Package: PopVar (via r-universe)

October 21, 2024

Title Genomic Breeding Tools: Genetic Variance Prediction and Cross-Validation

Version 1.3.2

Description The main attribute of 'PopVar' is the prediction of genetic variance in bi-parental populations, from which the package derives its name. 'PopVar' contains a set of functions that use phenotypic and genotypic data from a set of candidate parents to 1) predict the mean, genetic variance, and superior progeny value of all, or a defined set of pairwise bi-parental crosses, and 2) perform cross-validation to estimate genome-wide prediction accuracy of multiple statistical models. More details are available in Mohammadi, Tiede, and Smith (2015, <doi:10.2135/cropsci2015.01.0030>). A dataset 'think_barley.rda' is included for reference and examples.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/UMN-BarleyOatSilphium/PopVar

Depends R (>= 3.5.0)

Imports BGLR, qtl, rrBLUP, stats, utils, methods, parallel

Suggests knitr, rmarkdown

VignetteBuilder knitr

BugReports https://github.com/UMN-BarleyOatSilphium/PopVar/issues

Repository https://umn-barleyoatsilphium.r-universe.dev

RemoteUrl https://github.com/umn-barleyoatsilphium/popvar

RemoteRef HEAD

RemoteSha 598f1eda39106f8e44ab2f4eda4a1367acd9952b

Contents

internal 2
mppop.predict 3
pop.predict 6
pop.predict2 11
think_barley.rda 14
x.val 14

Index **18**

`internal` *Internal functions*

Description

Internal functions generally not to be called by the user.

Usage

```
par_position(crossing.table, par.entries)

par_name(crossing.mat, par.entries)

tails(GEBVs, tail.p)

maf_filt(G)

XValidate_nonInd(
  y.CV = NULL,
  G.CV = NULL,
  models.CV = NULL,
  frac.train.CV = NULL,
  nCV.iter.CV = NULL,
  burnIn.CV = NULL,
  nIter.CV = NULL
)

XValidate_Ind(
  y.CV = NULL,
  G.CV = NULL,
  models.CV = NULL,
  nFold.CV = NULL,
  nFold.CV.reps = NULL,
  burnIn.CV = NULL,
  nIter.CV = NULL
)
```

```
calc_marker_effects(
  M,
  y.df,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  nIter,
  burnIn
)
```

Arguments

`crossing.table`	The crossing table.
`par.entries`	The parent entries.
`crossing.mat`	The crossing matrix.
`GEBVs`	The genomic estimated breeding values.
`tail.p`	The proportion from the population to select.
`G`	The marker genotypes
`y.CV`	The phenotypes for cross-validation.
`G.CV`	The marker genotypes for cross-validation.
`models.CV`	The models for cross-validation.
`frac.train.CV`	The fraction of data to use as training data in cross-validation.
`nCV.iter.CV`	The number of iterations of cross-validation.
`burnIn.CV`	The burn-in number for cross-validation.
`nIter.CV`	The number of iterations for Bayesian models in cross-validation.
`nFold.CV`	The number of folds in k-fold cross-validation.
`nFold.CV.reps`	The number of replications of k-fold cross-validation.
`M`	The marker matrix.
`y.df`	The phenotype data.
`models`	The models to use.
`nIter`	The number of iterations.
`burnIn`	The burn-in rate.

`mppop.predict` *Predict genetic variance and genetic correlations in multi-parent populations using a deterministic equation.*

Description

Predicts the genotypic mean, genetic variance, and usefulness criterion (superior progeny mean) in a set of multi-parent populations using marker effects and a genetic map. If more than two traits are specified, the function will also return predictions of the genetic correlation in the population and the correlated response to selection.

