value

A character vector containing variable names

Examples

```
get_vars(treatment_details(
  trt_flag_col = "treat_fl",
  trt_prior = prior_normal(0, 1000)
))
```

---

half_cauchy_prior *Legacy function for the half-cauchy prior*

Description

Please use prior_half_cauchy() instead.

Usage

```
half_cauchy_prior(...)
```

Arguments

...	Deprecated arguments to half_cauchy_prior().

Value

This function does not return a value. When called, it triggers an error message indicating that half_cauchy_prior() is deprecated and that prior_half_cauchy() should be used instead.

---

half_normal_prior *Legacy function for the normal half prior*

Description

Please use prior_half_normal() instead.

Usage

```
half_normal_prior(...)
```

Arguments

...	Deprecated arguments to half_normal_prior().

Value

This function does not return a value. When called, it triggers an error message indicating that half_normal_prior() is deprecated and that prior_half_normal() should be used instead.

logistic_bin_outcome *Legacy function for binary logistic regression*

Description

Please use `outcome_bin_logistic()` instead.

Usage

```
logistic_bin_outcome(...)
```

Arguments

`...`	Deprecated arguments to `logistic_bin_outcome`.

Value

This function does not return a value. When called, it triggers an error message indicating that `logistic_bin_outcome()` is deprecated and that `outcome_bin_logistic()` should be used instead.

make_model_string_model *Create Stan Code for Model*

Description

Create Stan Code for Model

Usage

```
make_model_string_model(borrowing, outcome, analysis_obj)

## S4 method for signature 'ANY,ANY,Analysis'
make_model_string_model(borrowing, outcome, analysis_obj)

## S4 method for signature 'BorrowingFull,ANY,Analysis'
make_model_string_model(borrowing, outcome, analysis_obj)

## S4 method for signature 'BorrowingNone,ANY,Analysis'
make_model_string_model(borrowing, outcome, analysis_obj)

## S4 method for signature 'BorrowingHierarchicalCommensurate,ANY,Analysis'
make_model_string_model(borrowing, outcome, analysis_obj)
```

Arguments

borrowing	borrowing object
outcome	outcome object
analysis_obj	analysis object

Value

glue character containing the Stan code for the data block.

Examples

```
anls_obj <- create_analysis_obj(
    data_matrix = example_matrix,
    outcome = outcome_surv_exponential(
      "time",
      "cnsr",
      baseline_prior = prior_normal(0, 1000)
    ),
    borrowing = borrowing_hierarchical_commensurate(
      "ext",
      prior_exponential(.001)
    ),
    treatment = treatment_details(
      "trt",
      prior_normal(0, 1000)
    ),
    covariates = add_covariates(
      covariates = c("cov1", "cov2"),
      priors = prior_normal(0, 1000)
    )
  )
make_model_string_model(anls_obj@borrowing, anls_obj@outcome, anls_obj)
```

MCMCSimulationResult-class
MCMCSimulationResult *Class*

Description

A class for defining Simulation study results. Objects of class MCMCSimulationResult should not be created directly but by mcmc_sample().

Slots

results data.frame. The results of the simulation study summarized in a data.frame

cmd_stan_models list. List of lists of CmdStanmodels corresponding to the different parameters in Simulation@guide and different datasets in Simulation@data_matrix_list.

`mcmc_sample` *Sample from Stan model*

Description

Method to sample from compiled Stan model and return a `CmdStanMCMC` object with draws.

Usage

```
mcmc_sample(x, ...)

## S4 method for signature 'ANY'
mcmc_sample(x, ...)

## S4 method for signature 'Analysis'
mcmc_sample(
  x,
  iter_warmup = 1000L,
  iter_sampling = 10000L,
  chains = 4L,
  verbose = FALSE,
  ...
)

## S4 method for signature 'Simulation'
mcmc_sample(
  x,
  posterior_quantiles = c(0.025, 0.975),
  iter_warmup = 1000L,
  iter_sampling = 10000L,
  chains = 4L,
  verbose = FALSE,
  keep_cmd_stan_models = FALSE,
  ...
)
```

Arguments

`x`	object to sample, such as `Analysis` (created with `create_analysis_obj()`) or `Simulation`.
`...`	additional arguments passed to the $sample() method of a `cmdstanr` Stan model. See https://mc-stan.org/cmdstanr/reference/model-method-sample.html
`iter_warmup`	integer. The number of warm up iterations to run per chain. The default is 1000.
`iter_sampling`	integer. The number of post-warm up iterations to run per chain. The default is 10000.
`chains`	integer. The number of Markov chains to run. The default is 4.

verbose logical. Whether to print sampler updates (TRUE) or not (FALSE)

posterior_quantiles
: numeric vector of length two. The posterior quantiles used for summarizing simulation results. The default is c(0.025, 0.975) See details.

keep_cmd_stan_models
: logical. Whether to keep the CmdStanModel objects from the mcmc_sampler (TRUE, discouraged in most scenarios) or not (FALSE). The default is FALSE.

Details

Simulation objects:

This function takes draws from an MCMC sampler and summarizes results.

Value

An object of class CmdStanMCMC

An object of class MCMCSimulationResult

Examples

```
## Analysis objects
if (check_cmdstan()) {
  anls <- create_analysis_obj(
    data_matrix = example_matrix,
    covariates = add_covariates(
      covariates = c("cov1", "cov2"),
      priors = prior_normal(0, 1000)
    ),
    outcome = outcome_surv_weibull_ph(
      "time",
      "cnsr",
      shape_prior = prior_normal(0, 1000),
      baseline_prior = prior_normal(0, 1000)
    ),
    borrowing = borrowing_hierarchical_commensurate(
      "ext",
      prior_exponential(.001)
    ),
    treatment = treatment_details("trt", prior_normal(0, 1000))
  )

  mcmc_results <- mcmc_sample(anls, chains = 1, iter_warmup = 500L, iter_sampling = 1000L)
}

## Simulation objects
base_mat <- matrix(
  c(
    rep(0, 200), rep(0, 200), rep(1, 200),
    rep(1, 200), rep(0, 200), rep(0, 200),
    rep(0, 600)
  ),
```

```
  ncol = 3,
  dimnames = list(NULL, c("ext", "trt", "driftOR"))
)

add_binary_endpoint <- function(odds_ratio,
                                base_matrix = base_mat) {
  linear_predictor <- base_matrix[, "trt"] * log(odds_ratio)
  prob <- 1 / (1 + exp(-linear_predictor))

  bin_endpoint <- rbinom(
    NROW(base_matrix),
    1,
    prob
  )

  cbind(base_matrix, matrix(bin_endpoint, ncol = 1, dimnames = list(NULL, "ep")))
}

data_list <- list(
  list(add_binary_endpoint(1.5), add_binary_endpoint(1.5)),
  list(add_binary_endpoint(2.5), add_binary_endpoint(2.5))
)

guide <- data.frame(
  trueOR = c(1.5, 2.5),
  driftOR = c(1.0, 1.0),
  index = 1:2
)

sdl <- sim_data_list(
  data_list = data_list,
  guide = guide,
  effect = "trueOR",
  drift = "driftOR",
  index = "index"
)

if (check_cmdstan()) {
  sim_object <- create_simulation_obj(
    data_matrix_list = sdl,
    outcome = outcome_bin_logistic("ep", prior_normal(0, 1000)),
    borrowing = sim_borrowing_list(list(
      full_borrowing = borrowing_full("ext"),
      bdb = borrowing_hierarchical_commensurate("ext", prior_exponential(0.0001))
    )),
    treatment = treatment_details("trt", prior_normal(0, 1000))
  )

  mcmc_sample(sim_object, chains = 1, iter_warmup = 500L, iter_sampling = 1000L)
}
## Not run:
library(future)
# Use two separate R processes
```

```
plan("multisession", workers = 2)

# and two parallel threads in each.
mcmc_sample(sim_object, chains = 1, iter_warmup = 500L, iter_sampling = 1000L, parallel_chains = 2)

# Tidy up processes when finished
plan("sequential")

## End(Not run)
```

`normal_prior` *Legacy function for the normal prior*

Description

Please use `prior_normal()` instead.

