Package: historicalborrowlong (via r-universe)

October 13, 2026

Title Longitudinal Bayesian Historical Borrowing Models

Description Historical borrowing in clinical trials can improve precision and operating characteristics. This package supports a longitudinal hierarchical model to borrow historical control data from other studies to better characterize the control response of the current study. It also quantifies the amount of borrowing through longitudinal benchmark models (independent and pooled). The hierarchical model approach to historical borrowing is discussed by Viele et al. (2013) <doi:10.1002/pst.1589>.

Version 0.1.1

License MIT + file LICENSE

URL https://wlandau.github.io/historicalborrowlong/, https://github.com/wlandau/historicalborrowlong

BugReports https://github.com/wlandau/historicalborrowlong/issues

Depends R (>= 4.0.0)

Imports clustermq, dplyr, ggplot2, MASS, Matrix, methods, posterior, Rcpp, RcppParallel, rlang, rstan (>= 2.26.0), rstantools, stats, tibble, tidyr, tidyselect, trialr, utils, withr, zoo

Suggests knitr, markdown, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

Language en-US

VignetteBuilder knitr

Config/testthat/edition 3

Biarch true

LinkingTo BH, Rcpp, RcppEigen, RcppParallel, rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

SystemRequirements GNU make

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author William Michael Landau [aut, cre] (ORCID: <https://orcid.org/0000-0003-1878-3253>), Albert Man [rev], Eli Lilly and Company [cph]

Maintainer William Michael Landau <will.landau.oss@gmail.com>

Config/pak/sysreqs make

Repository https://cran.r-universe.dev

Date/Publication 2026-09-13 16:28:16 UTC

RemoteUrl https://github.com/cran/historicalborrowlong

RemoteRef HEAD

RemoteSha 9085dfc56277a29924b8570b3a455f8aad14eb7b

Contents

historicalborrowlong-package . . . 2
hbl_convergence . . . 3
hbl_data . . . 4
hbl_ess . . . 6
hbl_mcmc_hierarchical . . . 9
hbl_mcmc_independent . . . 13
hbl_mcmc_pool . . . 17
hbl_mcmc_sge . . . 21
hbl_plot_borrow . . . 25
hbl_plot_group . . . 27
hbl_plot_tau . . . 29
hbl_sim_hierarchical . . . 30
hbl_sim_independent . . . 33
hbl_sim_pool . . . 35
hbl_summary . . . 37

Index **42**

`historicalborrowlong-package`
historicalborrowlong: Bayesian longitudinal historical borrowing models for clinical studies.

Description

Bayesian longitudinal historical borrowing models for clinical studies.

hbl_convergence *Check convergence diagnostics*

Description

Check the convergence diagnostics on a model.

Usage

```
hbl_convergence(mcmc)
```

Arguments

mcmc	A wide data frame of posterior samples returned by hbl_mcmc_hierarchical() or similar MCMC function.

Value

A data frame of summarized convergence diagnostics. max_rhat is the maximum univariate Gelman/Rubin potential scale reduction factor over all the parameters of the model, min_ess_bulk is the minimum bulk effective sample size over the parameters, and min_ess_tail is the minimum tail effective sample size. max_rhat should be below 1.01, and the ESS metrics should both be above 100 times the number of MCMC chains. If any of these conditions are not true, the MCMC did not converge, and it is recommended to try running the model for more saved iterations (and if max_rhat is high, possibly more warmup iterations).

See Also

Other mcmc: hbl_mcmc_hierarchical(), hbl_mcmc_independent(), hbl_mcmc_pool(), hbl_mcmc_sge()

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_pool(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_pool(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
```

```
  )
)
hbl_convergence(mcmc)
}
```

`hbl_data` *Standardize data*

Description

Standardize a tidy input dataset.

Usage

```
hbl_data(
  data,
  response,
  study,
  study_reference,
  group,
  group_reference,
  patient,
  rep,
  rep_reference,
  covariates
)
```

Arguments

`data`	A tidy data frame or `tibble` with the data.
`response`	Character of length 1, name of the column in `data` with the response/outcome variable. `data[[response]]` must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.
`study`	Character of length 1, name of the column in `data` with the study ID.
`study_reference`	Atomic of length 1, element of the `study` column that indicates the current study. (The other studies are historical studies.)
`group`	Character of length 1, name of the column in `data` with the group ID.
`group_reference`	Atomic of length 1, element of the `group` column that indicates the control group. (The other groups may be treatment groups.)
`patient`	Character of length 1, name of the column in `data` with the patient ID.
`rep`	Character of length 1, name of the column in `data` with the rep ID.

`rep_reference` Atomic of length 1, element of the `rep` column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)

`covariates` Character vector of column names in `data` with the columns with baseline covariates. These can be continuous, categorical, or binary. Regardless, `historicalborrowlong` derives the appropriate model matrix.

Each baseline covariate column must truly be a *baseline* covariate: elements must be equal for all time points within each patient (after the steps in the "Data processing" section). In other words, covariates must not be time-varying.

A large number of covariates, or a large number of levels in a categorical covariate, can severely slow down the computation. Please consider carefully if you really need to include such complicated baseline covariates.

Details

Users do not normally need to call this function. It mainly serves exposes the indexing behavior of studies and group levels to aid in interpreting summary tables.

Value

A standardized tidy data frame with one row per patient and the following columns:

- `response`: continuous response/outcome variable. (Should be change from baseline of an outcome of interest.)
- `study_label`: human-readable label of the study.
- `study`: integer study index with the max index equal to the current study (at `study_reference`).
- `group_label`: human-readable group label (e.g. treatment arm name).
- `group`: integer group index with an index of 1 equal to the control group (at `group_reference`).
- `patient_label`: original patient ID.
- `patient`: integer patient index.
- `rep_label`: original rep ID (e.g. time point or patient visit).
- `rep`: integer rep index.
- `covariate_*`: baseline covariate columns.

Data processing

Before running the MCMC, dataset is pre-processed. This includes expanding the rows of the data so every rep of every patient gets an explicit row. So if your original data has irregular rep IDs, e.g. unscheduled visits in a clinical trial that few patients attend, please remove them before the analysis. Only the most common rep IDs should be added.

After expanding the rows, the function fills in missing values for every column except the response. That includes covariates. Missing covariate values are filled in, first with last observation carried forward, then with last observation carried backward. If there are still missing values after this process, the program throws an informative error.

Examples

```
set.seed(0)
data <- hbl_sim_independent(n_continuous = 1, n_study = 2)$data
data <- dplyr::select(
  data,
  study,
  group,
  rep,
  patient,
  response,
  tidyselect::everything()
)
data <- dplyr::rename(
  data,
  change = response,
  trial = study,
  arm = group,
  subject = patient,
  visit = rep,
  cov1 = covariate_study1_continuous1,
  cov2 = covariate_study2_continuous1
)
data$trial <- paste0("trial", data$trial)
data$arm <- paste0("arm", data$arm)
data$subject <- paste0("subject", data$subject)
data$visit <- paste0("visit", data$visit)
hbl_data(
  data = data,
  response = "change",
  study = "trial",
  study_reference = "trial1",
  group = "arm",
  group_reference = "arm1",
  patient = "subject",
  rep = "visit",
  rep_reference = "visit1",
  covariates = c("cov1", "cov2")
)
```

`hbl_ess` *Effective sample size (ESS)*

Description

Quantify borrowing with effective sample size (ESS) as cited and explained in the methods vignette at https://wlandau.github.io/historicalborrowlong/articles/methods.html.

Usage