Usage

```
mppop.predict(
  G.in,
  y.in,
  map.in,
  crossing.table,
  parents,
  n.parents = 4,
  tail.p = 0.1,
  self.gen = 10,
  DH = FALSE,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  n.core = 1,
  ...
)

mppop_predict2(
  M,
  y.in,
  marker.effects,
  map.in,
  crossing.table,
  parents,
  n.parents = 4,
  tail.p = 0.1,
  self.gen = 10,
  DH = FALSE,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  n.core = 1,
  ...
)
```

Arguments

`G.in`	See `G.in` in `pop.predict`.
`y.in`	See `y.in` in `pop.predict`.
`map.in`	See `map.in` in `pop.predict`.
`crossing.table`	A `data.frame` with 2 columns (for bi-parental crosses) or 4 columns (for four-way crosses), each of which contains the names of parents to use in a potential cross. Rows contain individual crosses. See Details.
`parents`	See `parents` in `pop.predict`.
`n.parents`	Integer number of parents per cross. May be 2 or 4. If `crossing.table` is passed, this argument is ignored.
`tail.p`	See `tail.p` in `pop.predict`.
`self.gen`	The number of selfing generations in the potential cross. Can be an integer or `Inf` for recombinant inbreds. Note: `self.gen` = 1 corresponds to an F2 population.

`DH` Indicator if doubled-haploids are to be induced after the number of selfing generations indicated by `self.gen`. For example, if `self.gen = 0` and `DH = TRUE`, then doubled-haploids are assumed to be induced using gametes from F1 plants.

`models` See `models` in `pop.predict`.

`n.core` Number of cores for parallelization. Parallelization is supported only on a Linux or Mac OS operating system; if working on a Windows system, the function is executed on a single core.

`...` Additional arguments to pass depending on the choice of `model`.

`M` A Matrix of marker genotypes of dimensions `nLine` x `nMarker`, coded as -1, 0, and 1.

`marker.effects` A data frame of marker effects. The first column should include the marker name and subsequent columns should include the marker effects. Supercedes `y.in` if passed.

Details

Predictions are based on the deterministic equations specified by Allier et al. (2019).

In the case of four-way crosses (i.e. 4 parents), the function assumes that the first two parents are mated, producing a F_1 offspring; then, the next two parents are mated, producing another F_1 offspring. The two F_1 offspring are then mated and inbreeding or doubled haploid induction (if specified) proceeds from there. For example, say cross i uses parents P1, P2, P3, and P4. P1 and P2 are first mated, producing O1; then, P3 and P4 are mated, producing O2; then, O1 and O2 are mated, producing a segregating family.

The `mppop.predict` function takes similarly formatted arguments as the `pop.predict` function in the `PopVar` package. For the sake of simplicity, we also include the `mppop_predict2` function, which takes arguments in a format more consistent with other genomewide prediction packages/functions.

If you select a `model` other than "rrBLUP", you must specify the following additional arguments:

- `nIter`: See `pop.predict`.
- `burnIn`: See `pop.predict`.

Value

A `data.frame` containing predictions of μ, V_G, and μ_{sp} for each trait for each potential multi-parent cross. When multiple traits are provided, the correlated responses and correlation between all pairs of traits is also returned.

References

Allier, A., L. Moreau, A. Charcosset, S. Teyssèdre, and C. Lehermeier, 2019 Usefulness Criterion and Post-selection Parental Contributions in Multi-parental Crosses: Application to Polygenic Trait Introgression. G3 (Bethesda) 9: 1469–1479. https://doi.org/https://doi.org/10.1534/g3.119.400129

Examples

```
# Load data
data("think_barley")

# Vector with 8 parents
parents <- sample(y.in_ex$Entry, 8)

# Create a crossing table with four parents per cross
cross_tab <- as.data.frame(t(combn(x = parents, m = 4)))
names(cross_tab) <- c("parent1", "parent2", "parent3", "parent4")

out <- mppop_predict2(M = G.in_ex_mat, y.in = y.in_ex, map.in = map.in_ex,
                      crossing.table = cross_tab, models = "rrBLUP")
```

`pop.predict` *A genome-wide procedure for predicting genetic variance and correlated response in bi-parental breeding populations*

Description

`pop.predict` uses phenotypic and genotypic data from a set of individuals known as a training population (TP) and a set of candidate parents, which may or may not be included in the TP, to predict the mean (μ), genetic variance (*V_G*), and superior progeny values (μ*_sp*) of the half-diallel, or a defined set of pairwise bi-parental crosses between parents. When multiple traits are provided `pop.predict` will also predict the correlated responses and correlation between all pairwise traits. See *Mohammadi, Tiede, and Smith (2015)* for further details.

```
NOTE - \code{pop.predict} writes and reads files to disk so it is highly recommended to set your wo
```

