

Package ‘PopVar’

February 7, 2021

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

Version 1.3.0

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

RoxygenNote 7.1.1

Depends R (>= 3.5.0)

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

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Tyler Tiede [aut],
Jeffrey Neyhart [aut, ctb, cre]
(<<https://orcid.org/0000-0002-1991-5310>>),
Mohsen Mohammadi [ctb] (<<https://orcid.org/0000-0002-4536-1200>>),
Kevin Smith [ctb] (<<https://orcid.org/0000-0001-8253-3326>>)

Maintainer Jeffrey Neyhart <neyharttje@gmail.com>

Repository CRAN

Date/Publication 2021-02-07 20:10:17 UTC

R topics documented:

PopVar-package	2
internal	3
mppop.predict	4
pop.predict	6
pop.predict2	11
think_barley.rda	14
x.val	15

Index	18
--------------	-----------

PopVar-package	<i>Genomic breeding tools to 1) predict standard statistics and correlated response in plant populations, and 2) performs cross-validation to estimate genome-wide prediction accuracy</i>
----------------	--

Description

PopVar includes two functions useful for genome-based breeding:

- `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 value (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.
- `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, see Details in `x.val` documentation for more information.

The dataset `think_barley.rda`, 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 the functions within PopVar.

Author(s)

Tyler Tiede (maintainer) <tyler.tiede7@gmail.com> and Mohsen Mohammadi

Many thanks to Kevin Smith for supporting the project and Jeff Neyhart for helping with the initial 'd

References

- Mohammadi M., T. Tiede, and K.P. Smith. 2015. PopVar: A genome-wide procedure for predicting genetic variance and correlated response in bi-parental breeding populations. *Crop Sci. Accepted*.
- 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)

internal	<i>Internal functions</i>
----------	---------------------------

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	<i>Predict genetic variance and genetic correlations in multi-parent populations using a deterministic equation.</i>
---------------	--

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	See crossing.table in pop.predict .
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. Supersedes `y.in` if passed.

Details

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

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`.

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>

<code>pop.predict</code>	<i>A genome-wide procedure for predicting genetic variance and correlated response in bi-parental breeding populations</i>
--------------------------	--

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 - `pop.predict` writes and reads files to disk so it is highly recommended to set your work

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 = F,
min.maf = 0.01,
mkr.cutoff = 0.5,
entry.cutoff = 0.5,
remove.dups = T,
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 = F
)

```

Arguments

G.in	<p>Matrix of genotypic data. First row contains marker names and the first column contains entry (taxa) names. Genotypes should be coded as follows:</p> <ul style="list-style-type: none"> • 1: homozygous for minor allele • 0: heterozygous • -1: homozygous for major allele • NA: missing data • Imputed genotypes can be passed, see <code>impute</code> below for details <p>TIP - Set <code>header=FALSE</code> within <code>read.table</code> or <code>read.csv</code> when importing a tab-delimited file containing data for G.in.</p>
y.in	<p>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 <code>header=TRUE</code> within <code>read.table</code> or <code>read.csv</code> when importing a tab-delimited file containing data for y.in.</p>
map.in	<p>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 <code>header=TRUE</code> within <code>read.table</code> or <code>read.csv</code> when importing a tab-delimited file containing data for map.in.</p>
crossing.table	<p>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).</p>
parents	<p>Optional character vector. If <code>parents="TP"</code> 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.</p>
tail.p	<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.</p>

nInd	Optional integer indicating the number of progeny simulated per cross, per iteration, using <code>sim.cross</code> 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 <code>plot.map</code> . 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 <code>impute</code> below). Default is TRUE.
impute	Options include <code>c("EM", "mean", "pass")</code> . By default (i.e. "EM"), after filtering missing genotypic data will be imputed via the EM algorithm implemented in <code>rrBLUP</code> (Endelman, 2011; Poland et al., 2012). If "mean" missing genotypic data will be imputed via the 'marker mean' method, also implemented in <code>rrBLUP</code> . 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 <code>x.val</code>). The remaining $(1 - frac.train)$ of the TP will then comprise the prediction set.
nCV.iter	Optional integer indicating the number of times to iterate <i>CV method 1</i> (see Details in <code>x.val</code>). Default is 100.
nFold	Optional integer. If a number is provided, denoting the number of "folds", then CV will be conducted using <i>CV method 2</i> (see Details in <code>x.val</code>). Default is NULL, resulting in the default use of the <i>CV method 1</i> .
nFold.reps	Optional integer indicating the number of times <i>CV method 2</i> is repeated. The CV accuracy returned is the average r of each rep. Default is 1.
nIter, burnIn	Optional integer arguments used by <code>BGLR</code> (de los Campos 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 <code>rrBLUP</code> , <code>BayesA</code> , <code>BayesB</code> , <code>BayesC</code> , <code>BL</code> , <code>BRR</code> , 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 <code>pop.predict</code> will return the results of each simulation in addition to the summarized dataframe. Default is FALSE.

Details

pop.predict can be used to predict the mean (μ), genetic variance (V_G), superior progeny values (μ_{sp}), 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 rrBLUP. 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 rrBLUP (*Endelman, 2011*) and BayesA, BayesB, BayesC π , Bayesian lasso (BL), and Bayesian ridge regression (BRR) implemented in BGLR (*de los Campos 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 (μ_{sp}) 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 `\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 Populations. *Genetics* 196:1111-1121.
- Broman, K. W., H. Wu, S. Sen and G.A. Churchill. 2003. R/qtl: QTL mapping in experimental crosses. *Bioinformatics* 19:1899-1901.
- Endelman, J. B. 2011. Ridge regression and other kernels for genomic selection with R package rrBLUP. *PLoS ONE* 6:e23764.
- Gustavo de los Campos and Paulino Perez Rodriguez, (2014). BGLR: Bayesian Generalized Linear Regression. *BMC Bioinformatics* 15:1-12.
- Mohammadi M., T. Tiede, and K.P. Smith. 2015. PopVar: A genome-wide procedure for predicting genetic variance components. *Genetics* 195:1111-1121.
- Munoz-Amatriain, M., M. J. Moscou, P. R. Bhat, J. T. Svensson, J. Bartos, P. Suchankova, H. Simkova, T. F. C. S. et al. 2015. Genomic selection in experimental crosses using a Bayesian approach. *Genetics* 195:1111-1121.
- Poland, J., J. Endelman, J. Dawson, J. Rutkoski, S. Wu, Y. Manes, S. Dreisigacker, J. Crossa, H. Sanchez

Examples