```
hbl_ess(
  mcmc_pool,
  mcmc_hierarchical,
  data,
  response = "response",
  study = "study",
  study_reference = max(data[[study]]),
  group = "group",
  group_reference = min(data[[group]]),
  patient = "patient",
  rep = "rep",
  rep_reference = min(data[[rep]])
)
```

Arguments

`mcmc_pool`	A fitted model from `hbl_mcmc_pool()`.
`mcmc_hierarchical`	A fitted model from `hbl_mcmc_hierarchical()`.
`data`	A tidy data frame or `tibble` with the data.
`response`	Character of length 1, name of the column in `data` with the response/outcome variable. `data[[response]]` must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.
`study`	Character of length 1, name of the column in `data` with the study ID.
`study_reference`	Atomic of length 1, element of the `study` column that indicates the current study. (The other studies are historical studies.)
`group`	Character of length 1, name of the column in `data` with the group ID.
`group_reference`	Atomic of length 1, element of the `group` column that indicates the control group. (The other groups may be treatment groups.)
`patient`	Character of length 1, name of the column in `data` with the patient ID.
`rep`	Character of length 1, name of the column in `data` with the rep ID.
`rep_reference`	Atomic of length 1, element of the `rep` column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)

Value

A data frame with one row per discrete time point ("rep") and the following columns:

- `v0`: posterior predictive variance of the control group mean of a hypothetical new study given the pooled model. Calculated as the mean over MCMC samples of `1 / sum(sigma_i ^ 2)`, where each `sigma_i` is the residual standard deviation of study `i` estimated from the pooled model.
- `v_tau`: posterior predictive variance of a hypothetical new control group mean under the hierarchical model. Calculated by averaging over predictive draws, where each predictive draw is from `rnorm(n = 1, mean = mu_, sd = tau_)` and `mu_` and `tau_` are the `mu` and `tau` components of an MCMC sample.
- `n`: number of non-missing historical control patients.
- `weight`: strength of borrowing as a ratio of variances: `v0 / v_tau`.
- `ess`: strength of borrowing as a prior effective sample size: `n v0 / v_tau`, where `n` is the number of non-missing historical control patients.

See Also

Other summary: `hbl_summary()`

Examples

```
set.seed(0)
data <- hbl_sim_independent(n_continuous = 2)$data
data$group <- sprintf("group%s", data$group)
data$study <- sprintf("study%s", data$study)
data$rep <- sprintf("rep%s", data$rep)
tmp <- utils::capture.output(
  suppressWarnings(
    pool <- hbl_mcmc_pool(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
tmp <- utils::capture.output(
  suppressWarnings(
    hierarchical <- hbl_mcmc_hierarchical(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
hbl_ess(
  mcmc_pool = pool,
  mcmc_hierarchical = hierarchical,
  data = data
)
```

`hbl_mcmc_hierarchical` *Longitudinal hierarchical MCMC*

Description

Run the longitudinal hierarchical model with MCMC.

Usage

```
hbl_mcmc_hierarchical(
  data,
  response = "response",
  study = "study",
  study_reference = max(data[[study]]),
  group = "group",
  group_reference = min(data[[group]]),
  patient = "patient",
  rep = "rep",
  rep_reference = min(data[[rep]]),
  covariates = grep("^covariate", colnames(data), value = TRUE),
  constraint = FALSE,
  s_delta = 30,
  s_beta = 30,
  s_sigma = 30,
  s_lambda = 1,
  s_mu = 30,
  s_tau = 30,
  d_tau = 4,
  prior_tau = "half_t",
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  control = list(max_treedepth = 17, adapt_delta = 0.99),
  ...
)
```

Arguments

`data`	Tidy data frame with one row per patient per rep, indicator columns for the response variable, study, group, patient, rep, and covariates. All columns must be atomic vectors (e.g. not lists).
`response`	Character of length 1, name of the column in `data` with the response/outcome variable. `data[[response]]` must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.

`study` Character of length 1, name of the column in `data` with the study ID.

`study_reference`
Atomic of length 1, element of the `study` column that indicates the current study. (The other studies are historical studies.)

`group` Character of length 1, name of the column in `data` with the group ID.

`group_reference`
Atomic of length 1, element of the `group` column that indicates the control group. (The other groups may be treatment groups.)

`patient` Character of length 1, name of the column in `data` with the patient ID.

`rep` Character of length 1, name of the column in `data` with the rep ID.

`rep_reference` Atomic of length 1, element of the `rep` column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)

`covariates` Character vector of column names in `data` with the columns with baseline covariates. These can be continuous, categorical, or binary. Regardless, `historicalborrowlong` derives the appropriate model matrix.

Each baseline covariate column must truly be a *baseline* covariate: elements must be equal for all time points within each patient (after the steps in the "Data processing" section). In other words, covariates must not be time-varying.

A large number of covariates, or a large number of levels in a categorical covariate, can severely slow down the computation. Please consider carefully if you really need to include such complicated baseline covariates.

`constraint` Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

`s_delta` Numeric of length 1, prior standard deviation of the study-by-group effect parameters `delta`.

`s_beta` Numeric of length 1, prior standard deviation of the fixed effects `beta`.

`s_sigma` Numeric of length 1, prior upper bound of the residual standard deviations.

`s_lambda` shape parameter of the LKJ priors on the unstructured correlation matrices.

`s_mu` Numeric of length 1, prior standard deviation of `mu`.

`s_tau` Non-negative numeric of length 1. If `prior_tau` is `"half_t"`, then `s_tau` is the scale parameter of the Student t prior of `tau` and analogous to the `sigma` parameter of the Student-t parameterization given at https://mc-stan.org/docs/functions-reference/unbounded_continuous_distributions.html. # nolint If `prior_tau` is `"uniform"`, then `s_tau` is the upper bound of `tau`. Upper bound on `tau` if `prior_tau` is `"uniform"`.

`d_tau` Positive numeric of length 1. Degrees of freedom of the Student t prior of `tau` if `prior_tau` is `"half_t"`.

`prior_tau` Character string, family of the prior of `tau`. If `prior_tau` equals `"uniform"`, then the prior on `tau` is a uniform prior with lower bound 0 and upper bound `s_tau`. If `prior_tau` equals `"half_t"`, then the prior on `tau` is a half Student-t prior with center 0, lower bound 0, scale parameter `s_tau`, and degrees of

freedom d_tau. The scale parameter s_tau is analogous to the sigma parameter of the Student-t parameterization given at https://mc-stan.org/docs/functions-reference/unbounded_continuous_distributions.html. # nolint

covariance_current

Character of length 1, covariance structure of the current study. Possible values are "unstructured" for fully parameterized covariance matrices, "ar1" for AR(1) covariance matrices, and "diagonal" for residuals independent across time within each patient. In MCMC (e.g. hbl_mcmc_hierarchical()), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for covariance_historical if there are many historical studies in the data.

covariance_historical

Same as covariance_current, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.

control

A named list of parameters to control the sampler's behavior. It defaults to NULL so all the default values are used. First, the following are adaptation parameters for sampling algorithms. These are parameters used in Stan with similar names here.

- adapt_engaged (logical)
- adapt_gamma (double, positive, defaults to 0.05)
- adapt_delta (double, between 0 and 1, defaults to 0.8)
- adapt_kappa (double, positive, defaults to 0.75)
- adapt_t0 (double, positive, defaults to 10)
- adapt_init_buffer (integer, positive, defaults to 75)
- adapt_term_buffer (integer, positive, defaults to 50)
- adapt_window (integer, positive, defaults to 25)

In addition, algorithm HMC (called 'static HMC' in Stan) and NUTS share the following parameters:

- stepsize (double, positive, defaults to 1) Note: this controls the *initial* stepsize only, unless adapt_engaged=FALSE.
- stepsize_jitter (double, [0,1], defaults to 0)
- metric (string, one of "unit_e", "diag_e", "dense_e", defaults to "diag_e")

For algorithm NUTS, we can also set:

- max_treedepth (integer, positive, defaults to 10)

For algorithm HMC, we can also set:

- int_time (double, positive)

For test_grad mode, the following parameters can be set:

- epsilon (double, defaults to 1e-6)
- error (double, defaults to 1e-6)

...

Other optional parameters:

- chain_id (integer)
- init_r (double, positive)

- test_grad (logical)
- append_samples (logical)
- refresh(integer)
- save_warmup(logical)
- deprecated: enable_random_init(logical)

chain_id can be a vector to specify the chain_id for all chains or an integer. For the former case, they should be unique. For the latter, the sequence of integers starting from the given chain_id are used for all chains.

init_r is used only for generating random initial values, specifically when init="random" or not all parameters are initialized in the user-supplied list or function. If specified, the initial values are simulated uniformly from interval [-init_r, init_r] rather than using the default interval (see the manual of (cmd)Stan).

test_grad (logical). If test_grad=TRUE, Stan will not do any sampling. Instead, the gradient calculation is tested and printed out and the fitted stanfit object is in test gradient mode. By default, it is FALSE.

append_samples (logical). Only relevant if sample_file is specified *and* is an existing file. In that case, setting append_samples=TRUE will append the samples to the existing file rather than overwriting the contents of the file.

refresh (integer) can be used to control how often the progress of the sampling is reported (i.e. show the progress every refresh iterations). By default, refresh = max(iter/10, 1). The progress indicator is turned off if refresh <= 0.

Deprecated: enable_random_init (logical) being TRUE enables specifying initial values randomly when the initial values are not fully specified from the user.

save_warmup (logical) indicates whether to save draws during the warmup phase and defaults to TRUE. Some memory related problems can be avoided by setting it to FALSE, but some diagnostics are more limited if the warmup draws are not stored.

Value

A tidy data frame of parameter samples from the posterior distribution. Columns .chain, .iteration, and .draw have the meanings documented in the posterior package.

Data processing

Before running the MCMC, dataset is pre-processed. This includes expanding the rows of the data so every rep of every patient gets an explicit row. So if your original data has irregular rep IDs, e.g. unscheduled visits in a clinical trial that few patients attend, please remove them before the analysis. Only the most common rep IDs should be added.

After expanding the rows, the function fills in missing values for every column except the response. That includes covariates. Missing covariate values are filled in, first with last observation carried forward, then with last observation carried backward. If there are still missing values after this process, the program throws an informative error.

See Also

Other mcmc: hbl_convergence(), hbl_mcmc_independent(), hbl_mcmc_pool(), hbl_mcmc_sge()

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_hierarchical(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_hierarchical(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
mcmc
}
```