Usage

```
pop.predict(
  G.in = NULL,
  y.in = NULL,
  map.in = NULL,
  crossing.table = NULL,
  parents = NULL,
  tail.p = 0.1,
  nInd = 200,
  map.plot = FALSE,
  min.maf = 0.01,
```

```
  mkr.cutoff = 0.5,
  entry.cutoff = 0.5,
  remove.dups = TRUE,
  impute = "EM",
  nSim = 25,
  frac.train = 0.6,
  nCV.iter = 100,
  nFold = NULL,
  nFold.reps = 1,
  nIter = 12000,
  burnIn = 3000,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  return.raw = FALSE,
  saveAt = tempdir()
)
```

Arguments

`G.in` Matrix of genotypic data. First row contains marker names and the first column contains entry (taxa) names. Genotypes should be coded as follows:

- `1`: homozygous for minor allele
- `0`: heterozygous
- `-1`: homozygous for major allele
- `NA`: missing data
- Imputed genotypes can be passed, see `impute` below for details

TIP - Set header=`FALSE` within `read.table` or `read.csv` when importing a tab-delimited file containing data for `G.in`.

`y.in` Matrix of phenotypic data. First column contains entry (taxa) names found in `G.in`, regardless of whether the entry has a phenotype for any or all traits. Additional columns contain phenotypic data; column names should reflect the trait name(s). TIP - Set header=`TRUE` within `read.table` or `read.csv` when importing a tab-delimited file containing data for `y.in`.

`map.in` Matrix of genetic map data, three columns total. Column 1 contains marker names, column 2 contains chromosome number, and column 3 contains cM positions. TIP - Set header=`TRUE` within `read.table` or `read.csv` when importing a tab-delimited file contianing data for `map.in`.

`crossing.table` Optional `matrix` specifying which crosses are to be simulated, two columns total. Column 1 contains the first parent of the cross (Par1) and column 2 contains the second parent of the cross (Par2).

`parents` Optional `character vector`. If `parents="TP"` then only the entries (taxa) within the training population (i.e. are phenotyped for the trait) are considered as parents; all pairwise crosses will be simulated for these. User could otherwise provide a character vector of entry names; all pairwise crosses will be simulated for these.

`tail.p` Optional `numeric` indicating the percentile of the simulated progeny to be included into the calculation of μ_sp and correlated response. Default is `0.10`.

nInd	Optional integer indicating the number of progeny simulated per cross, per iteration, using sim.cross in R/qtl (*Broman et al., 2003*). Default is 200.
map.plot	Optional logical. If TRUE then a plot of the genetic map will be generated by plot.map. Default is FALSE.
min.maf	Optional numeric indicating a minimum minor allele frequency (MAF) when filtering G.in. Markers with an MAF < min.maf will be removed. Default is 0.01 to remove monomorphic markers. Set to 0 for no filtering.
mkr.cutoff	Optional numeric indicating the maximum missing data per marker when filtering G.in. Markers missing > mkr.cutoff data will be removed. Default is 0.50. Set to 1 for no filtering.
entry.cutoff	Optional numeric indicating the maximum missing genotypic data per entry allowed when filtering G.in. Entries missing > entry.cutoff marker data will be removed. Default is 0.50. Set to 1 for no filtering.
remove.dups	Optional logical. If TRUE duplicate entries in the genotype matrix, if present, will be removed. This step may be necessary for missing marker imputation (see impute below). Default is TRUE.
impute	Options include c("EM", "mean", "pass"). By default (i.e. "EM"), after filtering missing genotypic data will be imputed via the EM algorithm implemented in A.mat (*Endelman, 2011*; *Poland et al., 2012*). If "mean" missing genotypic data will be imputed via the 'marker mean' method, also implemented in A.mat. Enter "pass" if a pre-filtered and imputed genotype matrix is provided to G.in.
nSim	Optional integer indicating the number of iterations a population should be simulated for each pairwise cross. Returned values are reported as means of parameters estimated in each of nSim simulations. Default is 25.
frac.train	Optional numeric indicating the fraction of the TP that is used to estimate marker effects (i.e. the prediction set) under cross-validation (CV) method 1 (see Details in x.val). The remaining $(1 - frac.trait)$ of the TP will then comprise the prediction set.
nCV.iter	Optional integer indicating the number of times to iterate *CV method 1* (see Details in x.val). Default is 100.
nFold	Optional integer. If a number is provided, denoting the number of "folds", then CV will be conducted using *CV method 2* (see Details in x.val). Default is NULL, resulting in the default use of the *CV method 1*.
nFold.reps	Optional integer indicating the number of times *CV method 2* is repeated. The CV accuracy returned is the average *r* of each rep. Default is 1.
nIter, burnIn	Optional integer arguments used by BGLR (*de los Compos and Rodriguez, 2014*) when fitting Bayesian models to estimate marker effects. The defaults are 12000 and 3000, respectively. These values when conducting CV are fixed 1500 and 500, respectively, for computational efficiency.
models	Optional Character vector of the regression models to be used in CV and to estimate marker effects. Options include rrBLUP, BayesA, BayesB, BayesC, BL, BRR, one or more may be included at a time. CV will be conducted regardless of how many models are included. By default all models are tested.
return.raw	Optional logical. If TRUE then pop.predict will return the results of each simulation in addition to the summarized dataframe. Default is FALSE.

