Package: ProbBreed (via r-universe)

October 29, 2026

Title Probability Theory for Selecting Candidates in Plant Breeding

Version 1.0.4.9

Description Use probability theory under the Bayesian framework for calculating the risk of selecting candidates in a multi-environment context. Contained are functions used to fit a Bayesian multi-environment model (based on the available presets), extract posterior values and maximum posterior values, compute the variance components, check the model's convergence, and calculate the probabilities. For both across and within-environments scopes, the package computes the probability of superior performance and the pairwise probability of superior performance. Furthermore, the probability of superior stability and the pairwise probability of superior stability across environments is estimated. A joint probability of superior performance and stability is also provided.

URL https://github.com/saulo-chaves/ProbBreed, https://saulo-chaves.github.io/ProbBreed_site/, https://saulo-chaves.github.io/ProbBreed/

BugReports https://github.com/saulo-chaves/ProbBreed/issues

License AGPL (>= 3)

Depends R (>= 3.5.0)

Imports ggplot2, lifecycle, methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.32.0), rstantools (>= 2.4.0), stats, utils

LinkingTo StanHeaders (>= 2.32.0), rstan (>= 2.32.0), BH (>= 1.72.0-2), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1)

Suggests knitr, rmarkdown

Encoding UTF-8

UseLTO true

NeedsCompilation yes

RoxygenNote 7.3.3

LazyData true

Biarch true

SystemRequirements GNU make

Author Saulo Chaves [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0694-1798>), Kaio Dias [aut, cph]
(ORCID: <https://orcid.org/0000-0002-9171-1021>), Matheus
Krause [aut] (ORCID: <https://orcid.org/0000-0003-2411-9287>)

Maintainer Saulo Chaves <saulochaves@usp.br>

Config/pak/sysreqs make

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

Date/Publication 2025-12-15 20:10:23 UTC

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

RemoteRef HEAD

RemoteSha 50ad3fef01bd97c6bff94bca09adc99f04219290

Contents

bayes_met 2
bpsi 7
extr_outs 9
maize 10
plot.bpsi 11
plot.extr 13
plot.probsup 14
print.bpsi 16
print.extr 17
print.probsup 17
prob_sup 18
soy 21
soy_pat 22

Index **24**

`bayes_met` *Bayesian model for multi-environment trials*

Description

Fits a Bayesian multi-environment model using `rstan`, the `R` interface to `Stan`.

Usage

```
bayes_met(
  data,
  gen,
  loc,
  repl,
  trait,
  reg = NULL,
  year = NULL,
  res.het = FALSE,
  iter = 2000,
  cores = 1,
  chains = 4,
  pars = NA,
  warmup = floor(iter/2),
  thin = 1,
  seed = sample.int(.Machine$integer.max, 1),
  init = "random",
  verbose = FALSE,
  algorithm = c("NUTS", "HMC", "Fixed_param"),
  control = NULL,
  include = TRUE,
  show_messages = TRUE,
  ...
)
```

Arguments

data	A data frame in which to interpret the variables declared in the other arguments.
gen, loc	A string. The name of the columns that contain the evaluated candidates and locations (or environments, if you are working with factor combinations), respectively.
repl	A string, a vector, or NULL. If the trial is randomized in complete blocks design, repl will be a string representing the name of the column that corresponds to the blocks. If the trial is randomized in incomplete blocks design, repl will be a string vector containing the name of the columns that correspond to the replicate and block effects on the first and second positions, respectively (c(replicate, block)). If the data does not have replicates, repl must be NULL.
trait	A string. The analysed variable. Currently, only single-trait models are fitted.
reg	A string or NULL. The name of the column that contain information on regions or mega-environments. NULL (default) if not applicable.
year	A string or NULL. The name of the column that contain information on years (or seasons). NULL (default) if not applicable.
res.het	Should the model consider heterogeneous residual variances? Defaults for FALSE. If TRUE, the model will estimate one residual variance per location (or environmnet).