Usage

```
normal_prior(...)
```

Arguments

`...`	Deprecated arguments to `normal_prior()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `normal_prior()` is deprecated and that `prior_normal()` should be used instead.

`Outcome-class` `Outcome` *class*

Description

`Outcome` class

See Also

Other outcome: `BinaryOutcome-class`, `ContinuousOutcome-class`, `OutcomeBinaryLogistic-class`, `OutcomeContinuousNormal-class`, `OutcomeSurvExponential-class`, `OutcomeSurvWeibullPH-class`, `TimeToEvent-class`

OutcomeBinaryLogistic-class
OutcomeBinaryLogistic *class*

Description

A class for defining a logistic regression with a binary outcome to be translated to Stan code. Objects of class OutcomeBinaryLogistic should not be created directly but by the constructor outcome_bin_logistic().

Slots

function_stan_code character. stan function code block containing text to interpolate into stan model. Empty string for OutcomeBinaryLogistic.

param_stan_code character. stan parameter code block containing text to interpolate into stan model. Empty string for OutcomeBinaryLogistic.

likelihood_stan_code character. stan model likelihood code block containing text to interpolate into stan model.

n_param integer. Number of ancillary parameters for the model to estimate (0).

param_priors list. Named list of prior distributions on the ancillary parameters in the model. Empty for OutcomeBinaryLogistic.

binary_var character. Variable used for outcome in OutcomeBinaryLogistic objects.

baseline_prior Prior. Object of class Prior specifying prior distribution for the baseline outcome.

name_beta_trt. Named vector for beta_trt.

name_exp_trt. Named vector for exponentiated beta_trt

alpha_type. How to interpret alpha.

name_addnl_params. Named vector for additional parameters.

See Also

Other outcome: BinaryOutcome-class, ContinuousOutcome-class, Outcome-class, OutcomeContinuousNormal-class
OutcomeSurvExponential-class, OutcomeSurvWeibullPH-class, TimeToEvent-class

OutcomeContinuousNormal-class OutcomeContinuousNormal *class*

Description

A class for defining a regression with a normal outcome to be translated to Stan code. Objects of class `OutcomeContinuousNormal` should not be created directly but by the constructor `outcome_cont_normal()`.

Slots

`function_stan_code` character. stan function code block containing text to interpolate into stan model. Empty string for `OutcomeContinuousNormal`.

`param_stan_code` character. stan parameter code block containing text to interpolate into stan model. Empty string for `OutcomeContinuousNormal`.

`likelihood_stan_code` character. stan model likelihood code block containing text to interpolate into stan model.

`n_param` integer. Number of ancillary parameters for the model to estimate (0).

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model. Empty for `OutcomeContinuousNormal`.

`continuous_var` character. Variable used for outcome in `OutcomeContinuousNormal` objects.

`baseline_prior` `Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome.

`name_beta_trt.` Named vector for beta_trt.

`name_exp_trt.` Named vector for exponentiated beta_trt

`alpha_type.` How to interpret alpha.

`name_addnl_params.` Named vector for additional parameters.

See Also

Other outcome: `BinaryOutcome-class`, `ContinuousOutcome-class`, `Outcome-class`, `OutcomeBinaryLogistic-class`, `OutcomeSurvExponential-class`, `OutcomeSurvWeibullPH-class`, `TimeToEvent-class`

OutcomeSurvExponential-class OutcomeSurvExponential *Class*

Description

A class for defining a time-to-event survival analysis with an exponential survival distribution. Objects of class `OutcomeSurvExponential` should not be created directly but by the constructor `outcome_surv_exponential()`.

Slots

`function_stan_code` character. stan function code block containing text to interpolate into stan model. Empty string for `OutcomeSurvExponential`.

`param_stan_code` character. stan parameter code block containing text to interpolate into stan model. Empty string for `OutcomeSurvExponential`.

`likelihood_stan_code` character. stan model likelihood code block containing text to interpolate into stan model.

`n_param` integer. Number of ancillary parameters for the model to estimate (0).

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model. Empty for `OutcomeSurvExponential`.

`time_var` character. Variable used for time in `TimeToEvent` objects.

`cens_var` character. Variable used for censoring in `TimeToEvent` objects.

`baseline_prior` `Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome.

`name_beta_trt`. Named vector for beta_trt.

`name_exp_trt`. Named vector for exponentiated beta_trt

`alpha_type`. How to interpret alpha.

`name_addnl_params`. Named vector for additional parameters.

See Also

Other outcome: `BinaryOutcome-class`, `ContinuousOutcome-class`, `Outcome-class`, `OutcomeBinaryLogistic-class`, `OutcomeContinuousNormal-class`, `OutcomeSurvWeibullPH-class`, `TimeToEvent-class`

OutcomeSurvWeibullPH-class

`OutcomeSurvWeibullPH` *Class*

Description

A class for defining a time-to-event survival analysis with a Weibull proportional hazards survival distribution. Objects of class `OutcomeSurvWeibullPH` should not be created directly but by the constructor `outcome_surv_weibull_ph()`.

Slots

`function_stan_code` character. Stan function code block containing text to interpolate into Stan model.

`param_stan_code` character. Stan parameter code block containing text to interpolate into Stan model.

`likelihood_stan_code` character. Stan model likelihood code block containing text to interpolate into Stan model.

`n_param` integer. Number of ancillary parameters for the model to estimate (1).

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.

`time_var` character. Variable used for time in `TimeToEvent` objects.

`cens_var` character. Variable used for censoring in `TimeToEvent` objects.

`baseline_prior` `Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome.

`name_beta_trt.` Named vector for beta_trt.

`name_exp_trt.` Named vector for exponentiated beta_trt

`alpha_type.` How to interpret alpha.

`name_addnl_params.` Named vector for additional parameters.

See Also

Other outcome: `BinaryOutcome-class`, `ContinuousOutcome-class`, `Outcome-class`, `OutcomeBinaryLogistic-class`, `OutcomeContinuousNormal-class`, `OutcomeSurvExponential-class`, `TimeToEvent-class`

`outcome_bin_logistic` *Bernoulli distribution with logit parametrization*

Description

Bernoulli distribution with logit parametrization

Usage

```
outcome_bin_logistic(binary_var, baseline_prior, weight_var = "")
```

Arguments

`binary_var`	character. Name of binary (1/0 or TRUE/FALSE) outcome variable in the model matrix
`baseline_prior`	`Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome. See `Details` for more information.
`weight_var`	character. Optional name of variable in model matrix for weighting the log likelihood.

Details

Baseline Prior:

The `baseline_prior` argument specifies the prior distribution for the baseline log odds. The interpretation of the `baseline_prior` differs slightly between borrowing methods selected.

- *Dynamic borrowing using* `borrowing_hierarchical_commensurate()`: the `baseline_prior` for Bayesian Dynamic Borrowing refers to the log odds of the external control arm.
- *Full borrowing* or *No borrowing* using `borrowing_full()` or `borrowing_none()`: the `baseline_prior` for these borrowing methods refers to the log odds for the internal control arm.

