

# Package: psborrow (via r-universe)

August 11, 2024

**Title** Bayesian Dynamic Borrowing with Propensity Score

**Version** 0.2.1

**Description** A tool which aims to help evaluate the effect of external borrowing using an integrated approach described in Lewis et al., (2019) <[doi:10.1080/19466315.2018.1497533](https://doi.org/10.1080/19466315.2018.1497533)> that combines propensity score and Bayesian dynamic borrowing methods.

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.2

**Imports** dplyr, data.table, rjags, mvtnorm, ggplot2 (>= 3.2.0),  
foreach, doParallel, parallel, MatchIt, survival,  
futile.logger, methods

**Suggests** knitr, rmarkdown, testthat, Matrix, assertthat, pkgload,  
flexsurv, R.rsp

**VignetteBuilder** R.rsp

**Collate** 'zzz.R' 'add\_cov.R' 'add\_time.R' 'add\_mcmc.R' 'apply\_mcmc.R'  
'simu\_cov.R' 'match\_cov.R' 'simu\_time.R' 'run\_mcmc.R'  
'get\_summary.R' 'utils.R'

**Depends** R (>= 3.5.0)

**Config/testthat/edition** 3

**License** Apache License (>= 2)

**Repository** <https://pharmaverse.r-universe.dev>

**RemoteUrl** <https://github.com/Genentech/psborrow>

**RemoteRef** HEAD

**RemoteSha** 37af85b2490c146aa63a437cf3efadba8b8b065e

## Contents

.clinClass-class	2
.covClass-class	2

.eventClass-class . . . . .	3
.priorClass-class . . . . .	3
apply_mcmc . . . . .	3
c,.covClass-method . . . . .	4
c,.priorClass-method . . . . .	5
fix_col_names . . . . .	6
get_summary . . . . .	6
is_psborrow_dev . . . . .	7
match_cov . . . . .	7
plot_bias . . . . .	8
plot_hr . . . . .	8
plot_mse . . . . .	9
plot_power . . . . .	10
plot_typeIerror . . . . .	10
ps_message . . . . .	11
rej_est . . . . .	11
run_mcmc . . . . .	12
run_mcmc_p . . . . .	12
set_clin . . . . .	13
set_cov . . . . .	15
set_event . . . . .	16
set_n . . . . .	17
set_prior . . . . .	18
simu_cov . . . . .	19
simu_time . . . . .	20

**Index** **22**

---

.clinClass-class	<i>S4 Class for specifying parameters for enrollment time, drop-out pattern and analysis start time</i>
------------------	---

---

**Description**

S4 Class for specifying parameters for enrollment time, drop-out pattern and analysis start time

---

.covClass-class	<i>S4 Class for setting up covariates</i>
-----------------	---

---

**Description**

S4 Class for setting up covariates

---

.eventClass-class      *S4 Class for setting parameters for time-to-events*

---

**Description**

S4 Class for setting parameters for time-to-events

---

.priorClass-class      *S4 Class for specifying prior distributions and predictors for MCMC methods*

---

**Description**

S4 Class for specifying prior distributions and predictors for MCMC methods

---

apply\_mcmc              *Fit Dynamic Borrowing MCMC Model*

---

**Description**

Fit a dynamic borrowing Weibull survival model to the given dataset and extract the posterior samples using MCMC. See the user guide for more information on the model formulation. See [run\\_mcmc\(\)](#) for more information on the available parameters for tuning the MCMC sampling process

**Usage**

```

apply_mcmc(dt, formula_cov, ...)

extract_samples(object)

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

```

**Arguments**

- dt                      A data.frame containing data required for modelling. See details
- formula\_cov          A one sided formula specifying which non-treatment covariates should be included into the model. See details
- ...                    Additional arguments passed onto [run\\_mcmc\(\)](#). Only exception being the path argument which is not supported by this function
- object                A apply\_mcmc object created by [apply\\_mcmc\(\)](#)

## Details

`apply_mcmc()`:

The `dt` data.frame must contain 1 row per subject with the following variables:

- **time** - A continuous non-zero number specifying the time that the subject had an event at
- **cnsr** - A column of 0/1's where 1 indicates that the event was censored/right truncated
- **ext** - A column of 0/1's where 1 indicates that the subject was part of the external control
- **trt** - A column of 0/1's where 1 indicates that the subject was receiving the experimental treatment

The `dt` data.frame may also contain any additional covariates to be used in the Weibull model as specified by `formula_cov`. In order to fit a valid model `formula_cov` must contain the intercept term. The formula will be automatically adjusted to include the treatment term and as such should not be included here, if you want to include a treatment interaction term this should be done by using `~ trt:covariate` and NOT via `~ trt*covariate`.

`extract_samples()`:

This function can be used to extract the samples generated by `apply_mcmc()`

`summary()`:

This function provides summary statistics about the samples generated by `apply_mcmc()`

### Extracted Samples:

The extracted samples can be roughly defined as follows (see the user guide for full details):

- `HR_cc_hc` - The hazard ratio between the concurrent control arm and the historical control arm. This can be thought of as the ratio of the scale parameter between the baseline trial distribution and the baseline external control distribution. This is equivalent to  $\exp(\alpha[2] - \alpha[1])$
- `HR_trt_cc` - The hazard ratio between the treatment arm and the concurrent control arm. This is equivalent to  $\exp(\beta_{trt})$
- `alpha[1]` - The shape parameter for the trial's baseline distribution
- `alpha[2]` - The shape parameter for the historical control's baseline distribution
- `beta_trt` - The log-hazard ratio for the treatment effect. This is equivalent to  $\log(\text{HR}_{trt\_cc})$
- `beta_<var>` - The log-hazard ratio for any other covariate provided to the model via `formula_cov`
- `r0` - The scale parameter for the baseline distribution of both the trial and the historical control
- `tau/sigma` - The precision/variance for `alpha[1]` i.e. controls how much information is borrowed from the historical control arm

---

`c,.covClass-method`      *Concatenate multiple .covClasss classes*

---

## Description

Concatenate multiple `.covClasss` classes

**Usage**

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

**Arguments**

x                    A .covClasss class with covariate information generated in [set\\_cov](#)  
 ...                  Other .covClasss classes with covariate information generated in [set\\_cov](#)

**Value**

A vector of .covClasss classes

**Examples**

```
# combine two sets of covariates
covset1 = set_cov(n_cat = 2, n_cont = 0, mu_int = 0, mu_ext = 0, var = 1)
covset2 = set_cov(n_cat = 0, n_cont = 1, mu_int = 62, mu_ext = 65, var = 11)
cov_list = c(covset1, covset2)
```

---

c,.priorClass-method    *Concatenate multiple .priorClasss class*

---

**Description**

Concatenate multiple .priorClasss class

**Usage**

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

**Arguments**

x                    A .priorClasss class with prior distribution information generated in [set\\_prior](#)  
 ...                  A .priorClasss class with prior distribution information generated in [set\\_prior](#)

**Value**

A vector of .priorClasss classes

fix\_col\_names

*Fix Column Names*

---

**Description**

Utility function to make the mcmc column names more human friendly

**Usage**

```
fix_col_names(x, column_names)
```

**Arguments**

`x` a mcmc results object created by `add_mcmc()`  
`column_names` The names to change the beta columns to

---

get\_summary

*Generate summary statistics of a simulation scenario*

---

**Description**

Generate summary statistics of a simulation scenario

**Usage**

```
get_summary(dt)
```

**Arguments**

`dt` a `data.frame` containing summary statistics for the posterior samples from each simulation

**Value**

a `data.frame` containing the mean and sd of posterior HR between treatment and control arm, the posterior mean and sd of HR between internal control and external control arm, reject rate, variance, bias and mse of the simulation set

---

is_psborrow_dev	<i>Check if user is in psborrow development environment</i>
-----------------	---

---

**Description**

Simple function which leverages the DESCRIPTION file to check if the user is in a development environment for psborrow.