```
## Not run:
## View formatting
## Use View() in RStudio or R GUI with X11 forwarding
## Use head() in R GUI without X11 forwarding
View(G.in_ex)
View(y.in_ex)
View(map.in_ex)
View(cross.tab_ex)

## setwd() - pop.predict writes and reads files to disk
## so it is recommended to set your working directory

## nSim and nFold are set to low values in the
## examples for sake of computing time

## 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=10)
ex1.out$predictions ## Predicted parameters
ex1.out$CVs         ## CV results

## Ex. 2 - Predict all pairwise crosses between a list of parents
## This example uses CV method 2 (see Details of x.val() function)
par.list <- sample(y.in_ex[,1], size = 10, replace = F)
ex2.out <- pop.predict(G.in = G.in_ex, y.in = y.in_ex,
  map.in = map.in_ex, parents = par.list,
```

```

nSim=5, nFold=5, nFold.reps=2)

## Ex. 3 - 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", "BL"), nSim=5, nCV.iter=10)

## End(Not run)

```

pop.predict2	<i>Predict genetic variance and genetic correlations in bi-parental populations using a deterministic model</i>
--------------	---

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

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

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. Teyssède, 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

```

## Not run:

# 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])

## End(Not run)

# 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])

# Make predictions for 5 crosses with various levels of inbreeding
out_list <- lapply(X = 1:10, FUN = function(self.gen) {
  out <- pop_predict2(M = G.in_ex_mat, 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])
```

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 $n \times p$ matrix of $n = 245$ barley lines genotyped with $p = 742$ SNP markers

G.in_ex_imputed A $n \times p$ 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 - `x.val`, specifically `\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 = T,
  impute = "EM",
  frac.train = 0.6,
  nCV.iter = 100,
  nFold = NULL,
  nFold.reps = 1,
  return.estimates = F,
  CV.burnIn = 750,
  CV.nIter = 1500,
  models = c("rrBLUP", "BayesA", "BayesB", "BayesC", "BL", "BRR")
)
```

Arguments

G.in	<p>Matrix of genotypic data. First row contains marker names and the first column contains entry (taxa) names. Genotypes should be coded as follows:</p> <ul style="list-style-type: none"> • 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 <p>TIP - Set header=FALSE within read.table or read.csv when importing a tab-delimited file containing data for G.in.</p>
y.in	<p>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.</p>

	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 rrBLUP (Endelman, 2011; Poland et al., 2012). If "mean" missing genotypic data will be imputed via the 'marker mean' method, also implemented in rrBLUP. 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.estimate	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 Campos 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.

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 amendability 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.estimate`s 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

```
## Not run:
## CV using method 1 with 25 iterations
CV.mthd1 <- x.val(G.in = G.in_ex, y.in = y.in_ex, nCV.iter = 25)
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)

## End(Not run)
```

Index

BGLR, [8](#), [9](#), [16](#)

`calc_marker_effects` (internal), [3](#)
`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, [3](#)

`maf_filt` (internal), [3](#)
`map.in_ex` (think_barley.rda), [14](#)
`mppop.predict`, [4](#)
`mppop.predict2` (mppop.predict), [4](#)

`par_name` (internal), [3](#)
`par_position` (internal), [3](#)
`plot.map`, [8](#)
`pop.predict`, [2](#), [5](#), [6](#), [6](#), [12](#), [14](#), [17](#)
`pop.predict2`, [11](#)
`pop_predict2` (pop.predict2), [11](#)
PopVar (PopVar-package), [2](#)
PopVar-package, [2](#)

`read.csv`, [7](#), [15](#)
`read.table`, [7](#), [15](#)
rrBLUP, [8](#), [9](#), [16](#)

`sim.cross`, [8](#)

`tails` (internal), [3](#)
think_barley.rda, [2](#), [14](#)

`x.val`, [2](#), [8](#), [14](#), [15](#)
`XValidate_Ind` (internal), [3](#)
`XValidate_nonInd` (internal), [3](#)

`y.in_ex` (think_barley.rda), [14](#)