Value

Object of class OutcomeBinaryLogistic.

See Also

Other outcome models: outcome_cont_normal(), outcome_surv_exponential(), outcome_surv_weibull_ph()

Examples

```
lg <- outcome_bin_logistic(
  binary_var = "response",
  baseline_prior = prior_normal(0, 1000)
)
```

---

outcome_cont_normal *Normal Outcome Distribution*

Description

Normal Outcome Distribution

Usage

```
outcome_cont_normal(
  continuous_var,
  baseline_prior,
  std_dev_prior,
  weight_var = ""
)
```

Arguments

continuous_var	character. Name of continuous outcome variable in the model matrix
baseline_prior	Prior. Object of class Prior specifying prior distribution for the baseline outcome. See Details for more information.
std_dev_prior	Prior. Object of class Prior specifying prior distribution for the standard deviation of the outcome distribution (i.e. "sigma").
weight_var	character. Optional name of variable in model matrix for weighting the log likelihood.

Details

Baseline Prior:

The baseline_prior argument specifies the prior distribution for the intercept of the linear model. The interpretation of the baseline_prior differs slightly between borrowing methods selected.

- *Dynamic borrowing using* `borrowing_hierarchical_commensurate()`: the `baseline_prior` for Bayesian Dynamic Borrowing refers to the intercept of the external control arm.
- *Full borrowing* or *No borrowing* using `borrowing_full()` or `borrowing_none()`: the `baseline_prior` for these borrowing methods refers to the intercept for the internal control arm.

Value

Object of class `OutcomeContinuousNormal`.

See Also

Other outcome models: `outcome_bin_logistic()`, `outcome_surv_exponential()`, `outcome_surv_weibull_ph()`

Examples

```
norm <- outcome_cont_normal(
  continuous_var = "tumor_size",
  baseline_prior = prior_normal(0, 100),
  std_dev_prior = prior_half_cauchy(1, 5)
)
```

---

outcome_surv_exponential

Exponential survival distribution

Description

Exponential survival distribution

Usage

```
outcome_surv_exponential(time_var, cens_var, baseline_prior, weight_var = "")
```

Arguments

`time_var`	character. Name of time variable column in model matrix
`cens_var`	character. Name of the censorship variable flag in model matrix
`baseline_prior`	`Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome. See `Details` for more information.
`weight_var`	character. Optional name of variable in model matrix for weighting the log likelihood.

Details

Baseline Prior:

The `baseline_prior` argument specifies the prior distribution for the baseline log hazard rate. The interpretation of the `baseline_prior` differs slightly between borrowing methods selected.

- *Dynamic borrowing using* `borrowing_hierarchical_commensurate()`: the `baseline_prior` for Bayesian Dynamic Borrowing refers to the log hazard rate of the external control arm.
- *Full borrowing* or *No borrowing* using `borrowing_full()` or `borrowing_none()`: the `baseline_prior` for these borrowing methods refers to the log hazard rate for the internal control arm.

Value

Object of class `OutcomeSurvExponential`.

See Also

Other outcome models: `outcome_bin_logistic()`, `outcome_cont_normal()`, `outcome_surv_weibull_ph()`

Examples

```
es <- outcome_surv_exponential(
  time_var = "time",
  cens_var = "cens",
  baseline_prior = prior_normal(0, 1000)
)
```

---

outcome_surv_weibull_ph

Weibull survival distribution (proportional hazards formulation)

Description

Weibull survival distribution (proportional hazards formulation)

Usage

```
outcome_surv_weibull_ph(
  time_var,
  cens_var,
  shape_prior,
  baseline_prior,
  weight_var = ""
)
```

Arguments

time_var	character. Name of time variable column in model matrix
cens_var	character. Name of the censorship variable flag in model matrix
shape_prior	Prior class object for the Weibull shape parameter. Default is prior_exponential(beta = 0.0001).
baseline_prior	Prior. Object of class Prior specifying prior distribution for the baseline outcome. See Details for more information.
weight_var	character. Optional name of variable in model matrix for weighting the log likelihood.

Details

Baseline Prior:

The baseline_prior argument specifies the prior distribution for the baseline log hazard rate. The interpretation of the baseline_prior differs slightly between borrowing methods selected.

- *Dynamic borrowing using* borrowing_hierarchical_commensurate(): the baseline_prior for Bayesian Dynamic Borrowing refers to the log hazard rate of the external control arm.
- *Full borrowing* or *No borrowing* using borrowing_full() or borrowing_none(): the baseline_prior for these borrowing methods refers to the log hazard rate for the internal control arm.

Value

Object of class OutcomeSurvWeibullPH.

See Also

Other outcome models: outcome_bin_logistic(), outcome_cont_normal(), outcome_surv_exponential()

Examples

```
ws <- outcome_surv_weibull_ph(
  time_var = "time",
  cens_var = "cens",
  shape_prior = prior_exponential(1),
  baseline_prior = prior_normal(0, 1000)
)
```

---

plot *Plot Prior Objects*

Description

Plot prior distributions as densities. Continuous distributions are plotted as curves and discrete distributions as bar plots.

Usage

```
## S4 method for signature 'Prior,missing'
plot(
  x,
  y,
  default_limits,
  dist_type = c("continuous", "discrete"),
  density_fun,
  add,
  ...
)

## S4 method for signature 'PriorNormal,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorExponential,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorHalfCauchy,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorBernoulli,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorBeta,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorCauchy,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorGamma,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorHalfNormal,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'PriorPoisson,missing'
plot(x, y, add = FALSE, ...)

## S4 method for signature 'UniformPrior,missing'
plot(x, y, add = FALSE, ...)
```

Arguments

x	Object inheriting from Prior
y	Not used.
default_limits	Numeric range to plot distribution over.

dist_type	Plot a continuous or discrete distribution.
density_fun	Function which takes a vector of values and returns a vector of density values.
add	logical. Add density to existing plot.
...	Optional arguments for plotting.

Details

Plot ranges are selected by default to show 99% of the density for unbounded distributions. The limits can be changed by specifying `xlim = c(lower, upper)`.

Colors, line types, and other typical `par()` parameters can be used.

Value

No return value, this function generates a plot in the current graphics device.

Examples

```
plot(prior_normal(1, 2))
plot(prior_exponential(0.1))
plot(prior_half_cauchy(0, 1), xlim = c(-20, 20))
plot(prior_half_cauchy(0, 2), xlim = c(-20, 20), col = 2, add = TRUE)
plot(prior_bernoulli(0.4), xlim = c(0, 15))
plot(prior_beta(2, 2))
plot(prior_cauchy(0, 1), xlim = c(-20, 20))
plot(prior_cauchy(0, 2), xlim = c(-20, 20), col = 2, add = TRUE)
plot(prior_gamma(0.1, 0.1))
plot(prior_half_normal(0, 1), xlim = c(-20, 20))
plot(prior_half_normal(0, 2), xlim = c(-20, 20), col = 2, add = TRUE)
plot(prior_poisson(5), xlim = c(0, 15))
plot(uniform_prior(1, 2), xlim = c(0, 3))
```

---

plot_pdf *Plot Probability Density Function Values*

Description

Plot Probability Density Function Values

Usage

```
plot_pdf(x, y, ...)
```

Arguments

x	values
y	probability density values `y = f(x)`
...	passed to `plot()`

Plots the density values as a curve with the lower vertical limit set to 0.

Value

No return value, this function generates a plot in the current graphics device.