**Usage**

```
is_psborrow_dev()
```

**Value**

TRUE/FALSE flag (TRUE = in development environment)

---

match_cov	<i>Match</i>
-----------	--------------

---

**Description**

Match

**Usage**

```
match_cov(dt, match)
```

**Arguments**

dt	a list of matrix
match	A vector of covariates name to match on

**Value**

a list of matrix containing matched cohort information

**Examples**

```
# match internal and external trial data using different covariates
smp = set_n(ssC = 140, ssE = 275, ssExt = 100)
covset1 = set_cov(n_cat = 2, n_cont = 0, mu_int = 0, mu_ext = 0, var = 1)
covset2 = set_cov(n_cat = 0, n_cont = 1, mu_int = 62, mu_ext = 65, var = 11)
cObj = c(covset1, covset2)
sample_cov <-
  simu_cov(ssObj = smp, covObj = cObj, HR = 1, driftHR = 1.2, nsim = 2)

# match on covariates 1 and 2
```

```

match_cov(dt = sample_cov, match = c("cov1", "cov2"))

# match on all 3 covariates
match_cov(dt = sample_cov, match = c("cov1", "cov2", "cov3"))

```

---

plot\_bias

*Plot bias*

---

### Description

Plot bias for each prior distribution according to selected simulation parameters

### Usage

```
plot_bias(dt, HR = 1, driftHR = 1, pred = "none")
```

### Arguments

dt	a data.frame containing summary statistics for the posterior samples from each simulation generated with get_summary.
HR	pre-specified HR between treatment and control arm in the internal trial for which the bias should be plotted. Must be within unique(dt\$HR).
driftHR	pre-specified HR between external control arm and internal control arm for which the bias should be plotted. Must be within unique(dt\$driftHR).
pred	predictors to use when fitting exponential distribution in MCMC for which the bias should be plotted. Must be within unique(dt\$pred).

### Value

a bar plot of class ggplot containing the bias for each prior distribution.

---

plot\_hr

*Plot mean posterior hazard ratio between treatment and control*

---

### Description

Plot mean posterior hazard ratio between treatment and control

### Usage

```
plot_hr(dt, HR = 0.67, driftHR = 1, pred = "none")
```



**Arguments**

dt	a data.frame containing summary statistics for the posterior samples from each simulation generated with get_summary.
HR	pre-specified HR between treatment and control arm in the internal trial for which the mean posterior hazard ratio should be plotted. Must be within unique(dt\$HR).
driftHR	pre-specified HR between external control arm and internal control arm for which the mean posterior hazard ratio should be plotted. Must be within unique(dt\$driftHR).
pred	predictors to use when fitting exponential distribution in MCMC for which the mean posterior hazard ratio should be plotted. Must be within unique(dt\$pred).

**Value**

a plot of class ggplot containing the mean posterior hazard ratio for each prior distribution.

---

plot_mse	<i>Plot mean squared error (MSE)</i>
----------	--------------------------------------

---

**Description**

Plot mean squared error (MSE) for each prior distribution according to selected simulation parameters

**Usage**

```
plot_mse(dt, HR = 1, driftHR = 1, pred = "none")
```

**Arguments**

dt	a data.frame containing summary statistics for the posterior samples from each simulation generated with get_summary.
HR	pre-specified HR between treatment and control arm in the internal trial for which the MSE should be plotted. Must be within unique(dt\$HR).
driftHR	pre-specified HR between external control arm and internal control arm for which the MSE should be plotted. Must be within unique(dt\$driftHR).
pred	predictors to use when fitting exponential distribution in MCMC for which the MSE should be plotted. Must be within unique(dt\$pred).

**Value**

a bar plot of class ggplot containing the MSE for each prior distribution.

