Package: GSbench (via r-universe)

July 1, 2026

Title Benchmarking Genomic Selection and Machine-Learning Prediction Models

Version 0.1.0

Description A unified interface to fit, cross-validate and benchmark genomic prediction models from SNP marker data. It implements genomic best linear unbiased prediction (GBLUP) and ridge-regression BLUP in base R, and offers a common interface to machine-learning predictors (elastic net, random forest and gradient boosting) through optional packages, together with a stacked ensemble. Cross-validation uses breeding-relevant schemes and reports prediction accuracy honestly, so models can be compared fairly. The genomic relationship matrix follows VanRaden (2008) <doi:10.3168/jds.2007-0980>; the mixed-model solver follows Endelman (2011) <doi:10.3835/plantgenome2011.08.0024>; the genomic-selection framework follows Meuwissen, Hayes and Goddard (2001) <doi:10.1093/genetics/157.4.1819>.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports graphics, stats, withr

Suggests rrBLUP, glmnet, ranger, xgboost, testthat (>= 3.0.0), knitr, rmarkdown, spelling

VignetteBuilder knitr

URL https://github.com/mqfarooqi1/GSbench

BugReports https://github.com/mqfarooqi1/GSbench/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Muhammad Farooqi [aut, cre]

Maintainer Muhammad Farooqi <mqfarooqi@gmail.com>

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

Date/Publication 2026-06-30 11:40:02 UTC

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

RemoteRef HEAD

RemoteSha 1e0b0c8356f73ccc5a7dbcc22b87d22ed5608e39

Contents

available_models . . . 2
gblup . . . 3
Gmatrix . . . 4
gs_benchmark . . . 4
gs_cv . . . 5
gs_ensemble . . . 7
gs_fit . . . 8
impute_markers . . . 8
plot.gs_cv . . . 9
predict.gblup . . . 10
predict.gs_ensemble . . . 10
qc_markers . . . 11
simulate_population . . . 12
summary.gs_cv . . . 12

Index **14**

`available_models` *Models available in this session*

Description

Returns the genomic-prediction models GSbench can currently run. `"gblup"` is always available; the machine-learning models require their (suggested) package to be installed.

Usage

```
available_models()
```

Value

A character vector of usable model names.

Examples

```
available_models()
```

gblup *Fit GBLUP by REML*

Description

Fits the genomic best linear unbiased prediction model `y = X b + g + e`, with `g ~ N(0, G s2u)` and `e ~ N(0, I s2e)`, estimating the variance components by restricted maximum likelihood (REML). The solver uses the spectral / eigendecomposition approach of Endelman (2011) (the same method as `rrBLUP::mixed.solve`): the REML log-likelihood is profiled to a one-dimensional optimisation over the variance ratio `lambda = s2e / s2u`.

Usage

```
gblup(y, geno = NULL, K = NULL, X = NULL)
```

Arguments

`y`	Numeric phenotype vector (length n), no missing values.
`geno`	Optional marker matrix (n x m, coded 0/1/2). Used to build `K` and to derive marker effects for out-of-sample prediction.
`K`	Optional n x n genomic relationship matrix. If `NULL`, built from `geno`. If you pass `K` directly (no `geno`), out-of-sample prediction is not available.
`X`	Optional fixed-effects design matrix (n x p). Defaults to an intercept.

Details

When `geno` is supplied, the equivalent ridge-regression marker effects are also returned, so the fit can predict breeding values for *new* genotypes via `predict.gblup()`.

Value

An object of class `gblup`: a list with `Vu`, `Ve`, `h2`, `beta` (fixed effects), `gebv` (length-n GEBVs), `lambda`, `K`, and — when `geno` was given — `marker_effects`, `marker_means` (2p) and `intercept`.

References

Endelman, J. B. (2011) "Ridge regression and other kernels for genomic selection with R package rrBLUP." The Plant Genome 4, 250-255. doi:10.3835/plantgenome2011.08.0024

Examples

```
sim <- simulate_population(n = 120, m = 500, seed = 1)
fit <- gblup(sim$pheno, sim$geno)
fit$h2
cor(fit$gebv, sim$bv)
```

Gmatrix *Genomic relationship matrix (VanRaden)*

Description

Builds the additive genomic relationship matrix from allele dosages using VanRaden's first method: with markers centred by twice the allele frequency, `G = W W' / (2 * sum(p (1 - p)))`. This is the standard additive GRM used in GBLUP.