iter	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
cores	Number of cores to use when executing the chains in parallel, which defaults to 1 but we recommend setting the mc.cores option to be as many processors as the hardware and RAM allow (up to the number of chains).
chains	A positive integer specifying the number of Markov chains. The default is 4.
pars	A vector of character strings specifying parameters of interest. The default is NA indicating all parameters in the model. If include = TRUE, only samples for parameters named in pars are stored in the fitted results. Conversely, if include = FALSE, samples for all parameters *except* those named in pars are stored in the fitted results.
warmup	A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than iter and the default is iter/2.
thin	A positive integer specifying the period for saving samples. The default is 1, which is usually the recommended value.
seed	The seed for random number generation. The default is generated from 1 to the maximum integer supported by R on the machine. Even if multiple chains are used, only one seed is needed, with other chains having seeds derived from that of the first chain to avoid dependent samples. When a seed is specified by a number, as.integer will be applied to it. If as.integer produces NA, the seed is generated randomly. The seed can also be specified as a character string of digits, such as "12345", which is converted to integer.
init	Initial values specification. See the detailed documentation for the init argument in stan.
verbose	TRUE or FALSE: flag indicating whether to print intermediate output from Stan on the console, which might be helpful for model debugging.
algorithm	One of sampling algorithms that are implemented in Stan. Current options are "NUTS" (No-U-Turn sampler, Hoffman and Gelman 2011, Betancourt 2017), "HMC" (static HMC), or "Fixed_param". The default and preferred algorithm is "NUTS".
control	A named list of parameters to control the sampler's behavior. See the details in the documentation for the control argument in stan.
include	Logical scalar defaulting to TRUE indicating whether to include or exclude the parameters given by the pars argument. If FALSE, only entire multidimensional parameters can be excluded, rather than particular elements of them.
show_messages	Either a logical scalar (defaulting to TRUE) indicating whether to print the summary of Informational Messages to the screen after a chain is finished or a character string naming a path where the summary is stored. Setting to FALSE is not recommended unless you are very sure that the model is correct up to numerical error.
...	Additional arguments can be chain_id, init_r, test_grad, append_samples, refresh, enable_random_init. See the documentation in stan.

Details

The function has nine available models, which will be fitted according to the options set in the arguments:

1. Entry-mean model : fitted when `repl` = `NULL`, `reg` = `NULL` and `year` = `NULL`:

$$y = \mu + g + l + \varepsilon$$

   Where y is the phenotype, μ is the intercept, g is the genotypic effect, l is the location (or environment) effect, and ε is the error (which contains the genotype-by-location interaction, in this case).

2. Randomized complete blocks design : fitted when `repl` is a single string. It will fit different models depending if `reg` and `year` are `NULL`:
   - `reg` = `NULL` and `year` = `NULL` :

$$y = \mu + g + l + gl + r + \varepsilon$$

     where gl is the genotype-by-location effect, and r is the replicate effect.
   - `reg` = `"reg"` and `year` = `NULL` :

$$y = \mu + g + m + l + gl + gm + r + \varepsilon$$

     where m is the region effect, and gm is the genotype-by-region effect.
   - `reg` = `NULL` and `year` = `"year"` :

$$y = \mu + g + t + l + gl + gt + r + \varepsilon$$

     where t is the year effect, and gt is the genotype-by-year effect.
   - `reg` = `"reg"` and `year` = `"year"` :

$$y = \mu + g + m + t + l + gl + gm + gt + r + \varepsilon$$

3. Incomplete blocks design : fitted when `repl` is a string vector of size 2. It will fit different models depending if `reg` and `year` are `NULL`:
   - `reg` = `NULL` and `year` = `NULL` :

$$y = \mu + g + l + gl + r + b + \varepsilon$$

     where b is the block within replicates effect.
   - `reg` = `"reg"` and `year` = `NULL` :

$$y = \mu + g + m + l + gl + gm + r + b + \varepsilon$$

   - `reg` = `NULL` and `year` = `"year"` :

$$y = \mu + g + t + l + gl + gt + r + b + \varepsilon$$

   - `reg` = `"reg"` and `year` = `"year"` :

$$y = \mu + g + m + t + l + gl + gm + gt + r + b + \varepsilon$$

The models described above have predefined priors:

$$x \sim \mathcal{N}\left(0, S^{[x]}\right)$$

$$\sigma \sim \mathcal{H}\dashv\updownarrow\{\mathcal{C}\dashv\sqcap\rfloor\langle\dagger\left(0, S^{[\sigma]}\right)$$

where x can be any effect but the error, and σ is the standard deviation of the likelihood. If `res.het = TRUE`, then $\sigma_k \sim \mathcal{H}\dashv\updownarrow\{\mathcal{C}\dashv\sqcap\rfloor\langle\dagger\left(0, S^{[\sigma_k]}\right)$. The hyperpriors are set as follows:

$$S^{[x]} \sim \mathcal{H}\dashv\updownarrow\{\mathcal{C}\dashv\sqcap\rfloor\langle\dagger\left(0, \phi\right)$$

where ϕ is the known global hyperparameter defined such as $\phi = max(y) \times 10$.