---

hbl_mcmc_independent *Longitudinal independent MCMC*

Description

Run the longitudinal independent model with MCMC.

Usage

```
hbl_mcmc_independent(
  data,
  response = "response",
  study = "study",
  study_reference = max(data[[study]]),
  group = "group",
  group_reference = min(data[[group]]),
  patient = "patient",
  rep = "rep",
  rep_reference = min(data[[rep]]),
  covariates = grep("^covariate", colnames(data), value = TRUE),
  constraint = FALSE,
```

```
  s_alpha = 30,
  s_delta = 30,
  s_beta = 30,
  s_sigma = 30,
  s_lambda = 1,
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  control = list(max_treedepth = 17, adapt_delta = 0.99),
  ...
)
```

Arguments

`data`	Tidy data frame with one row per patient per rep, indicator columns for the response variable, study, group, patient, rep, and covariates. All columns must be atomic vectors (e.g. not lists).
`response`	Character of length 1, name of the column in `data` with the response/outcome variable. `data[[response]]` must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.
`study`	Character of length 1, name of the column in `data` with the study ID.
`study_reference`	Atomic of length 1, element of the `study` column that indicates the current study. (The other studies are historical studies.)
`group`	Character of length 1, name of the column in `data` with the group ID.
`group_reference`	Atomic of length 1, element of the `group` column that indicates the control group. (The other groups may be treatment groups.)
`patient`	Character of length 1, name of the column in `data` with the patient ID.
`rep`	Character of length 1, name of the column in `data` with the rep ID.
`rep_reference`	Atomic of length 1, element of the `rep` column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)
`covariates`	Character vector of column names in `data` with the columns with baseline covariates. These can be continuous, categorical, or binary. Regardless, `historicalborrowlong` derives the appropriate model matrix. Each baseline covariate column must truly be a *baseline* covariate: elements must be equal for all time points within each patient (after the steps in the "Data processing" section). In other words, covariates must not be time-varying. A large number of covariates, or a large number of levels in a categorical covariate, can severely slow down the computation. Please consider carefully if you really need to include such complicated baseline covariates.
`constraint`	Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

s_alpha Numeric of length 1, prior standard deviation of the study-specific control group mean parameters alpha.

s_delta Numeric of length 1, prior standard deviation of the study-by-group effect parameters delta.

s_beta Numeric of length 1, prior standard deviation of the fixed effects beta.

s_sigma Numeric of length 1, prior upper bound of the residual standard deviations.

s_lambda shape parameter of the LKJ priors on the unstructured correlation matrices.

covariance_current
Character of length 1, covariance structure of the current study. Possible values are "unstructured" for fully parameterized covariance matrices, "ar1" for AR(1) covariance matrices, and "diagonal" for residuals independent across time within each patient. In MCMC (e.g. hbl_mcmc_hierarchical()), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for covariance_historical if there are many historical studies in the data.

covariance_historical
Same as covariance_current, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.

control A named list of parameters to control the sampler's behavior. It defaults to NULL so all the default values are used. First, the following are adaptation parameters for sampling algorithms. These are parameters used in Stan with similar names here.

- adapt_engaged (logical)
- adapt_gamma (double, positive, defaults to 0.05)
- adapt_delta (double, between 0 and 1, defaults to 0.8)
- adapt_kappa (double, positive, defaults to 0.75)
- adapt_t0 (double, positive, defaults to 10)
- adapt_init_buffer (integer, positive, defaults to 75)
- adapt_term_buffer (integer, positive, defaults to 50)
- adapt_window (integer, positive, defaults to 25)

In addition, algorithm HMC (called 'static HMC' in Stan) and NUTS share the following parameters:

- stepsize (double, positive, defaults to 1) Note: this controls the *initial* stepsize only, unless adapt_engaged=FALSE.
- stepsize_jitter (double, [0,1], defaults to 0)
- metric (string, one of "unit_e", "diag_e", "dense_e", defaults to "diag_e")

For algorithm NUTS, we can also set:

- max_treedepth (integer, positive, defaults to 10)

For algorithm HMC, we can also set:

- int_time (double, positive)

For test_grad mode, the following parameters can be set:

- epsilon (double, defaults to 1e-6)

- error (double, defaults to 1e-6)

... Other optional parameters:

- chain_id (integer)
- init_r (double, positive)
- test_grad (logical)
- append_samples (logical)
- refresh(integer)
- save_warmup(logical)
- deprecated: enable_random_init(logical)

chain_id can be a vector to specify the chain_id for all chains or an integer. For the former case, they should be unique. For the latter, the sequence of integers starting from the given chain_id are used for all chains.

init_r is used only for generating random initial values, specifically when init="random" or not all parameters are initialized in the user-supplied list or function. If specified, the initial values are simulated uniformly from interval [-init_r, init_r] rather than using the default interval (see the manual of (cmd)Stan).

test_grad (logical). If test_grad=TRUE, Stan will not do any sampling. Instead, the gradient calculation is tested and printed out and the fitted stanfit object is in test gradient mode. By default, it is FALSE.

append_samples (logical). Only relevant if sample_file is specified *and* is an existing file. In that case, setting append_samples=TRUE will append the samples to the existing file rather than overwriting the contents of the file.

refresh (integer) can be used to control how often the progress of the sampling is reported (i.e. show the progress every refresh iterations). By default, refresh = max(iter/10, 1). The progress indicator is turned off if refresh <= 0.

Deprecated: enable_random_init (logical) being TRUE enables specifying initial values randomly when the initial values are not fully specified from the user.

save_warmup (logical) indicates whether to save draws during the warmup phase and defaults to TRUE. Some memory related problems can be avoided by setting it to FALSE, but some diagnostics are more limited if the warmup draws are not stored.

Value

A tidy data frame of parameter samples from the posterior distribution. Columns .chain, .iteration, and .draw have the meanings documented in the posterior package.

Data processing

Before running the MCMC, dataset is pre-processed. This includes expanding the rows of the data so every rep of every patient gets an explicit row. So if your original data has irregular rep IDs, e.g. unscheduled visits in a clinical trial that few patients attend, please remove them before the analysis. Only the most common rep IDs should be added.

After expanding the rows, the function fills in missing values for every column except the response. That includes covariates. Missing covariate values are filled in, first with last observation carried forward, then with last observation carried backward. If there are still missing values after this process, the program throws an informative error.