---

plot_power	<i>Plot power</i>
------------	-------------------

---

**Description**

Plot power for each prior distribution according to selected simulation parameters

**Usage**

```
plot_power(dt, HR = 0.67, driftHR = 1, pred = "none")
```

**Arguments**

dt	a data.frame containing summary statistics for the posterior samples from each simulation generated with <code>get_summary</code> .
HR	pre-specified HR between treatment and control arm in the internal trial for which the power should be plotted. Must be within <code>unique(dt\$HR)</code> .
driftHR	pre-specified HR between external control arm and internal control arm for which the power should be plotted. Must be within <code>unique(dt\$driftHR)</code> .
pred	predictors to use when fitting exponential distribution in MCMC for which the power should be plotted. Must be within <code>unique(dt\$pred)</code> .

**Value**

a bar plot of class `ggplot` containing the power for each prior distribution.

---

plot_type1error	<i>Plot type 1 error</i>
-----------------	--------------------------

---

**Description**

Plot type 1 error for each prior distribution according to selected simulation parameters

**Usage**

```
plot_type1error(dt, driftHR = 1, pred = "none")
```

**Arguments**

dt	a data.frame containing summary statistics for the posterior samples from each simulation generated with <code>get_summary()</code> . Must contain simulations for <code>HR = 1.0</code> .
driftHR	the driftHR between the external and internal control arms for which the type 1 error should be plotted. Must be within <code>unique(dt\$driftHR)</code> .
pred	the predictors used when fitting the exponential distribution in MCMC for which the type 1 error should be plotted. Must be within <code>unique(dt\$pred)</code> .

**Value**

a bar plot of class `ggplot` containing type 1 error proportions for each prior distribution.

---

ps_message	<i>Conditional Message</i>
------------	----------------------------

---

**Description**

Simple wrapper function around `message()` that will suppress printing messages if the option `psborrow.quiet` is set to `TRUE` i.e.

```
options("psborrow.quiet" = TRUE)
```

**Usage**

```
ps_message(...)
```

**Arguments**

... Values passed onto `message()`

---

rej_est	<i>Generate summary statistics for the MCMC chains</i>
---------	--

---

**Description**

Generate summary statistics for the MCMC chains

**Usage**

```
rej_est(samples)
```

**Arguments**

samples an object of class `mcmc.list`

**Value**

a vector containing the mean, median, sd, reject rate for the MCMC chains

---

run_mcmc	<i>Run MCMC for multiple scenarios with provided data</i>
----------	---

---

**Description**

Run MCMC for multiple scenarios with provided data

**Usage**

```
run_mcmc(dt, priorObj, n.chains, n.adapt, n.burn, n.iter, seed, path)
```

**Arguments**

dt	a list of matrix containing simulated time-to-events information
priorObj	an object of class <code>.priorClass</code> generated in <a href="#">set_prior</a>
n.chains	number of parallel chains for the model
n.adapt	number of iterations for adaptation
n.burn	number of iterations discarded as burn-in
n.iter	number of iterations to monitor
seed	the seed of random number generator. Default is the first element of <code>.Random.seed</code>
path	file name for saving the output including folder path

**Value**

a `data.frame` containing summary statistics of the posterior distribution for each simulation

**Examples**

```
# examples in vignette
```

---

run_mcmc_p	<i>Run MCMC for multiple scenarios with provided data with parallel processing</i>
------------	--

---

**Description**

Run MCMC for multiple scenarios with provided data with parallel processing

**Usage**

```
run_mcmc_p(
  dt,
  priorObj,
  n.chains,
  n.adapt,
  n.burn,
  n.iter,
  seed,
  path,
  n.cores = 2
)
```

**Arguments**

dt	a list of matrix containing simulated time-to-events information
priorObj	an object of class <code>.priorClass</code> generated in <a href="#">set_prior</a>
n.chains	number of parallel chains for the model
n.adapt	number of iterations for adaptation
n.burn	number of iterations discarded as burn-in
n.iter	number of iterations to monitor
seed	the seed of random number generator. Default is the first element of <code>.Random.seed</code>
path	file name for saving the output including folder path
n.cores	number of processes to parallelize over (default = 2)

**Value**

a `data.frame` containing summary statistics of the posterior distribution for each simulation

**Examples**

```
# similar to run_mcmc
```

---

set_clin	<i>Specify parameters for enrollment time, drop-out pattern and analysis start time</i>
----------	---

---

**Description**

This function allows user to specify the enrollment and drop-out rate, and the type of clinical cut-off Date. Both enrollment times and drop-out times follow piece-wise exponential distribution.