More details about the usage of `bayes_met` and other functions of the `ProbBreed` package can be found at https://saulo-chaves.github.io/ProbBreed_site/. Solutions to convergence or mixing issues can be found at https://mc-stan.org/misc/warnings.html.

Value

An object of S4 class `stanfit` representing the fitted results. Slot `mode` for this object indicates if the sampling is done or not.

Methods

`sampling signature(object = "stanmodel")` Call a sampler (NUTS, HMC, or Fixed_param depending on parameters) to draw samples from the model defined by S4 class `stanmodel` given the data, initial values, etc.

See Also

rstan::sampling, rstan::stan, rstan::stanfit

Examples

```
mod = bayes_met(data = maize,
                gen = "Hybrid",
                loc = "Location",
                repl = c("Rep","Block"),
                trait = "GY",
                reg = "Region",
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)
```

bpsi *Bayesian Probabilistic Selection Index (BPSI)*

Description

This function estimates the genotype's merit for multiple traits using the probabilities of superior performance across environments.

Usage

```
bpsi(problist, increase = NULL, lambda = NULL, int, save.df = FALSE)
```

Arguments

problist	A list of object of class probsup, obtained from the prob_sup function
increase	Optional logical vector with size corresponding to the number of traits of problist, in the same order.TRUE if the selection is for increasing the trait value, FALSE otherwise. If not declared, bpsi will consider the information provided in prob_sup
lambda	A numeric representing the weight of each trait. Defaults to 1 (equal weights). The trait with more economic interest should be greater.
int	A numeric representing the selection intensity (between 0 and 1), considering the selection index.
save.df	Logical. Should the data frames be saved in the work directory? TRUE for saving, FALSE (default) otherwise.

Details

- Bayesian Probabilistic Selection Index

$$BPSI_i = \sum_{m=1}^{t} \frac{\gamma_{pt} - \gamma_{it}}{(1/\lambda_t)}$$

where γ_p is the probability of superior performance of the worst genotype for the trait t, γ is the probability of superior performance of genotype i for trait t, t is the total number of traits evaluated, $(m = 1, 2, ..., t)$, and λ is the weight for each trait t.

More details about the usage of bpsi can be found at https://tiagobchagas.github.io/BPSI/.

Value

The function returns an object of class bpsi, which contains two lists, one with the BPSI- Bayesian Probabilistic Selection Index, and another with the original data- with across-environments probabilities of superior performance for each trait.

Author(s)

José Tiago Barroso Chagas

References

Chagas, J. T. B., Dias, K. O. G., Carneiro, V. Q., Oliveira, L. M. C., Nunes, N. X., Pereira Júnior, J. D., Carneiro, P. C. S., & Carneiro, J. E. S. (2025). Bayesian probabilistic selection index in the selection of common bean families. *Crop Science*, 65(3). doi:10.1002/CSC2.70072

See Also

`plot.bpsi()`

Examples

```
mod = bayes_met(data = soy_pat,
                gen = "gen",
                loc = "env",
                repl = NULL,
                trait = "PH",
                reg = NULL,
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)


mod2 = bayes_met(data = soy_pat,
                 gen = "gen",
                 loc = "env",
                 repl = NULL,
                 trait = "GY",
                 reg = NULL,
                 year = NULL,
                 res.het = TRUE,
                 iter = 2000, cores = 2, chain = 4)

mod3 = bayes_met(data = soy_pat,
                 gen = "gen",
                 loc = "env",
                 repl =  NULL,
                 trait = "NDM",
                 reg = NULL,
                 year = NULL,
                 res.het = TRUE,
                 iter = 2000, cores = 2, chain = 4)



models=list(mod,mod2,mod3)
names(models) <- c("PH","GY","NDM")
increase = c(FALSE,TRUE,FALSE)
names(increase) <- names(models)
```

```
probs = list()
for (i in names(models)) {
  outs <- extr_outs(model = models[[i]],
                    probs = c(0.05, 0.95),
                    verbose = TRUE)
  probs[[i]] <- prob_sup(
    extr = outs,
    int = .2,
    increase = increase[[i]],
    save.df = FALSE,
    verbose = TRUE
  )

}

index = bpsi(
  problist = probs,
  increase = increase,
  int = 0.1,
  lambda = c(1, 2, 1),
  save.df = FALSE
)
```

`extr_outs` *Extract outputs from* `stanfit` *objects obtained from bayes_met*

Description

Extracts outputs of the Bayesian model fitted using `bayes_met()`, and provides some diagnostics.