Examples

```
x <- seq(-2, 2, len = 100)
y <- dnorm(x)
plot_pdf(x, y)
```

`plot_pmf` *Plot Probability Mass Function Values*

Description

Plot Probability Mass Function Values

Usage

```
plot_pmf(x, y, ..., col = "grey", add = FALSE)
```

Arguments

`x`	values
`y`	probability mass values `y = f(x)`
`...`	passed to `plot()` and `rect()`
`col`	Fill color of bars.
`add`	Add bars to existing plot.
	Plots the probability values as a barplot.

Value

No return value, this function generates a plot in the current graphics device.

Examples

```
x <- seq(0, 5)
y <- dpois(x, lambda = 2)
plot_pmf(x, y)
```

poisson_prior *Legacy function for the poisson prior*

Description

Please use `prior_poisson()` instead.

Usage

```
poisson_prior(...)
```

Arguments

`...`	Deprecated arguments to `poisson_prior()`.

Value

This function does not return a value. When called, it triggers an error message indicating that `poisson_prior()` is deprecated and that `prior_poisson()` should be used instead.

possible_data_sim_vars *Get All Variable Names in Simulated Data Model Matrix*

Description

Get All Variable Names in Simulated Data Model Matrix

Usage

```
possible_data_sim_vars(object)
```

Arguments

`object`	`BaselineObject`

Value

A vector of variable names

Prior-class Prior *Class*

Description

A class for defining priors to be translated to Stan code. Objects of class `Prior` should not be created directly but by one of the specific prior class constructors.

Slots

`stan_code` character. Stan implementation of the prior, with placeholders for parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters.

`constraint` character. Support of prior distribution expressed as a Stan constraint, e.g. `"<lower=0, upper=1>"`.

See Also

Prior constructor functions: `prior_bernoulli()`, `prior_beta()`, `prior_cauchy()`, `prior_half_cauchy()`, `prior_gamma()`, `prior_normal()`, `prior_poisson()`, `uniform_prior()`

Other prior classes: `PriorBernoulli-class`, `PriorBeta-class`, `PriorCauchy-class`, `PriorExponential-class`, `PriorGamma-class`, `PriorHalfCauchy-class`, `PriorHalfNormal-class`, `PriorNormal-class`, `PriorPoisson-class`, `UniformPrior-class`

PriorBernoulli-class PriorBernoulli *Class*

Description

A class for defining bernoulli priors to be translated to Stan code. Objects of class `PriorBernoulli` should not be created directly but by the constructor `prior_bernoulli()`.

Slots

`stan_code` character. Stan implementation of the prior, with placeholders for bernoulli stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (1).

`constraint` character. Support of prior distribution, `"<lower=0, upper=1>"`.

`theta` numeric. Probability (in [0, 1]).

See Also

Other prior classes: `Prior-class`, `PriorBeta-class`, `PriorCauchy-class`, `PriorExponential-class`, `PriorGamma-class`, `PriorHalfCauchy-class`, `PriorHalfNormal-class`, `PriorNormal-class`, `PriorPoisson-class`, `UniformPrior-class`

PriorBeta-class PriorBeta *Class*

Description

A class for defining beta priors to be translated to Stan code. Objects of class PriorBeta should not be created directly but by the constructor prior_beta().

Slots

stan_code character. Stan implementation of the prior, with placeholders for beta stan function parameters surrounded with {{ and }} to be replaced with glue::glue().

n_param integer. Number of prior parameters (2).

constraint character. Support of prior distribution, "<lower=0, upper=1>".

alpha numeric. Shape (>=0).

beta numeric. Shape (>=0).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorCauchy-class, PriorExponential-class, PriorGamma-class, PriorHalfCauchy-class, PriorHalfNormal-class, PriorNormal-class, PriorPoisson-class, UniformPrior-class

PriorCauchy-class PriorCauchy *Class*

Description

A class for defining the cauchy priors to be translated to Stan code. Objects of class PriorCauchy should not be created directly but by the constructor prior_cauchy().

Slots

stan_code character. Stan implementation of the prior, with placeholders for cauchy stan function parameters surrounded with {{ and }} to be replaced with glue::glue().

n_param integer. Number of prior parameters (2).

constraint character. Support of prior distribution, (all values allowed in cauchy distribution).

mu numeric. Location.

sigma numeric. Scale (>0).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorBeta-class, PriorExponential-class, PriorGamma-class, PriorHalfCauchy-class, PriorHalfNormal-class, PriorNormal-class, PriorPoisson-class, UniformPrior-class

PriorExponential-class PriorExponential *Class*

Description

A class for defining exponential priors to be translated to Stan code. Objects of class `PriorExponential` should not be created directly but by the constructor `prior_exponential()`.

Slots

`stan_code` character. Stan implementation of the prior, with placeholders for exponential Stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (1).

`constraint` character. Support of prior distribution, `"<lower=0>"`.

`beta` numeric. Inverse scale (>0).

See Also

Other prior classes: `Prior-class`, `PriorBernoulli-class`, `PriorBeta-class`, `PriorCauchy-class`, `PriorGamma-class`, `PriorHalfCauchy-class`, `PriorHalfNormal-class`, `PriorNormal-class`, `PriorPoisson-class`, `UniformPrior-class`

PriorGamma-class PriorGamma *Class*

Description

A class for defining gamma priors to be translated to Stan code. Objects of class `PriorGamma` should not be created directly but by the constructor `prior_gamma()`.

Slots

`stan_code` character. Stan implementation of the prior, with placeholders for gamma stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, `"<lower=0>"`.

`alpha` numeric. Shape (>0).

`beta` numeric. Inverse scale (>=0).

See Also

Other prior classes: `Prior-class`, `PriorBernoulli-class`, `PriorBeta-class`, `PriorCauchy-class`, `PriorExponential-class`, `PriorHalfCauchy-class`, `PriorHalfNormal-class`, `PriorNormal-class`, `PriorPoisson-class`, `UniformPrior-class`

PriorHalfCauchy-class PriorHalfCauchy *Class*

Description

A class for defining half cauchy priors to be translated to Stan code. Objects of class PriorHalfCauchy should not be created directly but by the constructor prior_half_cauchy().

Slots

stan_code character. Stan implementation of the prior, with placeholders for the half cauchy stan function parameters surrounded with {{ and }} to be replaced with glue::glue().

n_param integer. Number of prior parameters (2).

constraint character. Support of prior distribution. In a half cauchy prior, constraint is mu

mu numeric. Location.

sigma numeric. Scale (>0).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorBeta-class, PriorCauchy-class, PriorExponential-class, PriorGamma-class, PriorHalfNormal-class, PriorNormal-class, PriorPoisson-class, UniformPrior-class

PriorHalfNormal-class PriorHalfNormal *Class*

Description

A class for defining half normal priors to be translated to Stan code. Objects of class PriorHalfNormal should not be created directly but by the constructor prior_half_normal().

Slots

stan_code character. Stan implementation of the prior, with placeholders for the half normal stan function parameters surrounded with {{ and }} to be replaced with glue::glue().

n_param integer. Number of prior parameters (2).

constraint character. Support of prior distribution. In a half normal prior, constraint is mu

mu numeric. Location.

sigma numeric. Scale (>0).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorBeta-class, PriorCauchy-class, PriorExponential-class, PriorGamma-class, PriorHalfCauchy-class, PriorNormal-class, PriorPoisson-class, UniformPrior-class

PriorNormal-class PriorNormal *Class*

Description

A class for defining normal priors to be translated to Stan code. Objects of class PriorNormal should not be created directly but by the constructor prior_normal().