**Usage**

```
set_clin(gamma, e_itv, CCOD, CCOD_t, etaC, etaE, d_itv)
```

**Arguments**

gamma	A vector of rate of enrollment per unit of time
e_itv	A vector of duration of time periods for recruitment with rates specified in gamma. Note that the length of e_itv should be same length as gamma or 1 less.
CCOD	Type of analysis start time. Analysis starts at CCOD_t months after the first or last patient's enrollment if CCOD = "fixed-first" or CCOD = "fixed-last" respectively. Analysis starts when CCOD_t events have been observed if CCOD = "event"
CCOD_t	Time difference between analysis start and first patient's enrollment if CCOD = "fixed-first". Time difference between analysis start and last patient's enrollment if CCOD = "fixed-last". Number of events observed when analysis starts if CCOD = "event". Patients enrolled after the analysis start time are excluded from the analysis
etaC	A vector for dropout rate per unit time for control arm
etaE	A vector for dropout rate per unit time for experimental arm. If left NULL, it uses the same dropout rate as eta.
d_itv	A vector of duration of time periods for dropping out the study with rates specified in etaC and etaE. Note that the length of d_itv should be same length as etaC or 1 less.

**Value**

A `.clinClass` class containing information on enrollment time, drop-out pattern and analysis start time

**Examples**

```
# set the operational parameter values for the trial
# analysis starts at 64 time units after first patient in
set_clin(gamma = 10, e_itv = 4, etaC = 0.003, CCOD = "fixed-first", CCOD_t = 64)

# analysis starts at 12 time units after last patient in
set_clin(gamma = 2, e_itv = 18, etaC = 0.005, CCOD = "fixed-last", CCOD_t = 12)
```

---

set_cov	<i>Set up covariates</i>
---------	--------------------------

---

### Description

This function saves the mean, variance and covariance among covariates. For technical details, see the vignette.

### Usage

```
set_cov(n_cat, n_cont, mu_int, mu_ext, var, cov, prob_int, prob_ext)
```

### Arguments

n_cat	Number of binary variable. See details
n_cont	Number of continuous variable
mu_int	Mean of covariates in the internal trial. All the covariates are simulated from a multivariate normal distribution. If left NULL, it uses default value 0 for all covariates. If provided one value, this value is used for all covariates
mu_ext	Mean of covariates in the external trial. If left NULL, it uses the same mean as mu_int
var	Variance of covariates. If left NULL, it uses default value 0 for all covariates. If provided one value, it uses this value for all covariates
cov	Covariance between each pair of covariates. Covariance needs to be provided in a certain order and users are encouraged to read the example provided in the vignette. If left NULL, it uses default value 0 for all covariates. If provided one value, it uses this value for every pair of covariates
prob_int	Probability of binary covariate equalling 1 in the internal trial. If left NULL, it uses default value 0.5 for all covariates. If provided one value, it uses this value for all covariates
prob_ext	Probability of binary covariate equalling 1 in the external trial. If left NULL, it uses the same probability as prob_int

### Details

Categorical variables are created by sampling a continuous variable from the multivariate normal distribution (thus respecting the correlation to other covariates specified by cov) and then applying a cut point derived from the prob\_int or prob\_ext quantile of said distribution i.e. for a univariate variable it would be derived as:

```
binvar <- as.numeric(rnorm(n, mu, sqrt(var)) < qnorm(prob, mu, sqrt(var)))
```

Please note that this means that the value of mu\_int & mu\_ext has no impact on categorical covariates and thus can be set to any value.