Usage

```
extr_outs(model, probs = c(0.025, 0.975), verbose = FALSE)
```

Arguments

`model`	An object of class `stanfit`, obtained using `bayes_met()`
`probs`	A vector with two elements representing the probabilities (in decimal scale) that will be considered for computing the quantiles.
`verbose`	A logical value. If `TRUE`, the function will indicate the completed steps. Defaults to `FALSE`

Details

More details about the usage of `extr_outs` and other functions of the `ProbBreed` package can be found at https://saulo-chaves.github.io/ProbBreed_site/.

Value

The function returns an object of class `extr`, which is a list with:

- `variances` : a data frame containing the variance components of the model effects, their standard deviation, naive standard error and highest posterior density interval.
- `post` : a list with the posterior of the effects, and the data generated by the model.
- `map` : a list with the maximum posterior values of each effect
- `ppcheck` : a matrix containing the p-values of maximum, minimum, median, mean and standard deviation; effective number of parameters, WAIC2 value, Rhat and effective sample size.

See Also

rstan::stan_diag, ggplot2::ggplot, rstan::check_hmc_diagnostics, plot.extr

Examples

```
mod = bayes_met(data = maize,
                gen = "Hybrid",
                loc = "Location",
                repl = c("Rep","Block"),
                trait = "GY",
                reg = "Region",
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)

outs = extr_outs(model = mod,
                 probs = c(0.05, 0.95),
                 verbose = TRUE)
```

`maize` *Maize real dataset*

Description

This dataset belongs to value of cultivation and use maize trials of Embrapa Maize and Sorghum, and was used by Dias et al. (2022). It contains the grain yield of 32 single-cross hybrids and four commercial checks (36 genotypes in total) evaluated in 16 locations across five regions or mega-environments. These trials were laid out in incomplete blocks design, using a block size of 6 and two replications per trial.

Usage

```
maize
```

Format

`maize`:
A data frame with 823 rows and 6 columns:

Location 16 locations

Region 5 regions

Rep 2 replicates

Block 6 blocks

Hybrid 36 genotypes

GY Grain yield (phenotypes)

Source

Dias, K. O. G, Santos, J. P. R., Krause, M. D., Piepho, H. -P., Guimarães, L. J. M., Pastina, M. M., and Garcia, A. A. F. (2022). Leveraging probability concepts for cultivar recommendation in multi-environment trials. *Theoretical and Applied Genetics*, 133(2):443-455. doi:10.1007/s0012202204041y

---

`plot.bpsi` *Plots for the* `bpsi` *object*

Description

Build plots using the outputs stored in the `bpsi` object.

Usage

```
## S3 method for class 'bpsi'
plot(x, ..., category = "BPSI")
```

Arguments

`x`	An object of class `bpsi`.
`...`	currently not used
`category`	A string indicating which plot to build. There are currently two types of visualizations. Set "Ranks" for bar plots along each trait and "BPSI" (default) for circular bar plots multitrait.

Author(s)

José Tiago Barroso Chagas

References

Chagas, J. T. B., Dias, K. O. das G., Quintão Carneiro, V., de Oliveira, L. M. C., Nunes, N. X., Júnior, J. D. P., Carneiro, P. C. S., & Carneiro, J. E. de S. (2025). Bayesian probabilistic selection index in the selection of common bean families. *Crop Science*, 65(3).doi:10.1002/CSC2.70072