Slots

stan_code character. Stan implementation of the prior, with placeholders for normal stan function parameters surrounded with {{ and }} to be replaced with glue::glue().

n_param integer. Number of prior parameters (2).

constraint character. Support of prior distribution, (all values allowed in normal distribution).

mu numeric. Location.

sigma numeric. Scale (>0).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorBeta-class, PriorCauchy-class, PriorExponential-class, PriorGamma-class, PriorHalfCauchy-class, PriorHalfNormal-class, PriorPoisson-class, UniformPrior-class

PriorPoisson-class PriorPoisson *Class*

Description

A class for defining poisson priors to be translated to Stan code. Objects of class PriorPoisson should not be created directly but by the constructor prior_poisson().

Slots

stan_code character. Stan implementation of the prior, with placeholders for poisson stan function parameters surrounded with {{ and }} to be replaced with glue::glue().

n_param integer. Number of prior parameters (1).

constraint character. Support of prior distribution, "<lower=0>".

lambda numeric. Rate (>0).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorBeta-class, PriorCauchy-class, PriorExponential-class, PriorGamma-class, PriorHalfCauchy-class, PriorHalfNormal-class, PriorNormal-class, UniformPrior-class

prior_bernoulli *Prior bernoulli distribution*

Description

Prior bernoulli distribution

Usage

```
prior_bernoulli(theta)
```

Arguments

theta	numeric. Probability (in [0, 1]).

Details

Stan reference https://mc-stan.org/docs/functions-reference/bernoulli-distribution.html

Value

Object of class PriorBernoulli.

See Also

Other priors: prior_beta(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
bp <- prior_bernoulli(0.23)
```

prior_beta *Prior beta distribution*

Description

Prior beta distribution

Usage

```
prior_beta(alpha, beta)
```

Arguments

alpha	numeric. Shape (>=0).
beta	numeric. Shape (>=0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/beta-distribution.html

Value

Object of class PriorBeta

See Also

Other priors: prior_bernoulli(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
bp <- prior_beta(9, 235)
```

---

prior_cauchy *Prior cauchy distribution*

Description

Prior cauchy distribution

Usage

```
prior_cauchy(mu, sigma)
```

Arguments

mu	numeric. Location.
sigma	numeric. Scale (>0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/cauchy-distribution.html

Value

Object of class PriorCauchy.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
cp <- prior_cauchy(1, 1)
```

---

prior_exponential *Prior exponential distribution*

Description

Prior exponential distribution

Usage

```
prior_exponential(beta)
```

Arguments

beta	numeric. Inverse scale (>0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/exponential-distribution.html

Value

Object of class PriorExponential.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
ep <- prior_exponential(1)
```

prior_gamma *Prior gamma distribution*

Description

Prior gamma distribution

Usage

```
prior_gamma(alpha, beta)
```

Arguments

alpha	numeric. Shape (>0).
beta	numeric. Inverse scale (>=0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/gamma-distribution.html

Value

Object of class PriorGamma.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_exponential(), prior_half_cauchy(), prior_half_normal(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
gp <- prior_gamma(0.001, 0.001)
```

prior_half_cauchy *Prior half-cauchy distribution*

Description

Prior half-cauchy distribution

Usage

```
prior_half_cauchy(mu, sigma)
```

Arguments

mu	numeric. Location.
sigma	numeric. Scale (>0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/cauchy-distribution.html

Value

Object of class PriorHalfCauchy.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_normal(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
hcp <- prior_half_cauchy(1, 1)
```

---

prior_half_normal *Prior half-normal distribution*

Description

Prior half-normal distribution

Usage

```
prior_half_normal(mu, sigma)
```

Arguments

mu	numeric. Location.
sigma	numeric. Scale (>0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/normal-distribution.html

Value

Object of class PriorHalfNormal.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_normal(), prior_poisson(), uniform_prior()

Examples

```
hcp <- prior_half_normal(1, 1)
```

---

prior_normal *Prior normal distribution*

Description

Prior normal distribution

Usage

```
prior_normal(mu, sigma)
```

Arguments

mu	numeric. Location.
sigma	numeric. Scale (>0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/normal-distribution.html

Value

Object of class PriorNormal.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_poisson(), uniform_prior()

Examples

```
np <- prior_normal(1, 1)
```

prior_poisson *Prior poisson distribution*

Description

Prior poisson distribution

Usage

```
prior_poisson(lambda)
```

Arguments

lambda	numeric. Rate (>0).

Details

Stan reference https://mc-stan.org/docs/functions-reference/poisson.html

Value

Object of class PriorPoisson.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_normal(), uniform_prior()

Examples

```
pp <- prior_poisson(100)
```

rename_draws_covariates *Rename Covariates in* draws *Object*

Description

Rename Covariates in draws Object

Usage

```
rename_draws_covariates(draws, analysis)
```

Arguments

draws	draws created from sampled analysis object. See example.
analysis	Analysis as created by create_analysis_obj().

Value

A draws[posterior::draws] object with covariate names.

Examples

```
if (check_cmdstan()) {
  analysis_object <- create_analysis_obj(
    data_matrix = example_matrix,
    covariates = add_covariates(
      covariates = c("cov1", "cov2"),
      priors = prior_normal(0, 1000)
    ),
    outcome = outcome_surv_exponential(
      "time",
      "cnsr",
      baseline_prior = prior_normal(0, 1000)
    ),
    borrowing = borrowing_hierarchical_commensurate(
      "ext",
      prior_exponential(.001)
    ),
    treatment = treatment_details(
      "trt",
      prior_normal(0, 1000)
    )
  )
  samples <- mcmc_sample(analysis_object, 200, 400, 1)
  draws <- samples$draws()
  renamed_draws <- rename_draws_covariates(draws, analysis_object)
  summary(renamed_draws)
}
```

---

set_cut_off *Set Clinical Cut Off Rule*

Description

Set Clinical Cut Off Rule

Usage

```
set_cut_off(object, internal = cut_off_none(), external = cut_off_none())
```

Arguments

object	DataSimObject
internal	DataSimCutOff object specified by one of the cut off functions: cut_off_after_events(), cut_off_after_first(), cut_off_after_last(), cut_off_none().
external	DataSimCutOff for the external data.

Value

A DataSimObject with updated cut_off_internal and cut_off_external slots.

Examples

```
data_sim <- create_data_simulation(
  create_baseline_object(10, 10, 10),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
set_cut_off(
  data_sim,
  cut_off_after_events(n = 10),
  cut_off_after_first(time = 30)
)
```

---

set_dropout *Set Drop Out Distribution*

Description

Set Drop Out Distribution

Usage

```
set_dropout(object, internal_treated, internal_control, external_control)
```

Arguments

object	DataSimObject
internal_treated	DataSimEvent object specifying distribution for internal treated patients.
internal_control	DataSimEvent object specifying distribution for internal control patients.
external_control	DataSimEvent object specifying distribution for external control patients.

Details

DataSimEvent objects can be specified with create_event_dist(). Currently no beta parameters can be used in drop out distributions (unlike for the survival outcome).

Value

A DataSimObject with updated internal_treated, internal_control and external_control slots.

Examples

```
data_sim <- create_data_simulation(
  create_baseline_object(10, 10, 10),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
set_dropout(
  data_sim,
  internal_treated = create_event_dist(dist = "exponential", lambdas = 1 / 55),
  internal_control = create_event_dist(dist = "exponential", lambdas = 1 / 50),
  external_control = create_event_dist(dist = "exponential", lambdas = 1 / 40)
)
```

---

set_enrollment *Set Enrollment Rates for Internal and External Trials*

Description

Set Enrollment Rates for Internal and External Trials

Usage

```
set_enrollment(object, internal, external = internal)
```

Arguments

object	A DataSimObject from create_data_simulation
internal	DataSimEnrollment object to define the enrollment times for internal data
external	DataSimEnrollment object to define the enrollment times for external data. Defaults to be the same as internal.