saveAt — When using models other than "rrBLUP" (i.e. Bayesian models), this is a path and prefix for saving temporary files the are produced by the BGLR function.

Details

pop.predict can be used to predict the mean (μ), genetic variance (*V_G*), superior progeny values ($\mu_s p$), as well as the predicted correlated response and correlations between all pairwise traits. The methodology and procedure to do so has been described in *Bernardo (2014)* and *Mohammadi, Tiede, and K.P. Smith (2015)*. Users familiar with genome-wide prediction, association mapping, and/or linkage mapping will be familiar with the required inputs of pop.predict. G.in includes all of the entries (taxa) in the TP as well as additional entries to be considered as parent candidates. Entries included in G.in that do have a phenotype for any or all traits in y.in are considered TP entries for those respective traits. G.in is filtered according to min.maf, mkr.cutoff, entry.cutoff, and remove.dups; remaining missing marker data is imputed using the EM algorithm (*Poland et al., 2012*) when possible, and the marker mean otherwise, both implemented in A.mat. For each trait, the TP (i.e. entries with phenotype) is used to:

1. Perform CV to select a regression model. NOTE - Using the model with the highest CV accuracy is expected to result in the most accurate marker effect estimates (*Bernardo, 2014*). This expectation, however, is yet to be empirically validated and the user is encouraged to investigate the various models in order to make an educated decision about which one to ultimately use.
2. Estimate marker effects using the model resulting in the highest CV accuracy

Models include ridge regression BLUP implemented in mixed.solve (*Endelman, 2011*) and BayesA, BayesB, BayesCπ, Bayesian lasso (BL), and Bayesian ridge regression (BRR) implemented in BGLR (*de los Compos and Rodriguez, 2014*). Information from the map.in is then used to simulate chromosomal recombination expected in a recombinant inbred line (i.e. *F-infinity*) (*Broman et al., 2003*) population (size=nInd). A function then converts the recombined chromosomal segments of the generic RIL population to the chromosomal segments of the population's respective parents and GEBVs of the simulated progeny are calculated. The simulation and conversion process is repeated *s* times, where *s* = nSim, to calculate dispersion statistics for μ and *V_G*; the remainder of the values in the predictions output are means of the s simulations. During each iteration the correlation (*r*) and correlated response of each pairwise combination of traits is also calculated and their mean across *n* simulations is returned. The correlated response of trait.B when predicting trait.A is the mean of trait.B for the ($\mu_s p$) of trait.A, and vice-versa; a correlated response for the bottom tail.p and upper $1 - tail.p$ is returned for each trait.

```
A dataset \code{\link{think_barley.rda}} is provided as an example of the proper formatting of input
```

Value

A list containing:

- predictions A list of dataframes containing predictions of (μ), (*V_G*), and (μ_*sp*). When multiple traits are provided the correlated responses and correlation between all pairwise traits is also included. More specifically, for a given trait pair the correlated response of the secondary trait with both the high and low superior progeny of the primary trait is returned since the favorable values cannot be known by PopVar.