See Also

bpsi

Examples

```
mod = bayes_met(data = soy_pat,
                gen = "gen",
                loc = "env",
                repl = NULL,
                trait = "PH",
                reg = NULL,
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)


mod2 = bayes_met(data = soy_pat,
                 gen = "gen",
                 loc = "env",
                 repl = NULL,
                 trait = "GY",
                 reg = NULL,
                 year = NULL,
                 res.het = TRUE,
                 iter = 2000, cores = 2, chain = 4)

mod3 = bayes_met(data = soy_pat,
                 gen = "gen",
                 loc = "env",
                 repl =  NULL,
                 trait = "NDM",
                 reg = NULL,
                 year = NULL,
                 res.het = TRUE,
                 iter = 2000, cores = 2, chain = 4)



models=list(mod,mod2,mod3)
names(models) <- c("PH","GY","NDM")
increase = c(FALSE,TRUE,FALSE)
names(increase) <- names(models)

probs = list()
for (i in names(models)) {
  outs <- extr_outs(model = models[[i]],
                    probs = c(0.05, 0.95),
                    verbose = TRUE)
  probs[[i]] <- prob_sup(
```

```
    extr = outs,
    int = .2,
    increase = increase[[i]],
    save.df = FALSE,
    verbose = TRUE
  )

}

index = bpsi(
  problist = probs,
  increase = increase,
  int = 0.1,
  lambda = c(1, 2, 1),
  save.df = FALSE
)

plot(index, category = "BPSI")
plot(index, category = "Ranks")
```

`plot.extr` *Plots for the* `extr` *object*

Description

Build plots using the outputs stored in the `extr` object.

Usage

```
## S3 method for class 'extr'
plot(x, ..., category = "ppdensity")
```

Arguments

`x`	An object of class `extr`.
`...`	Passed to ggplot2::geom_histogram, when `category = histogram`. Useful to change the number of bins.
`category`	A string indicating which plot to build. See options in the Details section.

Details

The available options are:

- `ppdensity` : Density plots of the empirical and sampled data, useful to assess the model's convergence.
- `density` : Density plots of the model's effects.
- `histogram` : Histograms of the model's effects.
- `traceplot`: Trace plot showing the changes in the effects' values across iterations and chains.

See Also

extr_outs

Examples

```
mod = bayes_met(data = maize,
                gen = "Hybrid",
                loc = "Location",
                repl = c("Rep","Block"),
                trait = "GY",
                reg = "Region",
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)

outs = extr_outs(model = mod,
                 probs = c(0.05, 0.95),
                 verbose = TRUE)
plot(outs, category = "ppdensity")
plot(outs, category = "density")
plot(outs, category = "histogram")
plot(outs, category = "traceplot")
```

`plot.probsup` *Plots for the* `probsup` *object*

Description

Build plots using the outputs stored in the `probsup` object.

Usage

```
## S3 method for class 'probsup'
plot(x, ..., category = "perfo", level = "across")
```

Arguments

`x`	An object of class `probsup`.
`...`	currently not used
`category`	A string indicating which plot to build. See options in the Details section.
`level`	A string indicating the information level to be used for building the plots. Options are `"across"` for focusing on the probabilities across environments, or `"within"` to focus on the within-environment effects. Defaults to `"across"`.

Details

The available options are:

- hpd : a caterpillar plot representing the marginal genotypic value of each genotype, and their respective highest posterior density interval (95% represented by the thick line, and 97.5% represented by the thin line). Available only if `level = "across"`.
- `perfo` : if `level = "across"`, a lollipop plot illustrating the probabilities of superior performance. If `level = "within"`, a heatmap with the probabilities of superior performance within environments. If a model with `reg` and/or `year` is fitted, multiple plots are produced.
- `stabi`: a lollipop plot with the probabilities of superior stability. If a model with `reg` and/or `year` is fitted, multiple plots are produced. Available only if `level = "across"`. Unavailable if an entry-mean model was used in `bayes_met`.
- `pair_perfo` : if `level = "across"`, a heatmap representing the pairwise probability of superior performance (the probability of genotypes at the *x*-axis being superior. to those on the *y*-axis). If `level = "within"`, a list of heatmaps representing the pairwise probability of superior performance within environments. If a model with `reg` and/or `year` is fitted, multiple plots (and multiple lists) are produced. Should this option is set, it is mandatory to store the outputs in an object. (e.g., `pl <- plot(obj, category = "pair_perfo", level = "within")`) so they can be visualized one at a time. The option `level = "within"` is unavailable if an entry-mean model was used in `bayes_met`.
- `pair_stabi`: a heatmap with the pairwise probabilities of superior stability (the probability of genotypes at the *x*-axis being more stable than those on the *y*-axis). If a model with `reg` and/or `year` is fitted, multiple plots are produced. Available only if `level = "across"`. Unavailable if an entry-mean model was used in `bayes_met`.
- `joint`: a lollipop plot with the joint probabilities of superior performance and stability. Unavailable if an entry-mean model was used in `bayes_met`.