Value

A DataSimObject with updated enrollment_internal and enrollment_external slots.

Examples

```
data_sim <- create_data_simulation(
  create_baseline_object(10, 10, 10),
  event_dist = create_event_dist(dist = "exponential", lambdas = 1 / 36)
)
set_enrollment(
  data_sim,
  internal = enrollment_constant(rate = c(10, 5), for_time = c(6, 6)),
  external = enrollment_constant(rate = c(5), for_time = c(20))
)
```

set_transformations *Set transformations in* BaselineObject *objects*

Description

Set transformations in BaselineObject objects

Usage

```
set_transformations(object, ..., overwrite = FALSE)
```

Arguments

object	BaselineObject object
...	Additional arguments passed to methods
overwrite	logical. Overwrite existing transformations?

Value

BaselineObject object with transformations

set_transformations,BaselineObject-method *Set Transformations in Baseline Objects*

Description

Set Transformations in Baseline Objects

Usage

```
## S4 method for signature 'BaselineObject'
set_transformations(object, ..., overwrite = FALSE)
```

Arguments

object	BaselineObject created by create_baseline_object.
...	named transformation functions. See details for more information.
overwrite	If TRUE overwrite existing transformation list and only include newly specified transformations.

Details

Transformation functions are evaluated in order and create or overwrite a column in the data.frame with that name. The function should have signature `function(data)`, taking a `data.frame` (specifically a `BaselineDataFrame` object from `generate(BaselineObject)`) and return a vector with length identical to the total number of patients. The `@BaselineObject` slot of the BaselineDataFrame may be accessed directly or with `get_quantiles()` to create transformations. See `binary_cutoff()`.

Value

An updated `BaselineObject`

Examples

```
baseline <- create_baseline_object(
  100, 50, 100,
  covariates = baseline_covariates(
    names = "age", means_int = 55,
    covariance_int = covariance_matrix(5)
  )
)
set_transformations(baseline, age_scaled = function(data) scale(data$age))
```

---

show_guide *Show guide for objects with guides*

Description

Show the guide in `Simulation` objects.

Usage

```
show_guide(object)

## S4 method for signature 'Simulation'
show_guide(object)
```

Arguments

`object`	`Simulation` object

Value

A `data.frame` showing all simulation scenarios.

SimBorrowingList-class SimBorrowingList *Class*

Description

A class for borrowing details as part of a simulation study. Objects of class `SimBorrowingList` should not be created directly but by the constructor `sim_borrowing_list()`.

Slots

`borrowing_list` named list of object of class `Borrowing`, one object for each parameter variation.

SimCovariateList-class SimCovariateList *Class*

Description

A class for covariate details as part of a simulation study. Objects of class `SimCovariateList` should not be created directly but by the constructor `sim_covariate_list()`.

Slots

`covariate_list` named list of object of class `Covariate`, one object for each parameter variation.

SimCovariates-class SimCovariates *Class*

Description

A class for specifying covariate distributions and covariance for simulation studies.

Slots

`covariates` list. List of covariate mean values or probabilities as generated through `bin_var()` (class `SimVarBin` or `cont_var()` (class `SimVarCont`).

`covariance_internal` matrix. Covariance matrix before binarization for internal patients.

`covariance_external` matrix. Covariance matrix before binarization for external patients.

SimDataList-class SimDataList *Class*

Description

A class for defining generated data for use as part of a simulation study. Objects of class `SimDataList` should not be created directly but by the constructor `sim_data_list()`.

Slots

`data_list` list of lists of matrices. The lists at the highest level differ in that the parameters used to generate the data. The matrices at lowest level are different iterations of the same data generation parameters.

`guide` data.frame. `guide` contains information on the parameters that differ at the highest level of `data_list`.

`effect` character. The column in `guide` that corresponds to the true treatment effect estimate (hazard ratio or odds ratio).

`drift` character. The column in `guide` that corresponds to the drift between external and internal control arms. A drift >1 means the external arm experiences greater effects.

`index` character. The column in `guide` that corresponds to the index of the parameter situations in `data_list`.

SimOutcomeList-class SimOutcomeList *Class*

Description

A class for outcome details as part of a simulation study. Objects of class `SimOutcomeList` should not be created directly but by the constructor `sim_outcome_list()`.

Slots

`outcome_list` named list of object of class `Outcome`, one object for each parameter variation.

SimSampleSize-class SimSampleSize *Class*

Description

A class for creating matrices for simulation studies containing flags specifying whether the patient is from the concurrent trial or not (`ext` = 0 for concurrent trial, `ext` = 1 for historical data) and whether the patient is on the experimental therapy or not (`trt` = 0 for no experimental therapy, `trt` = 1 for experimental therapy).

Slots

`n_internal_control` integer. Number of patients to be simulated in the internal control arm.

`n_external_control` integer. Number of patients to be simulated in the external control arm.

`n_internal_experimental` integer. Number of patients to be simulated in the internal experimental arm.

`mat` matrix. Matrix with two columns, `ext` (flag for being from external data source) and `trt` (flag for receiving experimental treatment)

SimTreatmentList-class SimTreatmentList *Class*

Description

A class for treatment details as part of a simulation study. Objects of class `SimTreatmentList` should not be created directly but by the constructor `sim_treatment_list()`.

Slots

`treatment_list` named list of object of class `Treatment`, one object for each parameter variation.

Simulation-class Simulation *Class*

Description

A class for defining Simulation study details. Objects of class Simulation should not be created directly but by the constructor create_simulation_obj().

Slots

data_matrix_list SimDataList. The list of lists of data matrices created with sim_data_list().

outcome SimOutcomeList. List of Outcome objects created with sim_outcome_list().

borrowing SimBorrowingList. List of Borrowing objects created with sim_borrowing_list().

covariate SimCovariateList or NULL. List of Covariate objects created with sim_covariate_list() or NULL (no covariate adjustment).

treatment SimTreatmentList. List of Treatment objects created with sim_treatment_list().

guide data.frame. Data.frame containing information on all combinations evaluated.

n_combos integer. Number of combinations of parameters to be evaluated.

n_analyses integer. Number of analyses (combos x datasets to be performed).

'analysis_obj_list' list. List of analysis objects indexed according to guide.

SimVar-class SimVar *Class*

Description

A parent class for defining covariates to be created in the simulation study calls to add_covariates().

SimVarBin-class SimVarBin *class*

Description

A constructor for making objects of class SimVarBin. Objects of class SimVarBin are used to hold proportions of binary variables specified in a simulation study.

Slots

`prob_internal` numeric. Proportion for the internal arms.

`prob_external` numeric. Proportion for the external arm.

`mu_internal_before_bin` numeric. Mean value of covariate before binarization for the internal arms.

`mu_external_before_bin` numeric. Mean value of covariate before binarization for the external arm.

`printval_int` numeric. Value to print to summarize internal arms.

`printval_ext` numeric. Value to print to summarize external arm.

`type_string` character. 'binary'

See Also

Other simvar classes: `SimVarCont-class`

`SimVarCont-class` `SimVarCont` *class*

Description

A constructor for making objects of class `SimVarCont`. Objects of class `SimVarCont` are used to hold mean values of of continuous variables specified in a simulation study.