- `preds.per.sim` If return.raw is `TRUE` then a `dataframe` containing the results of each simulation is returned. This is useful for calculating dispersion statistics for traits not provided in the standard `predictions` dataframe.
- `CVs` A `dataframe` of CV results for each trait/model combination specified.
- `models.chosen` A `matrix` listing the statistical model chosen for each trait.
- `markers.removed` A `vector` of markers removed during filtering for MAF and missing data.
- `entries.removed` A `vector` of entries removed during filtering for missing data and duplicate entries.

References

Bernardo, R. 2014. Genomewide Selection of Parental Inbreds: Classes of Loci and Virtual Biparental Pop

Broman, K. W., H. Wu, S. Sen and G.A. Churchill. 2003. R/qtl: QTL mapping in experimental crosses. Bioin

Endelman, J. B. 2011. Ridge regression and other kernels for genomic selection with R package rrBLUP. Pl

Gustavo de los Campos and Paulino Perez Rodriguez, (2014). BGLR: Bayesian Generalized Linear Regression

Mohammadi M., T. Tiede, and K.P. Smith. 2015. PopVar: A genome-wide procedure for predicting genetic var

Munoz-Amatriain, M., M. J. Moscou, P. R. Bhat, J. T. Svensson, J. Bartos, P. Suchankova, H. Simkova, T. R

Poland, J., J. Endelman, J. Dawson, J. Rutkoski, S. Wu, Y. Manes, S. Dreisigacker, J. Crossa, H. Sanches

Examples

```
## Not run:

# Load data
data("think_barley")

## The following examples only use the model 'rrBLUP' for the sake of testing. Functions
## BGLR package write temporary files to the disk.
##
## Ex. 1 - Predict a defined set of crosses
## This example uses CV method 1 (see Details of x.val() function)
ex1.out <- pop.predict(G.in = G.in_ex, y.in = y.in_ex,
   map.in = map.in_ex, crossing.table = cross.tab_ex,
   nSim=5, nCV.iter=2, models = "rrBLUP")
ex1.out$predictions  ## Predicted parameters
ex1.out$CVs          ## CV results

## Ex. 2 - Use only rrBLUP and Bayesian lasso (BL) models
ex3.out <- pop.predict(G.in = G.in_ex, y.in = y.in_ex,
   map.in = map.in_ex, crossing.table = cross.tab_ex,
   models = c("rrBLUP"), nSim=5, nCV.iter=10)

## Ex. 3 - Same as Ex. 3, but return all raw SNP and prediction data for each simulated population
ex4.out <- pop.predict(G.in = G.in_ex, y.in = y.in_ex,
```

```
    map.in = map.in_ex, crossing.table = cross.tab_ex,
    models = c("rrBLUP"), nSim=5, nCV.iter=2, return.raw = TRUE)

## End(Not run)
```

pop.predict2 *Predict genetic variance and genetic correlations in bi-parental populations using a deterministic model*

Description

Generates predictions of the genetic variance and genetic correlation in bi-parental populations using a set of deterministic equations instead of simulations.

Usage

```
pop.predict2(
  G.in,
  y.in,
  map.in,
  crossing.table,
  parents,
  tail.p = 0.1,
  self.gen = Inf,
  DH = FALSE,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  ...
)

pop_predict2(
  M,
  y.in,
  marker.effects,
  map.in,
  crossing.table,
  parents,
  tail.p = 0.1,
  self.gen = Inf,
  DH = FALSE,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  ...
)
```

Arguments

G.in	See G.in in pop.predict.

y.in	See y.in in pop.predict.
map.in	See map.in in pop.predict.
crossing.table	See crossing.table in pop.predict.
parents	See parents in pop.predict.
tail.p	See tail.p in pop.predict.
self.gen	The number of selfing generations in the potential cross. Can be an integer or Inf for recombinant inbreds. Note: self.gen = 1 corresponds to an F2 population.
DH	Indicator if doubled-haploids are to be induced after the number of selfing generations indicated by self.gen. For example, if self.gen = 0 and DH = TRUE, then doubled-haploids are assumed to be induced using gametes from F1 plants.
models	See models in pop.predict.
...	Additional arguments to pass depending on the choice of model.
M	A Matrix of marker genotypes of dimensions nLine x nMarker, coded as -1, 0, and 1.
marker.effects	A data frame of marker effects. The first column should include the marker name and subsequent columns should include the marker effects. Supercedes y.in if passed.