See Also

prob_sup

Examples

```
mod = bayes_met(data = maize,
                gen = "Hybrid",
                loc = "Location",
                repl = c("Rep","Block"),
                trait = "GY",
                reg = "Region",
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)

outs = extr_outs(model = mod,
                 probs = c(0.05, 0.95),
                 verbose = TRUE)

results = prob_sup(extr = outs,
                   int = .2,
```

```
                    increase = TRUE,
                    save.df = FALSE,
                    verbose = FALSE)

plot(results, category = "hpd")
plot(results, category = "perfo", level = "across")
plot(results, category = "perfo", level = "within")
plot(results, category = "stabi")
plot(results, category = "pair_perfo", level = "across")
plwithin = plot(results, category = "pair_perfo", level = "within")
plot(results, category = "pair_stabi")
plot(results, category = "joint")
```

`print.bpsi` *Print an object of class* `bpsi`

Description

Print a `bpsi` object in R console

Usage

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

Arguments

`x`	An object of class `bpsi`
`...`	currently not used

Author(s)

José Tiago Barroso Chagas

See Also

bpsi

print.extr *Print an object of class* extr

Description

Print a extr object in R console

Usage

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

Arguments

x	An object of class extr
...	currently not used

See Also

extr_outs

print.probsup *Print an object of class* probsup

Description

Print a probsup object in R console

Usage

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

Arguments

x	An object of class probsup
...	currently not used

See Also

prob_sup

prob_sup *Probabilities of superior performance and stability*

Description

This function estimates the probabilities of superior performance and stability across environments, and probabilities of superior performance within environments.

Usage

```
prob_sup(extr, int, increase = TRUE, save.df = FALSE, verbose = FALSE)
```

Arguments

extr	An object of class extr, obtained from the extr_outs function
int	A numeric representing the selection intensity (between 0 and 1)
increase	Logical.TRUE (default) if the selection is for increasing the trait value, FALSE otherwise.
save.df	Logical. Should the data frames be saved in the work directory? TRUE for saving, FALSE (default) otherwise.
verbose	A logical value. If TRUE, the function will indicate the completed steps. Defaults to FALSE.

Details

Probabilities provide the risk of recommending a selection candidate for a target population of environments or for a specific environment. prob_sup computes the probabilities of superior performance and the probabilities of superior stability:

- Probability of superior performance

Let Ω represent the subset of selected genotypes based on their performance across environments. A given genotype j will belong to Ω if its genotypic marginal value ($\hat{g}_j$) is high or low enough compared to its peers. prob_sup leverages the Monte Carlo discretized sampling from the posterior distribution to emulate the occurrence of S trials. Then, the probability of the j^{th} genotype belonging to Ω is the ratio of success ($\hat{g}_j \in \Omega$) events and the total number of sampled events, as follows:

$$Pr\left(\hat{g}_j \in \Omega | y\right) = \frac{1}{S}\sum_{s=1}^{S} I\left(\hat{g}_j^{(s)} \in \Omega | y\right)$$

where S is the total number of samples ($s = 1, 2, ..., S$), and $I\left(g_j^{(s)} \in \Omega | y\right)$ is an indicator variable that can assume two values: (1) if $\hat{g}_j^{(s)} \in \Omega$ in the s^{th} sample, and (0) otherwise. S is conditioned to the number of iterations and chains previously set at bayes_met.

Similarly, the within-environment probability of superior performance can be applied to individual environments. Let Ω_k represent the subset of superior genotypes in the k^{th} environment, so that the probability of the $j^{th} \in \Omega_k$ can calculated as follows:

$$Pr\left(\hat{g}_{jk} \in \Omega_k | y\right) = \frac{1}{S}\sum_{s=1}^{S} I\left(\hat{g}_{jk}^{(s)} \in \Omega_k | y\right)$$

where $I\left(\hat{g}_{jk}^{(s)} \in \Omega_k | y\right)$ is an indicator variable mapping success (1) if $\hat{g}_{jk}^{(s)}$ exists in Ω_k, and failure (0) otherwise, and $\hat{g}_{jk}^{(s)} = \hat{g}_{j}^{(s)} + \widehat{ge}_{jk}^{(s)}$. Note that when computing within-environment probabilities, we are accounting for the interaction of the j^{th} genotype with the k^{th} environment.