See Also

Other mcmc: hbl_convergence(), hbl_mcmc_hierarchical(), hbl_mcmc_pool(), hbl_mcmc_sge()

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_independent(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_independent(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
mcmc
}
```

---

hbl_mcmc_pool *Longitudinal pooled MCMC*

Description

Run the longitudinal pooled model with MCMC.

Usage

```
hbl_mcmc_pool(
  data,
  response = "response",
  study = "study",
  study_reference = max(data[[study]]),
  group = "group",
  group_reference = min(data[[group]]),
```

```
  patient = "patient",
  rep = "rep",
  rep_reference = min(data[[rep]]),
  covariates = grep("^covariate", colnames(data), value = TRUE),
  constraint = FALSE,
  s_alpha = 30,
  s_delta = 30,
  s_beta = 30,
  s_sigma = 30,
  s_lambda = 1,
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  control = list(max_treedepth = 17, adapt_delta = 0.99),
  ...
)
```

Arguments

`data`	Tidy data frame with one row per patient per rep, indicator columns for the response variable, study, group, patient, rep, and covariates. All columns must be atomic vectors (e.g. not lists).
`response`	Character of length 1, name of the column in `data` with the response/outcome variable. `data[[response]]` must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.
`study`	Character of length 1, name of the column in `data` with the study ID.
`study_reference`	Atomic of length 1, element of the `study` column that indicates the current study. (The other studies are historical studies.)
`group`	Character of length 1, name of the column in `data` with the group ID.
`group_reference`	Atomic of length 1, element of the `group` column that indicates the control group. (The other groups may be treatment groups.)
`patient`	Character of length 1, name of the column in `data` with the patient ID.
`rep`	Character of length 1, name of the column in `data` with the rep ID.
`rep_reference`	Atomic of length 1, element of the `rep` column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)
`covariates`	Character vector of column names in `data` with the columns with baseline covariates. These can be continuous, categorical, or binary. Regardless, `historicalborrowlong` derives the appropriate model matrix. Each baseline covariate column must truly be a *baseline* covariate: elements must be equal for all time points within each patient (after the steps in the "Data processing" section). In other words, covariates must not be time-varying.

A large number of covariates, or a large number of levels in a categorical covariate, can severely slow down the computation. Please consider carefully if you really need to include such complicated baseline covariates.

`constraint` Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

`s_alpha` Numeric of length 1, prior standard deviation of the study-specific control group mean parameters `alpha`.

`s_delta` Numeric of length 1, prior standard deviation of the study-by-group effect parameters `delta`.

`s_beta` Numeric of length 1, prior standard deviation of the fixed effects `beta`.

`s_sigma` Numeric of length 1, prior upper bound of the residual standard deviations.

`s_lambda` shape parameter of the LKJ priors on the unstructured correlation matrices.

`covariance_current`
Character of length 1, covariance structure of the current study. Possible values are `"unstructured"` for fully parameterized covariance matrices, `"ar1"` for AR(1) covariance matrices, and `"diagonal"` for residuals independent across time within each patient. In MCMC (e.g. `hbl_mcmc_hierarchical()`), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for `covariance_historical` if there are many historical studies in the data.

`covariance_historical`
Same as `covariance_current`, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.

`control` A named `list` of parameters to control the sampler's behavior. It defaults to `NULL` so all the default values are used. First, the following are adaptation parameters for sampling algorithms. These are parameters used in Stan with similar names here.

- `adapt_engaged` (`logical`)
- `adapt_gamma` (`double`, positive, defaults to 0.05)
- `adapt_delta` (`double`, between 0 and 1, defaults to 0.8)
- `adapt_kappa` (`double`, positive, defaults to 0.75)
- `adapt_t0` (`double`, positive, defaults to 10)
- `adapt_init_buffer` (`integer`, positive, defaults to 75)
- `adapt_term_buffer` (`integer`, positive, defaults to 50)
- `adapt_window` (`integer`, positive, defaults to 25)

In addition, algorithm HMC (called 'static HMC' in Stan) and NUTS share the following parameters:

- `stepsize` (`double`, positive, defaults to 1) Note: this controls the *initial* stepsize only, unless `adapt_engaged=FALSE`.
- `stepsize_jitter` (`double`, [0,1], defaults to 0)
- `metric` (`string`, one of "unit_e", "diag_e", "dense_e", defaults to "diag_e")

For algorithm NUTS, we can also set:

- `max_treedepth` (`integer`, positive, defaults to 10)

For algorithm HMC, we can also set:

- `int_time` (`double`, positive)

For `test_grad` mode, the following parameters can be set:

- `epsilon` (`double`, defaults to 1e-6)
- `error` (`double`, defaults to 1e-6)

`...` Additional named arguments of `rstan::sampling()`. See the documentation of `rstan::sampling()` for details.

Value

A tidy data frame of parameter samples from the posterior distribution. Columns `.chain`, `.iteration`, and `.draw` have the meanings documented in the `posterior` package.

Data processing

Before running the MCMC, dataset is pre-processed. This includes expanding the rows of the data so every rep of every patient gets an explicit row. So if your original data has irregular rep IDs, e.g. unscheduled visits in a clinical trial that few patients attend, please remove them before the analysis. Only the most common rep IDs should be added.

After expanding the rows, the function fills in missing values for every column except the response. That includes covariates. Missing covariate values are filled in, first with last observation carried forward, then with last observation carried backward. If there are still missing values after this process, the program throws an informative error.

See Also

Other mcmc: `hbl_convergence()`, `hbl_mcmc_hierarchical()`, `hbl_mcmc_independent()`, `hbl_mcmc_sge()`

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_pool(
  n_study = 3,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_pool(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
```

```
)
mcmc
}
```

`hbl_mcmc_sge` *Run all MCMCs on a Sun Grid Engine (SGE) cluster.*

Description

Run all MCMCs on a Sun Grid Engine (SGE) cluster. Different models run in different jobs, and different chains run on different cores.

Usage

```
hbl_mcmc_sge(
  data,
  response = "response",
  study = "study",
  study_reference = max(data[[study]]),
  group = "group",
  group_reference = min(data[[group]]),
  patient = "patient",
  rep = "rep",
  rep_reference = min(data[[rep]]),
  covariates = grep("^covariate", colnames(data), value = TRUE),
  constraint = FALSE,
  s_alpha = 30,
  s_delta = 30,
  s_beta = 30,
  s_sigma = 30,
  s_lambda = 1,
  s_mu = 30,
  s_tau = 30,
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  control = list(max_treedepth = 17, adapt_delta = 0.99),
  log = "/dev/null",
  scheduler = "sge",
  chains = 1,
  cores = chains,
  ...
)
```

Arguments

`data`	Tidy data frame with one row per patient per rep, indicator columns for the response variable, study, group, patient, rep, and covariates. All columns must be atomic vectors (e.g. not lists).

response — Character of length 1, name of the column in data with the response/outcome variable. data[[response]] must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.

study — Character of length 1, name of the column in data with the study ID.

study_reference — Atomic of length 1, element of the study column that indicates the current study. (The other studies are historical studies.)

group — Character of length 1, name of the column in data with the group ID.

group_reference — Atomic of length 1, element of the group column that indicates the control group. (The other groups may be treatment groups.)

patient — Character of length 1, name of the column in data with the patient ID.

rep — Character of length 1, name of the column in data with the rep ID.

rep_reference — Atomic of length 1, element of the rep column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)

covariates — Character vector of column names in data with the columns with baseline covariates. These can be continuous, categorical, or binary. Regardless, historicalborrowlong derives the appropriate model matrix.

Each baseline covariate column must truly be a *baseline* covariate: elements must be equal for all time points within each patient (after the steps in the "Data processing" section). In other words, covariates must not be time-varying.