As an example of how this process works assume  $n_{\text{cat}}=3$  and  $n_{\text{cont}}=2$ . First 5 variables are sampled from the multivariate normal distribution as specified by  $\mu_{\text{int}}/\mu_{\text{ext}}$ ,  $\text{var}$  &  $\text{cov}$ . Then, the first 3 of these variables are converted to binary based on the probabilities specified by  $\text{prob}_{\text{int}}$  and  $\text{prob}_{\text{ext}}$ . This means that that the 2 continuous variables will take their mean and sd from the last 2 entries in the vectors  $\mu_{\text{int}}/\mu_{\text{ext}}$  and  $\text{var}$ .

## Value

A `.covClass` class containing covariate information

---

set_event	<i>Set up time-to-events</i>
-----------	------------------------------

---

## Description

Defines the model formula and distribution to be used when simulating time-to-events. Please see the user-guide for the model formulations

## Usage

```
set_event(event, lambdaC, beta, shape, t_itv, change, keep)
```

## Arguments

event	Distribution of time-to-events: event = "pwexp" for piece-wise exponential distribution. event = "weibull" for Weibull distribution
lambdaC	Baseline hazard rate of internal control arm. Specify a vector for piece-wise hazard with duration specified in <code>t_itv</code> if event = "pwexp"
beta	covariates' coefficients (i.e. log hazard ratios). Must be equal in length to the number of covariates created by <code>simu_cov()</code> (or less if restricted by <code>keep</code> ) plus the number of covariates defined by <code>change</code> .
shape	the shape parameter of Weibull distribution if event = "weibull". NULL if event = "pwexp"
t_itv	a vector indicating interval lengths where the exponential rates provided in <code>lambdaC</code> apply. Note that the length of <code>t_itv</code> is at least 1 less than that of <code>lambdaC</code> and that the final value rate in <code>lambdaC</code> applies after time <code>sum(t_itv)</code> . NULL if event = "weibull"
change	A list of additional derived covariates to be used in simulating time-to-events. See details
keep	A character vector specifying which of the original covariates (i.e. those not derived via the <code>change</code> argument) should be included into the model to simulate time-to-events. If left unspecified all covariates will be included.



## Details

The `change` argument is used to specify additional derived covariates to be used when simulating time-to-events. For example, let's say we have 3 covariates `cov1`, `cov2` & `cov3` but that we also wish to include a new covariate that is an interaction between `cov1` and `cov2` as well as another covariate that is equal to the sum of `cov2` and `cov3`; we could implement this as follows:

```
set_event(
  event = "weibull",
  shape = 0.9,
  lambdaC = 0.0135,
  beta = c(5, 3, 1, 7, 9),
  change = list(
    c("cov1", "*", "cov2"),
    c("cov2", "+", "cov3")
  )
)
```

Note that in the above example 5 values have been specified to `beta`, 3 for the original three covariates and 2 for the two additional derived covariates included via `change`.

Variables derived via `change` are automatically included in the model regardless of whether they are listed in `keep` or not. Likewise, these covariates are derived separately and not via a standard R formula, that is to say including an interaction term does not automatically include the individual fixed effects.

## Value

a `.eventClass` class containing time-to-events information  
 a matrix containing simulated time-to-events information

## Examples

```
# time-to-event follows a Weibull distribution
set_event(event = "weibull", shape = 0.9, lambdaC = 0.0135)

# time-to-event follows a piece-wise exponential distribution
set_event(event = "pwexp", t_itv = 1, lambdaC = c(0.1, 0.02))
```

---

 set\_n

---

*Simulate external trial indicator and treatment arm indicator*


---

## Description

This function conducts validity check and generates a matrix with two binary variables indicating

1. if the observation belongs to the external trial
2. if the observation belongs to the treatment arm.