Usage

```
Gmatrix(geno, min_maf = 0)
```

Arguments

`geno`	A numeric marker matrix (individuals x markers), coded 0/1/2, with no missing values (run `qc_markers()` or `impute_markers()` first).
`min_maf`	Markers with minor allele frequency below this are dropped before building the matrix. Default 0 (keep all supplied markers).

Value

An `n` x `n` symmetric genomic relationship matrix with the row/column names taken from `rownames(geno)`.

References

VanRaden, P. M. (2008) "Efficient methods to compute genomic predictions." Journal of Dairy Science 91, 4414-4423. doi:10.3168/jds.20070980

Examples

```
sim <- simulate_population(n = 80, m = 400, seed = 1)
G <- Gmatrix(sim$geno)
dim(G)
```

gs_benchmark *Benchmark all available genomic prediction models*

Description

Convenience wrapper around `gs_cv()` that evaluates every model available in the session (including the stacked `"ensemble"`) under one cross-validation, so they can be compared on an equal footing. Returns a `gs_cv` object; use `plot()` to draw the comparison.

Usage

```
gs_benchmark(
  y,
  geno,
  models = available_models(),
  k = 5,
  reps = 1,
  scheme = c("kfold", "leave_group_out"),
  groups = NULL,
  seed = NULL,
  ...
)
```

Arguments

y	Numeric phenotype vector (length n), no missing values.
geno	Marker matrix (n x m, 0/1/2, no missing values).
models	Models to evaluate; defaults to all available (see available_models()). Unavailable models are dropped with a warning.
k	Number of folds for "kfold". Default 5.
reps	Number of repeats for "kfold". Default 1.
scheme	"kfold" or "leave_group_out".
groups	Grouping vector (length n) for "leave_group_out".
seed	Optional seed (applied via withr::with_seed()).
...	Passed to gs_fit() (model hyperparameters).

Value

A gs_cv object (see gs_cv()).

Examples

```
sim <- simulate_population(n = 120, m = 400, seed = 1)
bench <- gs_benchmark(sim$pheno, sim$geno, models = "gblup", k = 5, seed = 1)
bench$accuracy
```

---

gs_cv *Cross-validate genomic prediction models*

Description

Runs a breeding-relevant cross-validation and reports predictive ability (the correlation between predictions and observed phenotypes on held-out individuals) for each model. Two schemes are supported: "kfold" (random k-fold, i.e. predicting untested lines, repeated reps times) and "leave_group_out" (hold out one family/environment at a time, via groups).

Usage

```
gs_cv(
  y,
  geno,
  models = available_models(),
  k = 5,
  reps = 1,
  scheme = c("kfold", "leave_group_out"),
  groups = NULL,
  seed = NULL,
  ...
)
```

Arguments

y	Numeric phenotype vector (length n), no missing values.
geno	Marker matrix (n x m, 0/1/2, no missing values).
models	Models to evaluate; defaults to all available (see available_models()). Unavailable models are dropped with a warning.
k	Number of folds for "kfold". Default 5.
reps	Number of repeats for "kfold". Default 1.
scheme	"kfold" or "leave_group_out".
groups	Grouping vector (length n) for "leave_group_out".
seed	Optional seed (applied via withr::with_seed()).
...	Passed to gs_fit() (model hyperparameters).

Details

Fitting is wrapped so that a model which errors on a fold records NA for that fold rather than aborting the whole run.

Value

An object of class gs_cv: a list with accuracy (a data frame of model, mean, sd, n_folds), per_fold (the raw fold results), and the call settings.

Examples

```
sim <- simulate_population(n = 120, m = 400, seed = 1)
cv <- gs_cv(sim$pheno, sim$geno, models = "gblup", k = 5, seed = 1)
cv
```

`gs_ensemble` *Stacked super-learner ensemble of genomic prediction models*

Description

Combines several base models into one predictor by *stacking*: each base model's out-of-fold cross-validated predictions are used to learn a set of non-negative weights (constrained to sum to one), and the final prediction is that weighted average of the base models refit on all the data. This is the Breiman / van der Laan stacked-regression (super-learner) idea applied to genomic selection; in practice it tends to match or beat the best single model without having to know in advance which that is.