A large number of covariates, or a large number of levels in a categorical covariate, can severely slow down the computation. Please consider carefully if you really need to include such complicated baseline covariates.

constraint — Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

s_alpha — Numeric of length 1, prior standard deviation of the study-specific control group mean parameters alpha.

s_delta — Numeric of length 1, prior standard deviation of the study-by-group effect parameters delta.

s_beta — Numeric of length 1, prior standard deviation of the fixed effects beta.

s_sigma — Numeric of length 1, prior upper bound of the residual standard deviations.

s_lambda — shape parameter of the LKJ priors on the unstructured correlation matrices.

s_mu — Numeric of length 1, prior standard deviation of mu.

s_tau — Non-negative numeric of length 1. If prior_tau is "half_t", then s_tau is the scale parameter of the Student t prior of tau and analogous to the sigma parameter of the Student-t parameterization given at https://mc-stan.org/docs/functions-reference/unbounded_continuous_distributions.html. # nolint If prior_tau is "uniform", then s_tau is the upper bound of tau. Upper bound on tau if prior_tau is "uniform".

`covariance_current`
: Character of length 1, covariance structure of the current study. Possible values are `"unstructured"` for fully parameterized covariance matrices, `"ar1"` for AR(1) covariance matrices, and `"diagonal"` for residuals independent across time within each patient. In MCMC (e.g. `hbl_mcmc_hierarchical()`), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for `covariance_historical` if there are many historical studies in the data.

`covariance_historical`
: Same as `covariance_current`, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.

`control`
: A named `list` of parameters to control the sampler's behavior. It defaults to `NULL` so all the default values are used. First, the following are adaptation parameters for sampling algorithms. These are parameters used in Stan with similar names here.

  - `adapt_engaged` (`logical`)
  - `adapt_gamma` (`double`, positive, defaults to 0.05)
  - `adapt_delta` (`double`, between 0 and 1, defaults to 0.8)
  - `adapt_kappa` (`double`, positive, defaults to 0.75)
  - `adapt_t0` (`double`, positive, defaults to 10)
  - `adapt_init_buffer` (`integer`, positive, defaults to 75)
  - `adapt_term_buffer` (`integer`, positive, defaults to 50)
  - `adapt_window` (`integer`, positive, defaults to 25)

  In addition, algorithm HMC (called 'static HMC' in Stan) and NUTS share the following parameters:

  - `stepsize` (`double`, positive, defaults to 1) Note: this controls the *initial* stepsize only, unless `adapt_engaged=FALSE`.
  - `stepsize_jitter` (`double`, [0,1], defaults to 0)
  - `metric` (`string`, one of "unit_e", "diag_e", "dense_e", defaults to "diag_e")

  For algorithm NUTS, we can also set:

  - `max_treedepth` (`integer`, positive, defaults to 10)

  For algorithm HMC, we can also set:

  - `int_time` (`double`, positive)

  For `test_grad` mode, the following parameters can be set:

  - `epsilon` (`double`, defaults to 1e-6)
  - `error` (`double`, defaults to 1e-6)

`log`
: Character of length 1, path to a directory (with a trailing `/`) or a single file path. The SGE log files go here. Only works if `scheduler` is `"sge"`.

`scheduler`
: Either `"sge"` or `"local"`, high-performance computing scheduler / resource manager to use. Choose `"sge"` for serious use cases with a Sun Grid Engine (SGE) cluster. Otherwise, to run models sequentially on the current node, choose `"local"`.

`chains`
: A positive integer specifying the number of Markov chains. The default is 4.

cores — The number of cores to use when executing the Markov chains in parallel. The default is to use the value of the "mc.cores" option if it has been set and otherwise to default to 1 core. However, we recommend setting it to be as many processors as the hardware and RAM allow (up to the number of chains). See detectCores if you don't know this number for your system.

... — Other optional parameters:

- chain_id (integer)
- init_r (double, positive)
- test_grad (logical)
- append_samples (logical)
- refresh(integer)
- save_warmup(logical)
- deprecated: enable_random_init(logical)

chain_id can be a vector to specify the chain_id for all chains or an integer. For the former case, they should be unique. For the latter, the sequence of integers starting from the given chain_id are used for all chains.

init_r is used only for generating random initial values, specifically when init="random" or not all parameters are initialized in the user-supplied list or function. If specified, the initial values are simulated uniformly from interval [-init_r, init_r] rather than using the default interval (see the manual of (cmd)Stan).

test_grad (logical). If test_grad=TRUE, Stan will not do any sampling. Instead, the gradient calculation is tested and printed out and the fitted stanfit object is in test gradient mode. By default, it is FALSE.

append_samples (logical). Only relevant if sample_file is specified *and* is an existing file. In that case, setting append_samples=TRUE will append the samples to the existing file rather than overwriting the contents of the file.

refresh (integer) can be used to control how often the progress of the sampling is reported (i.e. show the progress every refresh iterations). By default, refresh = max(iter/10, 1). The progress indicator is turned off if refresh <= 0.

Deprecated: enable_random_init (logical) being TRUE enables specifying initial values randomly when the initial values are not fully specified from the user.

save_warmup (logical) indicates whether to save draws during the warmup phase and defaults to TRUE. Some memory related problems can be avoided by setting it to FALSE, but some diagnostics are more limited if the warmup draws are not stored.

Value

A list of tidy data frames of parameter samples from the posterior distribution. Columns .chain, .iteration, and .draw have the meanings documented in the posterior package.

Data processing

Before running the MCMC, dataset is pre-processed. This includes expanding the rows of the data so every rep of every patient gets an explicit row. So if your original data has irregular rep IDs, e.g. unscheduled visits in a clinical trial that few patients attend, please remove them before the analysis. Only the most common rep IDs should be added.

After expanding the rows, the function fills in missing values for every column except the response. That includes covariates. Missing covariate values are filled in, first with last observation carried forward, then with last observation carried backward. If there are still missing values after this process, the program throws an informative error.

See Also

Other mcmc: `hbl_convergence()`, `hbl_mcmc_hierarchical()`, `hbl_mcmc_independent()`, `hbl_mcmc_pool()`

Examples

```
if (identical(Sys.getenv("HBL_SGE"), "true")) {
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_hierarchical(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_sge(
      data,
      chains = 2,
      warmup = 10,
      iter = 20,
      seed = 0,
      scheduler = "local" # change to "sge" for serious runs
    )
  )
)
mcmc
}
}
```

---

hbl_plot_borrow *Plot the hierarchical model response against the benchmark models.*

Description

Plot the response from a hierarchical model. against the independent and pooled benchmark models.

Usage

```
hbl_plot_borrow(
  borrow,
  pool,
  independent,
  outcome = c("response", "change", "diff")
)
```

Arguments

borrow	A data frame returned by hbl_summary() for the hierarchical model.
pool	A data frame returned by hbl_summary() for the pooled model.
independent	A data frame returned by hbl_summary() for the independent model.
outcome	Character of length 1, either "response", "change", or "diff": the quantity to plot on the vertical axis.

Value

A ggplot object

See Also

Other plot: hbl_plot_group(), hbl_plot_tau()

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_independent(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc_borrow <- hbl_mcmc_hierarchical(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc_pool <- hbl_mcmc_pool(
      data,
      chains = 1,
```

```
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc_independent <- hbl_mcmc_independent(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
borrow <- hbl_summary(mcmc_borrow, data)
pool <- hbl_summary(mcmc_pool, data)
independent <- hbl_summary(mcmc_independent, data)
hbl_plot_borrow(
  borrow = borrow,
  pool = pool,
  independent = independent
)
}
```

`hbl_plot_group` *Plot the groups of the hierarchical model and its benchmark models.*

Description

Plot the groups against one another for a hierarchical model. and the independent and pooled benchmark models.

Usage

```
hbl_plot_group(
  borrow,
  pool,
  independent,
  outcome = c("response", "change", "diff")
)
```

Arguments

`borrow`	A data frame returned by `hbl_summary()` for the hierarchical model.
`pool`	A data frame returned by `hbl_summary()` for the pooled model.
`independent`	A data frame returned by `hbl_summary()` for the independent model.

outcome — Character of length 1, either "response", "change", or "diff": the quantity to plot on the vertical axis.

Value

A ggplot object

See Also

Other plot: hbl_plot_borrow(), hbl_plot_tau()

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_independent(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc_borrow <- hbl_mcmc_hierarchical(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc_pool <- hbl_mcmc_pool(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc_independent <- hbl_mcmc_independent(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
```

```
  )
)
borrow <- hbl_summary(mcmc_borrow, data)
pool <- hbl_summary(mcmc_pool, data)
independent <- hbl_summary(mcmc_independent, data)
hbl_plot_group(
  borrow = borrow,
  pool = pool,
  independent = independent
)
}
```

`hbl_plot_tau` *Plot tau*

Description

Plot the rep-specific tau parameters of a fitted hierarchical model.