Slots

`mu_internal` numeric. Mean covariate value for the internal arms.

`mu_external` numeric. Mean covariate value for the external arm.

`printval_int` numeric. Value to print to summarize internal arms.

`printval_ext` numeric. Value to print to summarize external arm.

`type_string` character. 'continuous'

See Also

Other simvar classes: `SimVarBin-class`

`sim_borrowing_list` *Input borrowing details for a simulation study*

Description

A function for defining which borrowing scenarios should be evaluated as part of a simulation study.

Usage

```
sim_borrowing_list(borrowing_list)
```

Arguments

`borrowing_list`	named list of objects of class `Borrowing` created by `borrowing_full()`, `borrowing_none()`, or `borrowing_hierarchical_commensurate()`.

Value

Object of class `SimBorrowingList`.

See Also

Other simulation classes: `sim_covariate_list()`, `sim_data_list()`, `sim_outcome_list()`, `sim_treatment_list()`

Examples

```
borrow_scenarios <- sim_borrowing_list(
  list(
    "No borrowing" = borrowing_none("ext"),
    "Full borrowing" = borrowing_full("ext"),
    "BDB, uninformative prior" = borrowing_hierarchical_commensurate(
      "ext",
      prior_gamma(0.001, 0.001)
    ),
    "BDB, informative prior" = borrowing_hierarchical_commensurate(
      "ext",
      prior_gamma(1, 0.001)
    )
  )
)
```

`sim_covariates` *Specify covariates for simulation study*

Description

Provide details on the desired covariate distributions and covariance for for a simulation study.

Usage

```
sim_covariates(
  covariates,
  covariance_internal,
  covariance_external = covariance_internal
)
```

Arguments

`covariates`	list. Named list of covariate mean values or probabilities as generated through `bin_var()` (class `SimVarBin` or `cont_var()` (class `SimVarCont`). See `details` for more information.
`covariance_internal`	matrix. Covariance matrix before binarization for internal patients.
`covariance_external`	matrix. Covariance matrix before binarization for external patients. Defaults to the internal covariance.

Details

This function is intended to specify the number of covariates and relationships between them for the purposes of designing a simulation study in `psborrow2`. Because the outcome model does not necessarily need to adjust for covariates, this function is not necessary in `create_simulation_obj()`. The relationship between the treatment and the outcome is specified elsewhere (i.e, in `sim_survival()` or `sim_binary_event()`).

We need a few things to

Value

Object of class `SimCovariates`

See Also

Other simulation: `sim_samplesize()`

Examples

```
set.seed(123)
covmat <- matrix(rWishart(1, 2, diag(2)), ncol = 2)

covset1 <- sim_covariates(
  covariates = list(
    cov1 = bin_var(0.5, 0.5),
    cov2 = cont_var(100, 130)
  ),
  covariance_internal = covmat
)
```

---

sim_covariates_summ *Summarize the number of continuous and binary covariates in a* SimCovariates *object created by* sim_covariates()

Description

Summarize the number of continuous and binary covariates in a SimCovariates object created by sim_covariates()

Usage

```
sim_covariates_summ(sim_covariates_obj)
```

Arguments

sim_covariates_obj
: SimCovariates. Object returned by sim_covariates().

Value

data.frame showing covariate names and types as well as counts of binary and continuous covariates.

---

sim_covariate_list *Input covariate adjustment details for a simulation study*

Description

A function for defining which covariate adjustment scenarios should be evaluated as part of a simulation study.

Usage

```
sim_covariate_list(covariate_list)
```

Arguments

`covariate_list` named list of objects of class `Covariate` created by `add_covariates()`.

Details

This function allows the user to specify covariate adjustment details that will be included as part of a simulation study. It is often of interest to compare several adjustment methods to no adjustment. To specify no adjustment, pass `NULL` as a list item to `covariate_list`.

Value

Object of class `SimCovariateList`.

See Also

Other simulation classes: `sim_borrowing_list()`, `sim_data_list()`, `sim_outcome_list()`, `sim_treatment_list()`

Examples

```
covariates <- sim_covariate_list(
  list(
    "No adjustment" = NULL,
    "Covariates 1 and 2" = add_covariates(c("cov1", "cov2"), prior_normal(0, 1000))
  )
)
```

`sim_data_list` *Input generated data for a simulation study*

Description

A function for defining generated data for use as part of a simulation study.

Usage

```
sim_data_list(data_list, guide, effect, drift, index)
```

Arguments

`data_list`	list of lists of matrices. The lists at the highest level differ in that the parameters used to generate the data. The matrices at lowest level are different iterations of the same data generation parameters. See `details`.
`guide`	data.frame. `guide` contains information on the parameters that differ at the highest level of `data_list`. See `details`.
`effect`	character. The column in `guide` that corresponds to the true treatment effect estimate (hazard ratio or odds ratio).

drift	character. The column in guide that corresponds to the true drift effect estimate (hazard ratio or odds ratio). A drift >1 means the external arm experiences greater effects.
index	character. The column in guide that corresponds to the index column.

Details

In this function, you are providing generated data for analysis in a simulation study in psborrow2. Note that this function does not do any data generation on your behalf; it assumes that you have generated the data already. For a full working example, refer to the relevant vignette: `vignette('simulation_study', package = 'psborrow2')`.

More information on the inputs is provided below.

Matrix requirements in `data_list`:

Each matrix embedded in `data_list` must have:

1. a flag for whether the patient is an external control
2. a flag for whether the patient is in the experimental treatment arm
3. outcome information (time and censorship for survival, flag for outcome in binary endpoints)

Optionally, the matrices may also contain covariates. See `examples`.

`data_list`**:**

Each set of distinct data generation parameters should be represented by a single list of matrices. Because multiple scenarios may want to be compared, a list of list of matrices is preferred. See `examples`.

`guide`**:**

The `guide` should be a data.frame with one row per scenario. As a consquence of this, the length of the list should equal the number of rows in the guide. See `examples`.

Value

Object of class `SimDataList`.

See Also

Other simulation classes: `sim_borrowing_list()`, `sim_covariate_list()`, `sim_outcome_list()`, `sim_treatment_list()`

Examples

```
base_mat <- matrix(
  c(
    rep(0, 200), rep(0, 200), rep(1, 200),
    rep(1, 200), rep(0, 200), rep(0, 200),
    rep(0, 600)
  ),
  ncol = 3,
  dimnames = list(NULL, c("ext", "trt", "driftOR"))
)
```

```
add_binary_endpoint <- function(odds_ratio,
                                base_matrix = base_mat) {
  linear_predictor <- base_matrix[, "trt"] * log(odds_ratio)
  prob <- 1 / (1 + exp(-linear_predictor))

  bin_endpoint <- rbinom(
    NROW(base_matrix),
    1,
    prob
  )

  cbind(base_matrix, matrix(bin_endpoint, ncol = 1, dimnames = list(NULL, "ep")))
}

data_list <- list(
  list(add_binary_endpoint(1.5), add_binary_endpoint(1.5)),
  list(add_binary_endpoint(2.5), add_binary_endpoint(2.5))
)

guide <- data.frame(
  trueOR = c(1.5, 2.5),
  driftOR = c(1.0, 1.0),
  ind = c(1, 2)
)

sdl <- sim_data_list(
  data_list = data_list,
  guide = guide,
  effect = "trueOR",
  drift = "driftOR",
  index = "ind"
)
```

`sim_outcome_list` *Input outcome details for a simulation study*

Description

A function for defining which outcome scenarios should be evaluated as part of a simulation study.