Usage

```
gs_ensemble(y, geno, base_models = NULL, inner_k = 5, seed = NULL, ...)
```

Arguments

`y`	Numeric phenotype vector (length n), no missing values.
`geno`	Marker matrix (n x m, 0/1/2, no missing values).
`base_models`	Character vector of base model names. Defaults to every available model except the ensemble itself.
`inner_k`	Folds for the inner stacking cross-validation. Default 5.
`seed`	Optional seed for the inner folds (via `withr::with_seed()`).
`...`	Passed to `gs_fit()` for the base models.

Value

An object of class `gs_ensemble` (and `gs_model`): a list with `base_names`, `weights` (named, summing to 1), the refit `base_fits`, and the out-of-fold prediction matrix `oof`.

References

van der Laan, M. J., Polley, E. C. and Hubbard, A. E. (2007) "Super Learner." Statistical Applications in Genetics and Molecular Biology 6, Article 25. doi:10.2202/15446115.1309

Examples

```
sim <- simulate_population(n = 100, m = 300, seed = 1)
ens <- gs_ensemble(sim$pheno, sim$geno, base_models = "gblup", seed = 1)
ens$weights
```

gs_fit *Fit a genomic prediction model*

Description

One interface for several model families: `"gblup"` (always available), `"elastic_net"` (**glmnet**), `"random_forest"` (**ranger**), `"xgboost"` (**xgboost**), and `"ensemble"` (a stacked super-learner; see `gs_ensemble()`). The returned object has a predict() method that takes a new marker matrix.

Usage

```
gs_fit(y, geno, model = "gblup", ...)
```

Arguments

`y`	Numeric phenotype vector (length n), no missing values.
`geno`	Marker matrix (n x m, coded 0/1/2, no missing values).
`model`	Model name; see `available_models()`.
`...`	Model-specific hyperparameters (e.g. `alpha` for elastic net, `num.trees` for random forest, `nrounds`/`eta`/`max_depth` for xgboost, `base_models` for the ensemble).

Value

An object of class `gs_model` wrapping the fitted model.

Examples

```
sim <- simulate_population(n = 120, m = 400, seed = 1)
fit <- gs_fit(sim$pheno, sim$geno, model = "gblup")
head(predict(fit, sim$geno))
```

impute_markers *Impute missing marker genotypes*

Description

Replaces missing values in each marker (column) with that marker's mean dosage (i.e. twice the estimated allele frequency). Simple and fast; for production work a model-based imputation upstream is preferable.

Usage

```
impute_markers(geno)
```

Arguments

geno	A numeric marker matrix (individuals x markers), 0/1/2, possibly with NAs.

Value

The matrix with NAs filled by column means. Columns that are entirely missing are filled with 0.

Examples

```
g <- matrix(c(0, 1, NA, 2, 2, 0), nrow = 3)
impute_markers(g)
```

---

plot.gs_cv *Plot a model comparison*

Description

Bar chart of mean predictive ability per model, with +/- 1 SD whiskers across folds.

Usage

```
## S3 method for class 'gs_cv'
plot(x, ...)
```

Arguments

x	A gs_cv object.
...	Passed to graphics::barplot().

Value

x, invisibly.

Examples

```
sim <- simulate_population(n = 120, m = 400, seed = 1)

plot(gs_cv(sim$pheno, sim$geno, models = "gblup", k = 5, seed = 1))
```

predict.gblup *Predict breeding values for new genotypes from a GBLUP fit*

Description

Predict breeding values for new genotypes from a GBLUP fit

Usage

```
## S3 method for class 'gblup'
predict(object, newgeno, ...)
```

Arguments

object	A gblup fit created with geno supplied.
newgeno	Marker matrix for new individuals (markers must match the training markers, in the same order).
...	Ignored.

Value

A numeric vector of predicted values (intercept + marker effects), one per row of newgeno.