Usage

```
hbl_plot_tau(mcmc)
```

Arguments

`mcmc`	Data frame of posterior samples generated by `hbl_mcmc_hierarchical()`.

Value

A `ggplot` object

See Also

Other plot: `hbl_plot_borrow()`, `hbl_plot_group()`

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_independent(n_continuous = 2)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_hierarchical(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
```

```
)
hbl_plot_tau(mcmc)
}
```

`hbl_sim_hierarchical` *Non-longitudinal hierarchical simulations.*

Description

Simulate from the non-longitudinal hierarchical model.

Usage

```
hbl_sim_hierarchical(
  n_study = 5,
  n_group = 3,
  n_patient = 100,
  n_rep = 4,
  n_continuous = 0,
  n_binary = 0,
  constraint = FALSE,
  s_delta = 1,
  s_beta = 1,
  s_sigma = 1,
  s_lambda = 1,
  s_mu = 1,
  s_tau = 1,
  d_tau = 4,
  prior_tau = "half_t",
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  alpha = NULL,
 delta = stats::rnorm(n = (n_group - 1) * (n_rep - as.integer(constraint)), mean = 0, sd
    = s_delta),
 beta = stats::rnorm(n = n_study * (n_continuous + n_binary), mean = 0, sd = s_delta),
  sigma = stats::runif(n = n_study * n_rep, min = 0, max = s_sigma),
  mu = stats::rnorm(n = n_rep, mean = 0, sd = s_mu),
  tau = NULL,
  rho_current = stats::runif(n = 1, min = -1, max = 1),
  rho_historical = stats::runif(n = n_study - 1, min = -1, max = 1)
)
```

Arguments

`n_study`	Number of studies to simulate.
`n_group`	Number of groups (e.g. study arms) to simulate per study.
`n_patient`	Number of patients to simulate per study per group.

n_rep Number of repeated measures (time points) per patient.

n_continuous Number of continuous covariates to simulate (all from independent standard normal distributions).

n_binary Number of binary covariates to simulate (all from independent Bernoulli distributions with p = 0.5).

constraint Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

s_delta Numeric of length 1, prior standard deviation of the study-by-group effect parameters delta.

s_beta Numeric of length 1, prior standard deviation of the fixed effects beta.

s_sigma Numeric of length 1, prior upper bound of the residual standard deviations.

s_lambda shape parameter of the LKJ priors on the unstructured correlation matrices.

s_mu Numeric of length 1, prior standard deviation of mu.

s_tau Non-negative numeric of length 1. If prior_tau is "half_t", then s_tau is the scale parameter of the Student t prior of tau and analogous to the sigma parameter of the Student-t parameterization given at https://mc-stan.org/docs/functions-reference/unbounded_continuous_distributions.html. # nolint If prior_tau is "uniform", then s_tau is the upper bound of tau. Upper bound on tau if prior_tau is "uniform".

d_tau Positive numeric of length 1. Degrees of freedom of the Student t prior of tau if prior_tau is "half_t".

prior_tau Character string, family of the prior of tau. If prior_tau equals "uniform", then the prior on tau is a uniform prior with lower bound 0 and upper bound s_tau. If prior_tau equals "half_t", then the prior on tau is a half Student-t prior with center 0, lower bound 0, scale parameter s_tau, and degrees of freedom d_tau. The scale parameter s_tau is analogous to the sigma parameter of the Student-t parameterization given at https://mc-stan.org/docs/functions-reference/unbounded_continuous_distributions.html. # nolint

covariance_current Character of length 1, covariance structure of the current study. Possible values are "unstructured" for fully parameterized covariance matrices, "ar1" for AR(1) covariance matrices, and "diagonal" for residuals independent across time within each patient. In MCMC (e.g. hbl_mcmc_hierarchical()), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for covariance_historical if there are many historical studies in the data.

covariance_historical Same as covariance_current, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.

alpha Numeric vector of length n_rep for the pooled and model and length n_study * n_rep for the independent and hierarchical models. alpha is the vector of control group mean parameters. alpha enters the model by multiplying with $matrices$x_alpha (see the return value). The control group in the data is the one with the group column equal to 1.

delta	Numeric vector of length (n_group - 1) * (n_rep - as.integer(constraint)) of treatment effect parameters. delta enters the model by multiplying with $matrices$x_delta (see the return value). The control (non-treatment) group in the data is the one with the group column equal to 1.
beta	Numeric vector of n_study * (n_continuous + n_binary) fixed effect parameters. Within each study, the first n_continuous betas are for the continuous covariates, and the rest are for the binary covariates. All the betas for one study appear before all the betas for the next study, and studies are arranged in increasing order of the sorted unique values in $data$study in the output. betas enters the model by multiplying with $matrices$x_alpha (see the return value).
sigma	Numeric vector of n_study * n_rep residual standard deviation parameters for each study and rep. The elements are sorted with all the standard deviations of study 1 first (all the reps), then all the reps of study 2, etc.
mu	Numeric of length n_rep, mean of the control group means alpha for each rep.
tau	Numeric of length n_rep, standard deviation of the control group means alpha for each rep.
rho_current	Numeric of length 1 between -1 and 1, AR(1) residual correlation parameter for the current study.
rho_historical	Numeric of length n_study - 1 between -1 and 1, AR(1) residual correlation parameters for the historical studies.

Value

A list with the following elements:

- data: tidy long-form dataset with the patient-level data. one row per patient per rep and indicator columns for the study, group (e.g. treatment arm), patient ID, and rep. The response columns is the patient response. The other columns are baseline covariates. The control group is the one with the group column equal to 1, and the current study (non-historical) is the one with the maximum value of the study column. Only the current study has any non-control-group patients, the historical studies have only the control group.
- parameters: named list of model parameter values. See the model specification vignette for details.
- matrices: A named list of model matrices. See the model specification vignette for details.

See Also

Other simulate: hbl_sim_independent(), hbl_sim_pool()

Examples

```
hbl_sim_hierarchical(n_continuous = 1)$data
```

`hbl_sim_independent` *Longitudinal independent simulations.*

Description

Simulate from the longitudinal independent model.

Usage