**Usage**

```
set_n(ssC, ssE, ssExt)
```

**Arguments**

ssC	Number of observations in the internal control arm. Default is 100
ssE	Number of observations in the internal experiment arm. Default is the same number of observations as ssC
ssExt	Number of observations in the external control arm. Default is the same number of observations as ssC

**Value**

A matrix containing external trial indicator and treatment indicator

---

set_prior	<i>Specify prior distributions and predictors for MCMC methods</i>
-----------	--

---

**Description**

Specify prior distributions and predictors for MCMC methods

**Usage**

```
set_prior(pred, prior, r0, alpha, sigma)
```

**Arguments**

pred	Predictors to include in the weibull distribution. No covariates except for treatment indicator is included if pred = NULL. Only propensity score generated using a logistic regression model on all covariates and treatment indicator are included if pred = ps. All covariates and treatment indicator are included if pred = all
prior	Prior distribution for the precision parameter that controls the degree of borrowing. Half-cauchy distribution if prior = "cauchy". No external data is included in the data if prior = "no_ext". External control arm is assumed to have the same baseline hazards as internal control arm if prior = "full_ext". Other options include "gamma" and "unif"
r0	Initial values for the shape of the weibull distribution for time-to-events
alpha	Initial values for log of baseline hazard rate for external and internal control arms. Length of alpha should be 1 if prior = "full_ext" or prior = "no_ext", and equal to 2 otherwise
sigma	Initial values for precision parameter if prior = "cauchy". If left NULL, default value 0.03 is used

**Value**

a `.priorClass` class containing survival data and prior information

**Examples**

```
# hierachical Bayesian model with precision parameter follows a half-cauchy distribution
set_prior(pred = "none", prior = "cauchy", r0 = 1, alpha = c(0, 0), sigma = 0.03)

# hierachical Bayesian model with precision parameter follows a gamma distribution
set_prior(pred = "none", prior = "gamma", r0 = 1, alpha = c(0, 0))

# conventional Bayesian model to not borrow from external control arm
set_prior(pred = "none", prior = "no_ext", alpha = 0)

# conventional Bayesian model to fully borrow from external control arm
set_prior(pred = "none", prior = "full_ext", alpha = 0)
```

---

simu\_cov

*Simulate covariates*


---

**Description**

This function generates continuous and binary covariates through simulating from a multivariate normal distribution. Outcomes are further converted to binary variables using quantiles of the normal distribution calculated from the probability provided. Then the covariates are added to the external trial and treatment arm indicators.

**Usage**

```
simu_cov(ssObj, covObj, driftHR, HR, nsim, seed, path)
```

**Arguments**

ssObj	an object of class <code>.covClass</code> generated in <a href="#">set_n</a>
covObj	an object of class <code>.covClass</code> generated in <a href="#">set_cov</a>
driftHR	hazard ratio of external control and internal control arms
HR	a list of hazard ratio of treatment and control arms
nsim	number of simulation. Default is 5
seed	the seed of R's random number generator. Default is the first element of <code>.Random.seed</code>
path	file name for saving the output including folder path

**Value**

a list of `matrix` containing simulated covariates information

**Examples**

```
# simulate patient-level data with 1 continuous covariate
sample = set_n(ssC = 10, ssE = 20, ssExt = 40)
cov1 = set_cov(n_cat = 0, n_cont = 1, mu_int = 0, mu_ext = 0, var = 1)
simu_cov(ssObj = sample, covObj = cov1, HR = 0.5, driftHR = 1, nsim = 2)

# simulate patient-level data with 1 binary and 2 continuous covariate
cov2 = set_cov(n_cat = 1, n_cont = 2, mu_int = 0, mu_ext = 0, var = 1,
              cov = 0.3, prob_int = 0.2, prob_ext = 0.3)
simu_cov(ssObj = sample, covObj = cov2, HR = 0.5, driftHR = 1, nsim = 2)
```