The pairwise probabilities of superior performance can also be calculated across or within environments. This metric assesses the probability of the j^{th} genotype being superior to another experimental genotype or a commercial check. The calculations are as follows, across and within environments, respectively:

$$Pr\left(\hat{g}_{j} > \hat{g}_{j'} | y\right) = \frac{1}{S}\sum_{s=1}^{S} I\left(\hat{g}_{j}^{(s)} > \hat{g}_{j'}^{(s)} | y\right)$$

or

$$Pr\left(\hat{g}_{jk} > \hat{g}_{j'k} | y\right) = \frac{1}{S}\sum_{s=1}^{S} I\left(\hat{g}_{jk}^{(s)} > \hat{g}_{j'k}^{(s)} | y\right)$$

These equations are set for when the selection direction is positive. If `increase = FALSE`, $>$ is simply switched by $<$.

- Probability of superior stability

This probability makes a direct analogy with the method of Shukla (1972): a stable genotype is the one that has a low genotype-by-environment interaction variance $[var(\widehat{ge})]$. Using the same probability principles previously described, the probability of superior stability is given as follows:

$$Pr\left[var\left(\widehat{ge}_{jk}\right) \in \Omega | y\right] = \frac{1}{S}\sum_{s=1}^{S} I\left[var\left(\widehat{ge}_{jk}^{(s)}\right) \in \Omega | y\right]$$

where $I\left[var\left(\widehat{ge}_{jk}^{(s)}\right) \in \Omega | y\right]$ indicates if $var\left(\widehat{ge}_{jk}^{(s)}\right)$ exists in Ω (1) or not (0). Pairwise probabilities of superior stability are also possible in this context:

$$Pr\left[var\left(\widehat{ge}_{jk}\right) < var\left(\widehat{ge}_{j'k}\right) | y\right] = \frac{1}{S}\sum_{s=1}^{S} I\left[var\left(\widehat{ge}_{jk}\right)^{(s)} < var\left(\widehat{ge}_{j'k}\right)^{(s)} | y\right]$$

Note that j will be superior to j' if it has a **lower** variance of the genotype-by-environment interaction effect. This is true regardless if `increase` is set to `TRUE` or `FALSE`.

The joint probability independent events is the product of the individual probabilities. The estimated genotypic main effects and the variances of GEI effects are independent by design, thus the joint probability of superior performance and stability as follows:

$$Pr\left[\hat{g}_j \in \Omega \cap var\left(\widehat{ge}_{jk}\right) \in \Omega\right] = Pr\left(\hat{g}_j \in \Omega\right) \times Pr\left[var\left(\widehat{ge}_{jk}\right) \in \Omega\right]$$

The estimation of these probabilities are strictly related to some key questions that constantly arises in plant breeding:

- **What is the risk of recommending a selection candidate for a target population of environments?**
- **What is the probability of a given selection candidate having good performance if recommended to a target population of environments? And for a specific environment?**
- **What is the probability of a given selection candidate having better performance than a cultivar check in the target population of environments? And in specific environments?**
- **How probable is it that a given selection candidate performs similarly across environments?**
- **What are the chances that a given selection candidate is more stable than a cultivar check in the target population of environments?**
- **What is the probability that a given selection candidate having a superior and invariable performance across environments?**

More details about the usage of `prob_sup`, as well as the other function of the `ProbBreed` package can be found at https://saulo-chaves.github.io/ProbBreed_site/.

Value

The function returns an object of class `probsup`, which contains two lists, one with the `across`-environments probabilities, and another with the `within`-environments probabilities. If an entry-mean model was used in `ProbBreed::bayes_met`, only the `across` list will be available.

The `across` list has the following elements:

- `g_hpd`: Highest posterior density (HPD) of the posterior genotypic main effects.
- `perfo`: the probabilities of superior performance.
- `pair_perfo`: the pairwise probabilities of superior performance.
- `stabi`: a list with the probabilities of superior stability. It contains the data frames `gl`, `gm` (when `reg` is not `NULL`) and `gt` (when `year` is not `NULL`). Unavailable if an entry-mean model was used in `bayes_met`.
- `pair_stabi`: a list with the pairwise probabilities of superior stability. It contains the data frames `gl`, `gm` (when `reg` is not `NULL`) and `gt` (when `year` is not `NULL`). Unavailable if an entry-mean model was used in `bayes_met`.
- `joint_prob`: the joint probabilities of superior performance and stability. Unavailable if an entry-mean model was used in `bayes_met`.