```
hbl_sim_independent(
  n_study = 5,
  n_group = 3,
  n_patient = 100,
  n_rep = 4,
  n_continuous = 0,
  n_binary = 0,
  constraint = FALSE,
  s_alpha = 1,
  s_delta = 1,
  s_beta = 1,
  s_sigma = 1,
  s_lambda = 1,
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  alpha = stats::rnorm(n = n_study * n_rep, mean = 0, sd = s_alpha),
 delta = stats::rnorm(n = (n_group - 1) * (n_rep - as.integer(constraint)), mean = 0, sd
    = s_delta),
 beta = stats::rnorm(n = n_study * (n_continuous + n_binary), mean = 0, sd = s_delta),
  sigma = stats::runif(n = n_study * n_rep, min = 0, max = s_sigma),
  rho_current = stats::runif(n = 1, min = -1, max = 1),
  rho_historical = stats::runif(n = n_study - 1, min = -1, max = 1)
)
```

Arguments

`n_study`	Number of studies to simulate.
`n_group`	Number of groups (e.g. study arms) to simulate per study.
`n_patient`	Number of patients to simulate per study per group.
`n_rep`	Number of repeated measures (time points) per patient.
`n_continuous`	Number of continuous covariates to simulate (all from independent standard normal distributions).
`n_binary`	Number of binary covariates to simulate (all from independent Bernoulli distributions with p = 0.5).
`constraint`	Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

s_alpha Numeric of length 1, prior standard deviation of the study-specific control group mean parameters alpha.

s_delta Numeric of length 1, prior standard deviation of the study-by-group effect parameters delta.

s_beta Numeric of length 1, prior standard deviation of the fixed effects beta.

s_sigma Numeric of length 1, prior upper bound of the residual standard deviations.

s_lambda shape parameter of the LKJ priors on the unstructured correlation matrices.

covariance_current
Character of length 1, covariance structure of the current study. Possible values are "unstructured" for fully parameterized covariance matrices, "ar1" for AR(1) covariance matrices, and "diagonal" for residuals independent across time within each patient. In MCMC (e.g. hbl_mcmc_hierarchical()), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for covariance_historical if there are many historical studies in the data.

covariance_historical
Same as covariance_current, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.

alpha Numeric vector of length n_rep for the pooled and model and length n_study * n_rep for the independent and hierarchical models. alpha is the vector of control group mean parameters. alpha enters the model by multiplying with $matrices$x_alpha (see the return value). The control group in the data is the one with the group column equal to 1.

delta Numeric vector of length (n_group - 1) * (n_rep - as.integer(constraint)) of treatment effect parameters. delta enters the model by multiplying with $matrices$x_delta (see the return value). The control (non-treatment) group in the data is the one with the group column equal to 1.

beta Numeric vector of n_study * (n_continuous + n_binary) fixed effect parameters. Within each study, the first n_continuous betas are for the continuous covariates, and the rest are for the binary covariates. All the betas for one study appear before all the betas for the next study, and studies are arranged in increasing order of the sorted unique values in $data$study in the output. betas enters the model by multiplying with $matrices$x_alpha (see the return value).

sigma Numeric vector of n_study * n_rep residual standard deviation parameters for each study and rep. The elements are sorted with all the standard deviations of study 1 first (all the reps), then all the reps of study 2, etc.

rho_current Numeric of length 1 between -1 and 1, AR(1) residual correlation parameter for the current study.

rho_historical Numeric of length n_study - 1 between -1 and 1, AR(1) residual correlation parameters for the historical studies.

Value

A list with the following elements:

- `data`: tidy long-form dataset with the patient-level data. one row per patient per rep and indicator columns for the study, group (e.g. treatment arm), patient ID, and rep. The `response` columns is the patient response. The other columns are baseline covariates. The control group is the one with the `group` column equal to 1, and the current study (non-historical) is the one with the maximum value of the `study` column. Only the current study has any non-control-group patients, the historical studies have only the control group.
- `parameters`: named list of model parameter values. See the model specification vignette for details.
- `matrices`: A named list of model matrices. See the model specification vignette for details.

See Also

Other simulate: `hbl_sim_hierarchical()`, `hbl_sim_pool()`

Examples

```
hbl_sim_independent(n_continuous = 1)$data
```

`hbl_sim_pool` *Longitudinal pooled simulations.*

Description

Simulate from the longitudinal pooled model.

Usage

```
hbl_sim_pool(
  n_study = 5,
  n_group = 3,
  n_patient = 100,
  n_rep = 4,
  n_continuous = 0,
  n_binary = 0,
  constraint = FALSE,
  s_alpha = 1,
  s_delta = 1,
  s_beta = 1,
  s_sigma = 1,
  s_lambda = 1,
  covariance_current = "unstructured",
  covariance_historical = "unstructured",
  alpha = stats::rnorm(n = n_rep, mean = 0, sd = s_alpha),
 delta = stats::rnorm(n = (n_group - 1) * (n_rep - as.integer(constraint)), mean = 0, sd
    = s_delta),
 beta = stats::rnorm(n = n_study * (n_continuous + n_binary), mean = 0, sd = s_delta),
  sigma = stats::runif(n = n_study * n_rep, min = 0, max = s_sigma),
```

```
  rho_current = stats::runif(n = 1, min = -1, max = 1),
  rho_historical = stats::runif(n = n_study - 1, min = -1, max = 1)
)
```

Arguments

n_study	Number of studies to simulate.
n_group	Number of groups (e.g. study arms) to simulate per study.
n_patient	Number of patients to simulate per study per group.
n_rep	Number of repeated measures (time points) per patient.
n_continuous	Number of continuous covariates to simulate (all from independent standard normal distributions).
n_binary	Number of binary covariates to simulate (all from independent Bernoulli distributions with p = 0.5).
constraint	Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.
s_alpha	Numeric of length 1, prior standard deviation of the study-specific control group mean parameters alpha.
s_delta	Numeric of length 1, prior standard deviation of the study-by-group effect parameters delta.
s_beta	Numeric of length 1, prior standard deviation of the fixed effects beta.
s_sigma	Numeric of length 1, prior upper bound of the residual standard deviations.
s_lambda	shape parameter of the LKJ priors on the unstructured correlation matrices.
covariance_current	Character of length 1, covariance structure of the current study. Possible values are "unstructured" for fully parameterized covariance matrices, "ar1" for AR(1) covariance matrices, and "diagonal" for residuals independent across time within each patient. In MCMC (e.g. hbl_mcmc_hierarchical()), the covariance structure affects computational speed. Unstructured covariance is slower than AR(1), and AR(1) is slower than diagonal. This is particularly true for covariance_historical if there are many historical studies in the data.
covariance_historical	Same as covariance_current, but for the covariance structure of each separate historical study. Each historical study has its own separate covariance matrix.
alpha	Numeric vector of length n_rep for the pooled and model and length n_study * n_rep for the independent and hierarchical models. alpha is the vector of control group mean parameters. alpha enters the model by multiplying with $matrices$x_alpha (see the return value). The control group in the data is the one with the group column equal to 1.
delta	Numeric vector of length (n_group - 1) * (n_rep - as.integer(constraint)) of treatment effect parameters. delta enters the model by multiplying with $matrices$x_delta (see the return value). The control (non-treatment) group in the data is the one with the group column equal to 1.

`beta`	Numeric vector of `n_study * (n_continuous + n_binary)` fixed effect parameters. Within each study, the first `n_continuous` betas are for the continuous covariates, and the rest are for the binary covariates. All the `betas` for one study appear before all the `betas` for the next study, and studies are arranged in increasing order of the sorted unique values in `$data$study` in the output. `betas` enters the model by multiplying with `$matrices$x_alpha` (see the return value).
`sigma`	Numeric vector of `n_study * n_rep` residual standard deviation parameters for each study and rep. The elements are sorted with all the standard deviations of study 1 first (all the reps), then all the reps of study 2, etc.
`rho_current`	Numeric of length 1 between -1 and 1, AR(1) residual correlation parameter for the current study.
`rho_historical`	Numeric of length `n_study - 1` between -1 and 1, AR(1) residual correlation parameters for the historical studies.

Value

A list with the following elements:

- `data`: tidy long-form dataset with the patient-level data. one row per patient per rep and indicator columns for the study, group (e.g. treatment arm), patient ID, and rep. The `response` columns is the patient response. The other columns are baseline covariates. The control group is the one with the `group` column equal to 1, and the current study (non-historical) is the one with the maximum value of the `study` column. Only the current study has any non-control-group patients, the historical studies have only the control group.
- `parameters`: named list of model parameter values. See the model specification vignette for details.
- `matrices`: A named list of model matrices. See the model specification vignette for details.

See Also

Other simulate: `hbl_sim_hierarchical()`, `hbl_sim_independent()`

Examples

```
hbl_sim_pool(n_continuous = 1)$data
```

`hbl_summary` *Model summary*

Description

Summarize a fitted model in a table.

Usage