Details

Predictions are based on the deterministic equations specified by Zhong and Jannink (2007), Allier et al. (2019), and Neyhart et al. (2019).

If you select a model other than "rrBLUP", you must specify the following additional arguments:

- nIter: See pop.predict.
- burnIn: See pop.predict.

Value

A data.frame containing predictions of μ, V_G, and μ_{sp} for each trait for each potential bi-parental cross. When multiple traits are provided, the correlated responses and correlation between all pairs of traits is also returned.

Functions

- pop_predict2():

References

Zhong, S., and J.-L. Jannink, 2007 Using quantitative trait loci results to discriminate among crosses on the basis of their progeny mean and variance. Genetics 177: 567–576. https://doi.org/10.1534/genetics.107.075358

Allier, A., L. Moreau, A. Charcosset, S. Teyssèdre, and C. Lehermeier, 2019 Usefulness Criterion and Post-selection Parental Contributions in Multi-parental Crosses: Application to Polygenic Trait Introgression. G3 9: 1469–1479. doi: 10.1534/g3.119.400129

Neyhart, J.L., A.J. Lorenz, and K.P. Smith, 2019 Multi-trait Improvement by Predicting Genetic Correlations in Breeding Crosses. G3 9: 3153-3165. doi: 10.1534/g3.119.400406

Examples

```
# Load data
data("think_barley")

# Use example data to make predictions
out <- pop.predict2(G.in = G.in_ex_imputed, y.in = y.in_ex, map.in = map.in_ex,
                    crossing.table = cross.tab_ex)

# Provide a vector of parents to predict all possible crosses (some parents
# have missing phenotypic data)
out <- pop.predict2(G.in = G.in_ex_imputed, y.in = y.in_ex, map.in = map.in_ex,
                    parents = y.in_ex$Entry[1:5])

# Make predictions for 5 crosses with various levels of inbreeding
out_list <- lapply(X = 1:10, FUN = function(self.gen) {
  out <- pop.predict2(G.in = G.in_ex_imputed, y.in = y.in_ex, map.in = map.in_ex,
                      crossing.table = cross.tab_ex[1:5,], self.gen = self.gen)
  out$self.gen <- self.gen
  out })

# Plot predictions of grain yield genetic variance over levels of inbreeding
dat <- do.call("rbind", lapply(out_list, subset, trait == "Yield"))
plot(pred_varG ~ self.gen, data = dat, type = "b",
     subset = parent1 == parent1[1] & parent2 == parent2[1])
```

```
# Load data
data("think_barley")

# Use example data to make predictions
out <- pop_predict2(M = G.in_ex_mat, y.in = y.in_ex, map.in = map.in_ex,
                    crossing.table = cross.tab_ex)

# Provide a vector of parents to predict all possible crosses (some parents
# have missing phenotypic data)
out <- pop_predict2(M = G.in_ex_mat, y.in = y.in_ex, map.in = map.in_ex,
                    parents = y.in_ex$Entry[1:10])
```

think_barley.rda — *An example barley dataset*

Description

A sample dataset, previously described in *Sallam et al. (2014)* is provided as an example of the proper formatting of input files and also for users to become familiar with `PopVar`; the `think_barley` dataset is useful in demonstrating both `pop.predict` and `x.val`. Note that a number of entries are missing data for one or both traits, which is representative of a real breeding scenario where phenotypic data may not be available for all parent candidates.