The `within` list has the following elements:

- `perfo`: a list of data frames containing the probabilities of superior performance within locations (`gl`), regions (`gm`) and years (`gt`).
- `pair_perfo`: lists with the pairwise probabilities of superior performance within locations (`gl`), regions (`gm`) and years (`gt`).

References

Dias, K. O. G, Santos, J. P. R., Krause, M. D., Piepho, H. -P., Guimarães, L. J. M., Pastina, M. M., and Garcia, A. A. F. (2022). Leveraging probability concepts for cultivar recommendation in multi-environment trials. *Theoretical and Applied Genetics*, 133(2):443-455. doi:10.1007/s00122-02204041y

Shukla, G. K. (1972) Some statistical aspects of partioning genotype environmental componentes of variability. *Heredity*, 29:237-245. doi:10.1038/hdy.1972.87

See Also

plot.probsup

Examples

```
mod = bayes_met(data = maize,
                gen = "Hybrid",
                loc = "Location",
                repl = c("Rep","Block"),
                trait = "GY",
                reg = "Region",
                year = NULL,
                res.het = TRUE,
                iter = 2000, cores = 2, chain = 4)

outs = extr_outs(model = mod,
                 probs = c(0.05, 0.95),
                 verbose = TRUE)

results = prob_sup(extr = outs,
                   int = .2,
                   increase = TRUE,
                   save.df = FALSE,
                   verbose = FALSE)
```

---

soy *Soybean real dataset*

Description

This dataset belongs to the USDA Northern Region Uniform Soybean Tests, and it is a subset of the data used by Krause et al. (2023). It contains the empirical best linear unbiased estimates of genotypic means of the seed yield from 39 experimental genotypes evaluated in 14 locations. The original data, available at the package `SoyURT`, has 4,257 experimental genotypes evaluated at 63 locations and 31 years resulting in 591 location-year combinations (environments) with 39,006 yield values.

Usage

```
soy
```

Format

`soy`**:**

A data frame with 823 rows and 3 columns:

Loc 14 locations

Gen 39 experimental genotypes

Y 435 EBLUEs (phenotypes)

Source

Krause, M. D., Dias, K. O. G., Singh A. K., Beavis W. D. (2023). Using soybean historical field trial data to study genotype by environment variation and identify mega-environments with the integration of genetic and non-genetic factors. *Agronomy Journal*, 117(1):170023. doi:10.1002/agj2.70023

`soy_pat` *Soybean Pan-African Trials data set*

Description

This data set belongs to the Soybean Pan-African Trials (PAT). This subset has the best linear unbiased estimates of grain yield (GY), plant height (PH) and number of days to maturity (NDM) of 65 soybean genotypes evaluated over 19 environments. The complete data set is available at Araújo et al. (2025) (check references). It contains the empirical best linear unbiased estimates of grain yield (GY), plant height (PH) and number of days to maturity (NDM) from 65 experimental genotypes evaluated in 19 locations.

Usage

```
soy_pat
```

Format

`soy_pat`:

A data frame with 540 rows and 5 columns:

Env 19 environments

Gen 65 experimental genotypes

Plant_Height 395 BLUEs - Plant height measurements

Grain_Yield 525 BLUEs - Grain yield measurements

Days_to_Maturity 312 BLUEs - Number of days to maturity

Source

Araújo, M. S., Chaves, S., Ferreira, G. N. C., Chigeza, G., Leles, E. P., Santos, M. F. S., Diers, B. W., Goldsmith, P., and Pinheiro, J. B. (2025). High-resolution soybean trial data supporting the expansion of agriculture in Africa. *Scientific Data*, 12:1908. doi:10.1038/s41597025061903

Index

∗ **datasets**
maize, 10
soy, 21
soy_pat, 22

bayes_met, 2, *9*, *18*
bayes_met(), *9*
bpsi, 7, *12*, *16*

extr_outs, 9, *14*, *17, 18*

ggplot2::geom_histogram, *13*
ggplot2::ggplot, *10*

maize, 10

plot.bpsi, 11
plot.bpsi(), *8*
plot.extr, *10*, 13
plot.probsup, 14, *21*
print.bpsi, 16
print.extr, 17
print.probsup, 17
prob_sup, *7*, *15*, *17*, 18

rstan::check_hmc_diagnostics, *10*
rstan::sampling, *6*
rstan::stan, *6*
rstan::stan_diag, *10*
rstan::stanfit, *6*

soy, 21
soy_pat, 22
stan, *4*