Usage

```
sim_outcome_list(outcome_list)
```

Arguments

`outcome_list`	named list of objects of class `Outcome` created by `outcome_details()`.

Value

Object of class SimOutcomeList.

See Also

Other simulation classes: sim_borrowing_list(), sim_covariate_list(), sim_data_list(), sim_treatment_list()

Examples

```
outcome_scenarios <- sim_outcome_list(
  list(
    "Exponential" = outcome_surv_exponential("time", "cnsr", prior_normal(0, 10000))
  )
)
```

---

sim_samplesize *Set simulation study parameters for sample size*

Description

Set simulation study parameters for sample size

Usage

```
sim_samplesize(n_internal_control, n_external_control, n_internal_experimental)
```

Arguments

n_internal_control	integer. Number of patients to be simulated in the internal control arm.
n_external_control	integer. Number of patients to be simulated in the external control arm.
n_internal_experimental	integer. Number of patients to be simulated in the internal experimental arm.

Value

Object of class SimSampleSize

See Also

Other simulation: sim_covariates()

Examples

```
ss <- sim_samplesize(200, 200, 500)
```

sim_treatment_list *Input treatment details for a simulation study*

Description

A function for defining which treatment scenarios should be evaluated as part of a simulation study.

Usage

```
sim_treatment_list(treatment_list)
```

Arguments

`treatment_list`	named list of objects of class `Treatment` created by `treatment_details()`.

Value

Object of class `SimTreatmentList`.

See Also

Other simulation classes: `sim_borrowing_list()`, `sim_covariate_list()`, `sim_data_list()`, `sim_outcome_list()`

Examples

```
treatment_scenarios <- sim_treatment_list(
  list(
    "Standard" = treatment_details("trt", prior_normal(0, 1000))
  )
)
```

TimeToEvent-class `TimeToEvent` *class*

Description

`TimeToEvent` class

Slots

`function_stan_code` character. Code to include in the Stan functions program block.

`param_stan_code` character. Code to include in the Stan parameters program block.

`likelihood_stan_code` character. Code defining the likelihood to include in the Stan model program block.

`data_stan_code` character. Code to include in the Stan data program block.

`n_param` integer. Number of ancillary parameters for the model to estimate.

`param_priors` list. Named list of prior distributions on the ancillary parameters in the model.

`time_var` character. Variable used for time in `TimeToEvent` objects.

`cens_var` character. Variable used for censoring in `TimeToEvent` objects.

`baseline_prior` `Prior`. Object of class `Prior` specifying prior distribution for the baseline outcome.

`name_beta_trt.` Named vector for beta_trt.

`name_exp_trt.` Named vector for exponentiated beta_trt

`alpha_type.` How to interpret alpha.

`name_addnl_params.` Named vector for additional parameters.

See Also

Other outcome: `BinaryOutcome-class`, `ContinuousOutcome-class`, `Outcome-class`, `OutcomeBinaryLogistic-class`, `OutcomeContinuousNormal-class`, `OutcomeSurvExponential-class`, `OutcomeSurvWeibullPH-class`

`Treatment-class` `Treatment` *Class*

Description

A class for defining treatment details. Objects of class `Treatment` should not be created directly but by the constructor `treatment_details()`.

Slots

`trt_flag_col` character. Character specifying the name of the column in the model matrix that corresponds to the treatment flag (`1`/`0` or `TRUE`/`FALSE`). This identifies patients as belonging to the experimental treatment arm.

`trt_prior` `Prior`. Object of class `Prior` specifying the prior distribution of the log effect estimate (log hazard ratio for time to event endpoints and log odds ratio for binary endpoints).

treatment_details *Specify Treatment Details*

Description

Specify the treatment arm column name in the model matrix and set a prior distribution for the treatment effect (log hazard ratio or log odds ratio)

Usage

```
treatment_details(trt_flag_col, trt_prior)
```

Arguments

trt_flag_col	character. The name of the column in the model matrix that corresponds to the treatment flag (1/0 or TRUE/FALSE). This identifies patients as belonging to the experimental treatment arm.
trt_prior	Object of class Prior specifying the prior distribution of the log effect estimate (log hazard ratio for time to event endpoints and log odds ratio for binary endpoints).

Value

Object of class Treatment.

Examples

```
sta <- treatment_details(
  trt_flag_col = "trt",
  trt_prior = prior_normal(0, 1000)
)
```

trim_cols *Trim columns from Data Matrix Based on Borrowing object type*

Description

Trim columns from Data Matrix Based on Borrowing object type

Usage

```
trim_cols(borrowing_object, analysis_object)

## S4 method for signature 'Borrowing'
trim_cols(borrowing_object, analysis_object)

## S4 method for signature 'BorrowingHierarchicalCommensurate'
trim_cols(borrowing_object, analysis_object)
```

Arguments

borrowing_object
: borrowing object

analysis_object
: analysis object

---

trim_rows *Trim Rows from Data Matrix Based on Borrowing object type*

Description

Trim Rows from Data Matrix Based on Borrowing object type

Usage

```
trim_rows(borrowing_object, analysis_object)

## S4 method for signature 'Borrowing'
trim_rows(borrowing_object, analysis_object)

## S4 method for signature 'BorrowingNone'
trim_rows(borrowing_object, analysis_object)
```

Arguments

borrowing_object
: borrowing object

analysis_object
: analysis object

UniformPrior-class `UniformPrior` *Class*

Description

A class for defining uniform priors to be translated to Stan code. Objects of class `UniformPrior` should not be created directly but by the constructor `uniform_prior()`.

Slots

`stan_code` character. Stan implementation of the prior, with placeholders for uniform stan function parameters surrounded with `{{` and `}}` to be replaced with `glue::glue()`.

`n_param` integer. Number of prior parameters (2).

`constraint` character. Support of prior distribution, `"<lower=alpha, upper=beta>"`.

`alpha` numeric. Lower bound.

`beta` numeric. Upper bound (`>alpha`).

See Also

Other prior classes: Prior-class, PriorBernoulli-class, PriorBeta-class, PriorCauchy-class, PriorExponential-class, PriorGamma-class, PriorHalfCauchy-class, PriorHalfNormal-class, PriorNormal-class, PriorPoisson-class

---

uniform_prior *Prior uniform distribution*

Description

Prior uniform distribution

Usage

```
uniform_prior(alpha, beta)
```

Arguments

alpha	numeric. Lower bound.
beta	numeric. Upper bound (>alpha).

Details

Stan reference https://mc-stan.org/docs/functions-reference/uniform-distribution.html

Value

Object of class UniformPrior.

See Also

Other priors: prior_bernoulli(), prior_beta(), prior_cauchy(), prior_exponential(), prior_gamma(), prior_half_cauchy(), prior_half_normal(), prior_normal(), prior_poisson()

Examples

```
up <- uniform_prior(0, 1)
```

variable_dictionary *Create Variable Dictionary*

Description

Create Variable Dictionary

Usage

```
variable_dictionary(analysis_obj)
```

Arguments

analysis_obj	Analysis. Object to describe variable names.

Value

A data.frame with the names of Stan variables and the descriptions.

weib_ph_surv_dist *Legacy function for the Weibull proportional Hazards survival distribution*

Description

Please use outcome_surv_weibull_ph() instead.

Usage

```
weib_ph_surv_dist(...)
```

Arguments

...	Deprecated arguments to weib_ph_surv_dist().

Value

This function does not return a value. When called, it triggers an error message indicating that weib_ph_surv_dist() is deprecated and that outcome_surv_weibull_ph() should be used instead.

Index