Format

The names of the example files are:

G.in_ex A set of 245 barley lines genotyped with 742 SNP markers

G.in_ex_mat A n x p matrix of n = 245 barley lines genotyped with p = 742 SNP markers

G.in_ex_imputed A n x p matrix of n = 245 barley lines and p = 742 *imputed* SNP marker genotypes

y.in_ex Phenotypes of four traits for a portion of the 245 barley lines, Fusarium head blight (FHB), deoxynivalenol (DON) in ppm, grain yield in bushels/acre, and plant height in cm.

map.in_ex Genetic map (i.e. chromosome assignment and genetic distance (cM) between markers) of the 742 SNP markers based on *Munoz-Amatriain et al., 2011*

cross.tab_ex A table of user-defined crosses

References

Sallam, A.H., J.B. Endelman, J-L. Jannink, and K.P. Smith. 2015. Assessing Genomic Selection Prediction Accuracy in a Dynamic Barley Breeding Population. Plant Gen. 8(1)

x.val — *Estimate genome-wide prediction accuracy using cross-validation*

Description

`x.val` performs cross-validation (CV) to estimate the accuracy of genome-wide prediction (otherwise known as genomic selection) for a specific training population (TP), i.e. a set of individuals for which phenotypic and genotypic data is available. Cross-validation can be conducted via one of two methods within `x.val`, see `Details` for more information.

```
NOTE - \code{x.val}, specifically \code{\link[BGLR]{BGLR}} writes and reads files to disk so it is
```

Usage

```
x.val(
  G.in = NULL,
  y.in = NULL,
  min.maf = 0.01,
  mkr.cutoff = 0.5,
  entry.cutoff = 0.5,
  remove.dups = TRUE,
  impute = "EM",
  frac.train = 0.6,
  nCV.iter = 100,
  nFold = NULL,
  nFold.reps = 1,
  return.estimates = FALSE,
  CV.burnIn = 750,
  CV.nIter = 1500,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR"),
  saveAt = tempdir()
)
```

Arguments

`G.in` — `Matrix` of genotypic data. First row contains marker names and the first column contains entry (taxa) names. Genotypes should be coded as follows:

- `1`: homozygous for minor allele
- `0`: heterozygous
- `-1`: homozygous for major allele
- `NA`: missing data
- Imputed genotypes can be passed, see `impute` below for details

TIP - Set header=`FALSE` within `read.table` or `read.csv` when importing a tab-delimited file containing data for `G.in`.

`y.in` — `Matrix` of phenotypic data. First column contains entry (taxa) names found in `G.in`, regardless of whether the entry has a phenotype for any or all traits. Additional columns contain phenotypic data; column names should reflect the trait name(s). TIP - Set header=`TRUE` within `read.table` or `read.csv` when importing a tab-delimited file containing dat

`min.maf` — Optional `numeric` indicating a minimum minor allele frequency (MAF) when filtering `G.in`. Markers with an MAF < `min.maf` will be removed. Default is `0.01` to remove monomorphic markers. Set to `0` for no filtering.

`mkr.cutoff` — Optional `numeric` indicating the maximum missing data per marker when filtering `G.in`. Markers missing > `mkr.cutoff` data will be removed. Default is `0.50`. Set to 1 for no filtering.

`entry.cutoff` — Optional `numeric` indicating the maximum missing genotypic data per entry allowed when filtering `G.in`. Entries missing > `entry.cutoff` marker data will be removed. Default is `0.50`. Set to 1 for no filtering.

remove.dups
: Optional logical. If TRUE duplicate entries in the genotype matrix, if present, will be removed. This step may be necessary for missing marker imputation (see impute). Default is TRUE.

impute
: Options include c("EM", "mean", "pass"). By default (i.e. "EM"), after filtering missing genotypic data will be imputed via the EM algorithm implemented in A.mat (*Endelman, 2011*; *Poland et al., 2012*). If "mean" missing genotypic data will be imputed via the 'marker mean' method, also implemented in A.mat. Enter "pass" if a pre-filtered and imputed genotype matrix is provided to G.in.

frac.train
: Optional numeric indicating the fraction of the TP that is used to estimate marker effects (i.e. the prediction set) under cross-validation (CV) method 1 (see Details). The remaining $(1 - frac.trait)$ of the TP will then comprise the prediction set.

nCV.iter
: Optional integer indicating the number of times to iterate *CV method 1* described in Details. Default is 100.

nFold
: Optional integer. If a number is provided, denoting the number of "folds", then CV will be conducted using *CV method 2* (see Details). Default is NULL, resulting in the default use of the *CV method 1*.

nFold.reps
: Optional integer indicating the number of times *CV method 2* is repeated. The CV accuracy returned is the average *r* of each rep. Default is 1.

return.estimates
: Optional logical. If TRUE additional items including the marker effect and beta estimates from the selected prediction model (i.e. highest CV accuracy) will be returned.