Examples

```
sim <- simulate_population(n = 150, m = 400, seed = 1)
fit <- gblup(sim$pheno[1:120], sim$geno[1:120, ])
predict(fit, sim$geno[121:150, ])
```

predict.gs_ensemble *Predict from a fitted genomic prediction model*

Description

Predict from a fitted genomic prediction model

Usage

```
## S3 method for class 'gs_ensemble'
predict(object, newgeno, ...)

## S3 method for class 'gs_model'
predict(object, newgeno, ...)
```

Arguments

object	A gs_model from gs_fit().
newgeno	Marker matrix for the individuals to predict (same markers as training).
...	Ignored.

Value

A numeric vector of predictions, one per row of newgeno.

Examples

```
sim <- simulate_population(n = 120, m = 400, seed = 1)
fit <- gs_fit(sim$pheno[1:90], sim$geno[1:90, ], model = "gblup")
predict(fit, sim$geno[91:120, ])
```

---

qc_markers *Quality-control filter for marker data*

Description

Drops markers failing a call-rate or minor-allele-frequency threshold, and (optionally) monomorphic markers, then imputes any remaining missing values.

Usage

```
qc_markers(geno, maf = 0.05, max_missing = 0.1, impute = TRUE)
```

Arguments

geno	A numeric marker matrix (individuals x markers), coded 0/1/2.
maf	Minimum minor allele frequency to keep a marker. Default 0.05.
max_missing	Maximum fraction of missing calls to keep a marker. Default 0.1.
impute	Whether to mean-impute remaining missing values. Default TRUE.

Value

A list with geno (the filtered, optionally imputed matrix) and removed (a named integer vector counting markers dropped by each rule).

Examples

```
sim <- simulate_population(n = 50, m = 200, seed = 1)
qc <- qc_markers(sim$geno)
dim(qc$geno)
qc$removed
```

simulate_population *Simulate a genomic prediction dataset*

Description

Generates a small marker matrix and a phenotype with a known narrow-sense heritability, for examples, tests and demonstrations. Markers are coded as allele dosages (0, 1, 2). A random subset of markers are QTL with additive effects; the phenotype is the resulting breeding value plus normal noise scaled to the target heritability.

Usage

```
simulate_population(n = 200, m = 1000, n_qtl = 50, h2 = 0.5, seed = NULL)
```

Arguments

n	Number of individuals. Default 200.
m	Number of markers. Default 1000.
n_qtl	Number of markers with a non-zero (QTL) effect. Default 50.
h2	Target narrow-sense heritability in (0, 1]. Default 0.5.
seed	Optional integer seed for reproducibility. Applied with withr::with_seed(), which scopes the seed locally and leaves the caller's random-number state unchanged.

Value

A list with geno (an n x m 0/1/2 matrix, row/column names set), pheno (length-n numeric), bv (true breeding values), qtl (indices of the causal markers) and h2 (the realised heritability).

Examples

```
sim <- simulate_population(n = 100, m = 300, seed = 1)
dim(sim$geno)
length(sim$pheno)
```

summary.gs_cv *Summary of a cross-validation / benchmark*

Description

Summary of a cross-validation / benchmark

Usage

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

Arguments

`object`	A `gs_cv` object.
`...`	Ignored.

Value

The accuracy data frame, invisibly.

Examples

```
sim <- simulate_population(n = 100, m = 300, seed = 1)
summary(gs_cv(sim$pheno, sim$geno, models = "gblup", seed = 1))
```

Index

available_models, 2
available_models(), *5*, *6*, *8*

gblup, 3
Gmatrix, 4
graphics::barplot(), *9*
gs_benchmark, 4
gs_cv, 5
gs_cv(), *4*, *5*
gs_ensemble, 7
gs_ensemble(), *8*
gs_fit, 8
gs_fit(), *5–7*, *11*

impute_markers, 8
impute_markers(), *4*

plot(), *4*
plot.gs_cv, 9
predict(), *8*
predict.gblup, 10
predict.gblup(), *3*
predict.gs_ensemble, 10
predict.gs_model *(*predict.gs_ensemble*)*, 10

qc_markers, 11
qc_markers(), *4*

simulate_population, 12
summary.gs_cv, 12

withr::with_seed(), *5–7*, *12*