---

 simu\_time

---

*Simulate time-to-events for multiple scenarios*


---

**Description**

Simulate time-to-events for multiple scenarios

**Usage**

```
simu_time(dt, eventObj, clinInt, clinExt, seed, path)
```

**Arguments**

dt	a list of matrix generated in <a href="#">simu_cov</a> containing simulated covariates information
eventObj	an object of class <code>.eventClass</code> generated in <a href="#">set_event</a> including event information
clinInt	an object of class <code>.clinClass</code> generated in <a href="#">set_clin</a> including internal trial information
clinExt	an object of class <code>.clinClass</code> generated in <a href="#">set_clin</a> including external trial information
seed	the seed of R's random number generator. Default is the first element of <code>.Random.seed</code>
path	file name for saving the output including folder path

**Value**

a list of matrix containing simulated time-to-events information

**Examples**

```
# simulate patient-level data without covariates
# simulate survival time following weibull distribution

# simulate trial indicator and set hazard ratios
sample = set_n(ssC = 10, ssE = 20, ssExt = 40)
sample_hr <- simu_cov(ssObj = sample, HR = 1, driftHR=c(1,1.2), nsim = 10)

# enrollment pattern, drop-out, analysis start time
c_int = set_clin(gamma = 2, e_itv = 10, etaC = 0.5, CCOD = "fixed-first", CCOD_t = 64)
c_ext = c_int

# simulate time-to-event with a weibull distribution
evt1 <- set_event(event = "weibull", shape = 0.8, lambdaC = 0.01)
simu_time(dt = sample_hr, eventObj = evt1, clinInt = c_int, clinExt = c_ext)

# simulate time-to-event with an exponential distribution
evt2 <- set_event(event = "pwexp", t_itv = 1, lambdaC = c(0.1, 0.02))
simu_time(dt = sample_hr, eventObj = evt2, clinInt = c_int, clinExt = c_int)
```

# Index

- \* **classes**
  - .covClass-class, 2
- \* **class**
  - .clinClass-class, 2
  - .eventClass-class, 3
  - .priorClass-class, 3
- \* **constructor**
  - match\_cov, 7
  - set\_clin, 13
  - set\_cov, 15
  - set\_event, 16
  - set\_n, 17
  - set\_prior, 18
- \* **helper**
  - c, .covClass-method, 4
  - c, .priorClass-method, 5
- \* **method**
  - c, .covClass-method, 4
  - c, .priorClass-method, 5
  - get\_summary, 6
  - plot\_bias, 8
  - plot\_hr, 8
  - plot\_mse, 9
  - plot\_power, 10
  - plot\_type1error, 10
  - rej\_est, 11
- \* **simulator**
  - run\_mcmc, 12
  - run\_mcmc\_p, 12
  - simu\_cov, 19
  - simu\_time, 20
- .clinClass (.clinClass-class), 2
- .clinClass-class, 2
- .covClass (.covClass-class), 2
- .covClass-class, 2
- .eventClass (.eventClass-class), 3
- .eventClass-class, 3
- .priorClass (.priorClass-class), 3
- .priorClass-class, 3
- add\_mcmc(), 6
- apply\_mcmc, 3
- apply\_mcmc(), 3
- c, .covClass-method, 4
- c, .priorClass-method, 5
- extract\_samples (apply\_mcmc), 3
- fix\_col\_names, 6
- get\_summary, 6
- is\_psborrow\_dev, 7
- match\_cov, 7
- message(), 11
- plot\_bias, 8
- plot\_hr, 8
- plot\_mse, 9
- plot\_power, 10
- plot\_type1error, 10
- ps\_message, 11
- rej\_est, 11
- run\_mcmc, 12
- run\_mcmc(), 3
- run\_mcmc\_p, 12
- set\_clin, 13, 20
- set\_cov, 5, 15, 19
- set\_event, 16, 20
- set\_n, 17, 19
- set\_prior, 5, 12, 13, 18
- simu\_cov, 19, 20
- simu\_cov(), 16
- simu\_cov, matrix-method (simu\_cov), 19
- simu\_time, 20
- summary.apply\_mcmc (apply\_mcmc), 3