CV.burnIn
: Optional integer argument used by BGLR when fitting Bayesian models. Default is 750.

CV.nIter
: Optional integer argument used by BGLR (*de los Compos and Rodriguez, 2014*) when fitting Bayesian models. Default is 1500.

models
: Optional character vector of the regression models to be used in CV and to estimate marker effects. Options include rrBLUP, BayesA, BayesB, BayesC, BL, BRR, one or more may be included at a time. By default all models are tested.

saveAt
: When using models other than "rrBLUP" (i.e. Bayesian models), this is a path and prefix for saving temporary files the are produced by the BGLR function.

Details

Two CV methods are available within PopVar:

- CV method 1: During each iteration a training (i.e. model training) set will be **randomly sampled** from the TP of size $N * (frac.train)$, where *N* is the size of the TP, and the remainder of the TP is assigned to the validation set. The accuracies of individual models are expressed as average Pearson's correlation coefficient (*r*) between the genome estimated breeding value (GEBV) and observed phenotypic values in the validation set across all nCV.iter iterations. Due to its amendibility to various TP sizes, *CV method 1* is the default CV method in pop.predict.

- `CV method 2`: `nFold` **independent** validation sets are sampled from the TP and predicted by the remainder. For example, if $nFold = 10$ the TP will be split into 10 equal sets, each containing $1/10$-th of the TP, which will be predicted by the remaining $9/10$-ths of the TP. The accuracies of individual models are expressed as the average (*r*) between the GEBV and observed phenotypic values in the validation set across all `nFold` folds. The process can be repeated `nFold.reps` times with `nFold` new independent sets being sampled each replication, in which case the reported prediction accuracies are averages across all folds and replications.

Value

A list containing:

- `CVs` A `dataframe` of CV results for each trait/model combination specified
- If `return.estimates` is `TRUE` the additional items will be returned:
  - `models.used` A `list` of the models chosen to estimate marker effects for each trait
  - `mkr.effects` A `vector` of marker effect estimates for each trait generated by the respective prediction model used
  - `betas` A `list` of beta values for each trait generated by the respective prediction model used

Examples

```
## The following examples only use the model 'rrBLUP' for the sake of testing. Functions
## BGLR package write temporary files to the disk.

## CV using method 1 with 25 iterations
CV.mthd1 <- x.val(G.in = G.in_ex, y.in = y.in_ex, nCV.iter = 25, models = "rrBLUP")
CV.mthd1$CVs

## CV using method 2 with 5 folds and 3 replications
x.val(G.in = G.in_ex, y.in = y.in_ex, nFold = 5, nFold.reps = 3, models = "rrBLUP")
```

Index

A.mat, *8, 9*, *16*

BGLR, *8, 9*, *16*

calc_marker_effects *(*internal*)*, 2
cross.tab_ex *(*think_barley.rda*)*, 14

G.in_ex *(*think_barley.rda*)*, 14
G.in_ex_imputed *(*think_barley.rda*)*, 14
G.in_ex_mat *(*think_barley.rda*)*, 14

internal, 2

maf_filt *(*internal*)*, 2
map.in_ex *(*think_barley.rda*)*, 14
mixed.solve, *9*
mppop.predict, 3
mppop_predict2 *(*mppop.predict*)*, 3

par_name *(*internal*)*, 2
par_position *(*internal*)*, 2
plot.map, *8*
pop.predict, *4, 5*, 6, *11, 12*, *14*, *16*
pop.predict2, 11
pop_predict2 *(*pop.predict2*)*, 11

read.csv, *7*, *15*
read.table, *7*, *15*

sim.cross, *8*

tails *(*internal*)*, 2
think_barley.rda, 14

x.val, *8*, *14*, 14
XValidate_Ind *(*internal*)*, 2
XValidate_nonInd *(*internal*)*, 2

y.in_ex *(*think_barley.rda*)*, 14