```
hbl_summary(
  mcmc,
  data,
  response = "response",
  response_type = "raw",
  study = "study",
  study_reference = max(data[[study]]),
  group = "group",
  group_reference = min(data[[group]]),
  patient = "patient",
  rep = "rep",
  rep_reference = min(data[[rep]]),
  covariates = grep("^covariate", colnames(data), value = TRUE),
  constraint = FALSE,
  eoi = 0,
  direction = "<"
)
```

Arguments

`mcmc`	A wide data frame of posterior samples returned by `hbl_mcmc_hierarchical()` or similar MCMC function.
`data`	Tidy data frame with one row per patient per rep, indicator columns for the response variable, study, group, patient, rep, and covariates. All columns must be atomic vectors (e.g. not lists).
`response`	Character of length 1, name of the column in `data` with the response/outcome variable. `data[[response]]` must be a continuous variable, and it *should* be the change from baseline of a clinical endpoint of interest, as opposed to just the raw response. Treatment differences are computed directly from this scale, please supply change from baseline unless you are absolutely certain that treatment differences computed directly from this quantity are clinically meaningful.
`response_type`	Character of length 1: `"raw"` if the response column in the data is the raw response, `"change"` if the response columns is change from baseline. In the latter case, the `change_*` columns in the output table are omitted because the response is already a change from baseline. Must be one of `"raw"` or `"change"`.
`study`	Character of length 1, name of the column in `data` with the study ID.
`study_reference`	Atomic of length 1, element of the `study` column that indicates the current study. (The other studies are historical studies.)
`group`	Character of length 1, name of the column in `data` with the group ID.
`group_reference`	Atomic of length 1, element of the `group` column that indicates the control group. (The other groups may be treatment groups.)
`patient`	Character of length 1, name of the column in `data` with the patient ID.
`rep`	Character of length 1, name of the column in `data` with the rep ID.

`rep_reference` Atomic of length 1, element of the `rep` column that indicates baseline, i.e. the first rep chronologically. (The other reps may be post-baseline study visits or time points.)

`covariates` Character vector of column names in `data` with the columns with baseline covariates. These can be continuous, categorical, or binary. Regardless, `historicalborrowlong` derives the appropriate model matrix.

Each baseline covariate column must truly be a *baseline* covariate: elements must be equal for all time points within each patient (after the steps in the "Data processing" section). In other words, covariates must not be time-varying.

A large number of covariates, or a large number of levels in a categorical covariate, can severely slow down the computation. Please consider carefully if you really need to include such complicated baseline covariates.

`constraint` Logical of length 1, whether to pool all study arms at baseline (first rep). Appropriate when the response is the raw response (as opposed to change from baseline) and the first rep (i.e. time point) is prior to treatment.

`eoi` Numeric of length at least 1, vector of effects of interest (EOIs) for critical success factors (CSFs).

`direction` Character of length `length(eoi)` indicating how to compare the treatment effect to each EOI. `">"` means Prob(treatment effect > EOI), and `"<"` means Prob(treatment effect < EOI). All elements of `direction` must be either `">"` or `"<"`.

Details

The `hb_summary()` function post-processes the results from the model. It estimates marginal means of the response, treatment effect, and other quantities of interest.

Value

A tidy data frame with one row per group (e.g. treatment arm) and the columns in the following list. Unless otherwise specified, the quantities are calculated at the group-by-rep level. Some are calculated for the current (non-historical) study only, while others pertain to the combined dataset which includes all historical studies.

- `group`: group index.
- `group_label`: original group label in the data.
- `rep`: rep index.
- `rep_label`: original rep label in the data.
- `data_mean`: observed mean of the response specific to the current study.
- `data_sd`: observed standard deviation of the response specific to the current study.
- `data_lower`: lower bound of a simple frequentist 95% confidence interval of the observed data mean specific to the current study.
- `data_upper`: upper bound of a simple frequentist 95% confidence interval of the observed data mean specific to the current study.
- `data_n`: number of non-missing observations in the combined dataset (all studies).

- `data_N`: total number of observations (missing and non-missing) in the combined dataset (all studies).
- `data_n_study_*`: number of non-missing observations in each study. The suffixes of these column names are integer study indexes. Call `dplyr::distinct(hbl_data(your_data), study, study_label)` to see which study labels correspond to these integer indexes.
- `data_N_study_*`: total number of observations (missing and non-missing) within each study. The suffixes of these column names are integer study indexes. Call `dplyr::distinct(hbl_data(your_data), study, study_label)` to see which study labels correspond to these integer indexes.
- `response_mean`: Estimated posterior mean of the response from the model. (Here, the response variable in the data should be a change from baseline outcome.) Specific to the current study.
- `response_sd`: Estimated posterior standard deviation of the mean response from the model. Specific to the current study.
- `response_variance`: Estimated posterior variance of the mean response from the model. Specific to the current study.
- `response_lower`: Lower bound of a 95% posterior interval on the mean response from the model. Specific to the current study.
- `response_upper`: Upper bound of a 95% posterior interval on the mean response from the model. Specific to the current study.
- `response_mean_mcse`: Monte Carlo standard error of `response_mean`.
- `response_sd_mcse`: Monte Carlo standard error of `response_sd`.
- `response_lower_mcse`: Monte Carlo standard error of `response_lower`.
- `response_upper_mcse`: Monte Carlo standard error of `response_upper`.
- `change_*`: same as the `response_*` columns, but for change from baseline instead of the response. Not included if `response_type` is `"change"` because in that case the response is already change from baseline.
- `change_percent_*`: same as the `change_*` columns, but for the *percent* change from baseline (from 0% to 100%). Not included if `response_type` is `"change"` because in that case the response is already change from baseline. Specific to the current study.
- `diff_*`: same as the `response_*` columns, but for treatment effect.
- `P(diff > EOI)`, `P(diff < EOI)`: CSF probabilities on the treatment effect specified with the `eoi` and `direction` arguments. Specific to the current study.
- `effect_mean`: same as the `response_*` columns, but for the effect size (diff / residual standard deviation). Specific to the current study.
- `precision_ratio*`: same as the `response_*` columns, but for the precision ratio, which compares within-study variance to among-study variance. Only returned for the hierarchical model. Specific to the current study.

See Also

Other summary: `hbl_ess()`

Examples

```
if (!identical(Sys.getenv("HBL_TEST", unset = ""), "")) {
set.seed(0)
data <- hbl_sim_pool(
  n_study = 2,
  n_group = 2,
  n_patient = 5,
  n_rep = 3
)$data
tmp <- utils::capture.output(
  suppressWarnings(
    mcmc <- hbl_mcmc_hierarchical(
      data,
      chains = 1,
      warmup = 10,
      iter = 20,
      seed = 0
    )
  )
)
hbl_summary(mcmc, data)
}
```

Index

∗ **data**
    hbl_data, 4
∗ **help**
    historicalborrowlong-package, 2
∗ **mcmc**
    hbl_convergence, 3
    hbl_mcmc_hierarchical, 9
    hbl_mcmc_independent, 13
    hbl_mcmc_pool, 17
    hbl_mcmc_sge, 21
∗ **plot**
    hbl_plot_borrow, 25
    hbl_plot_group, 27
    hbl_plot_tau, 29
∗ **simulate**
    hbl_sim_hierarchical, 30
    hbl_sim_independent, 33
    hbl_sim_pool, 35
∗ **summary**
    hbl_ess, 6
    hbl_summary, 37

detectCores, *24*

hbl_convergence, 3
hbl_convergence(), *13*, *17*, *20*, *25*
hbl_data, 4
hbl_ess, 6
hbl_ess(), *40*
hbl_mcmc_hierarchical, 9
hbl_mcmc_hierarchical(), *3*, *7*, *11*, *15*, *17*, *19*, *20*, *23*, *25*, *29*, *31*, *34*, *36*, *38*
hbl_mcmc_independent, 13
hbl_mcmc_independent(), *3*, *13*, *20*, *25*
hbl_mcmc_pool, 17
hbl_mcmc_pool(), *3*, *7*, *13*, *17*, *25*
hbl_mcmc_sge, 21
hbl_mcmc_sge(), *3*, *13*, *17*, *20*
hbl_plot_borrow, 25
hbl_plot_borrow(), *28*, *29*
hbl_plot_group, 27
hbl_plot_group(), *26*, *29*
hbl_plot_tau, 29
hbl_plot_tau(), *26*, *28*
hbl_sim_hierarchical, 30
hbl_sim_hierarchical(), *35*, *37*
hbl_sim_independent, 33
hbl_sim_independent(), *32*, *37*
hbl_sim_pool, 35
hbl_sim_pool(), *32*, *35*
hbl_summary, 37
hbl_summary(), *8*, *26*, *27*
historicalborrowlong-package, 2