Package ‘HaploCatcher’

July 21, 2025

Title A Predictive Haplotyping Package

Date 2023-3-23

Version 1.0.4

Description Used for predicting a genotype’s allelic state at a specific locus/QTL/gene. This is accomplished by using both a genotype matrix and a separate file which has categorizations about loci/QTL/genes of interest for the individuals in the genotypic matrix. A training population can be created from a panel of individuals who have been previously screened for specific loci/QTL/genes, and this previous screening could be summarized into a category. Using the categorization of individuals which have been genotyped using a genome wide marker platform, a model can be trained to predict what category (haplotype) an individual belongs in based on their genetic sequence in the region associated with the locus/QTL/gene. These trained models can then be used to predict the haplotype of a locus/QTL/gene for individuals which have been genotyped with a genome wide platform yet not genotyped for the specific locus/QTL/gene. This package is based off work done by Winn et al 2021. For more specific information on this method, refer to <doi:10.1007/s00122-022-04178-w>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports base, parallel, doParallel, foreach, caret, ggplot2, graphics, knitr, patchwork, lattice, randomForest

Depends R (>= 2.10)

LazyData true

Suggests rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Zachary Winn [aut, cre] (ORCID: <https://orcid.org/0000-0003-1543-1527>)

Maintainer Zachary Winn <zwinn@outlook.com>

Repository CRAN

Date/Publication 2023-04-21 23:32:39 UTC

Contents

auto_locus 2
gene_comp 5
geno_mat 6
locus_cv 6
locus_perm_cv 9
locus_pred 12
locus_train 13
marker_info 16
plot_locus_perm_cv 17

Index **18**

`auto_locus` *Auto Locus - An Automated Pipeline for Locus Prediction*

Description

This function takes all other functions of the HaploCatcher package and weaves them into one pipeline that performs locus prediction with minimal user intervention.

Usage

```
auto_locus(
  geno_mat,
  gene_file,
  gene_name,
  marker_info,
  chromosome,
  training_genotypes,
  testing_genotypes,
  ncor_markers = 50,
  n_neighbors = 50,
  cv_percent_testing = 0.2,
  cv_percent_training = 0.8,
  n_perms = 30,
  model_selection_parameter = "kappa",
  n_votes = 30,
  set_seed = NULL,
  predict_by_vote = FALSE,
  include_hets = FALSE,
  include_models = FALSE,
  verbose = TRUE,
  parallel = FALSE,
  n_cores = NULL,
  plot_cv_results = TRUE
)
```

Arguments

`geno_mat` An imputed, number-coded, genotypic matrix which has n rows of individuals and m columns of markers. Row names of the matrix should be representative of genotypic IDs and column names should be representative of marker IDs. Missing data is not allowed. Numeric coding of genotypes can vary as long as it remains consistant among markers.

`gene_file` A dataframe containing at least three columns labeled as follows: 'Gene', 'FullSampleName', and 'Call'. The 'Gene' column contains the name of the gene for which the observation belongs to. The 'FullSampleName' column contains the genotypic ID which corresponds exactly to the column name in the genotypic matrix. The 'Call' column contains the marker call which corresponds to the gene for that genotype. Other information may be present in this dataframe beyond these columns, but the three listed columns above are obligatory.

`gene_name` A character string which matches the name of the gene which you are trying to perform cross validation for. This character string must be present in your gene_file 'Gene' column.

`marker_info` A dataframe containing the following three columns: 'Marker', 'Chromosome', and 'BP_Position'. The 'Marker' column contains the names of the marker which are present in the genotypic matrix. The 'Chromosome' column contains the corresponding chromosome/linkage group to which the marker belongs. The 'Position' column contains the physical or centimorgan position of the marker. All markers present in the genotypic matrix must be listed in this dataframe. If physical or centimorgan positions are unavailable for the listed markers, a numeric dummy variable ranging from one to n number of markers may be provided instead.

`chromosome` A character string which matches the name of the chromosome upon which the gene resides. This chromosome name must be present in the marker_info file.

`training_genotypes` A character vector of fullsamplenames found in the geno_mat and gene_file. These are lines which will be used in cross validation and to train the model which will predict the lines which have no locus call available.

`testing_genotypes` A character vector of fullsamplenames found in the geno_mat. These are lines which will have predictions made for them.

`ncor_markers` A numeric variable which represents the number of markers the user want to use in model training. Correlation among markers to the gene call is calculated and the top n markers specified are retained for training. The default setting is 50 markers.

`n_neighbors` A numeric variable which represents the number of neighbors to use in KNN. Default is 50.

`cv_percent_testing` A numeric variable which ranges such that x|0<x<1. This means that this number can be neither zero nor one. This number represents the percent of the total data available the user wants to retain to validate the model. The default setting is 0.20.

`cv_percent_training`
: A numeric variable which ranges such that x|0<x<1. This means that the number can be neither zero nor one. This number represents the percent of the total data available the user wants to retain for training of the model.The default setting is 0.80.

`n_perms`
: A numeric variable defining the number of permutations to perform. This value may range from one to infinity. Default is 30.

`model_selection_parameter`
: A character string which defines which model performance parameter the user wants to use to select the best performing model. Only two strings are accepted either "kappa" for selection on Cohen's kappa value or "accuracy" for selection on total accuracy. Default is "kappa".

`n_votes`
: A numeric variable defining the number of models to train and predict with. This method will take n number of predictions over multiple iterations, sum across those multiple iterations and provide a prediction which is based on majority rule. The default is 30.

`set_seed`
: A numeric variable that is used to set a seed for reproducible results if the user is running the "locus_trait" function once for use in the "locus_pred" function. If the user wishes to run the function many times with a random seed and decide the outcome by voting, set 'predict_by_vote' to TRUE instead. The default setting is NULL.

`predict_by_vote`
: A logical variable which defines if the user wants predictions generated by voting over several runs of machine models summarized into a majority rule call. Default is FALSE.

`include_hets`
: A logical variable which determines if the user wishes to include heterozygous calls or not. The default setting is FALSE.

`include_models`
: A logical variable which determines if the user wishes to include the trained models in the results object for further testing. Warning: the models are quite large and running this will result in a very large results object. The default setting is FALSE.

`verbose`
: A logical variable which determines if the user wants plots displayed and text feedback from each permutation. Regardless of this parameter, the function will display the name of the gene which is being cross validated and the current progress of the permutations. Default setting is TRUE.

`parallel`
: A logical variable which determines if the user wants the cross validation and voting predictions to be performed in parallel. Default is FALSE. If the user defines that parallel is TRUE, all visual and textual feedback will not be rendered.

`n_cores`
: A numeric variable that is used to set a seed for reproducible results if the user is running the function once for use in the "locus_pred" function. If the user wishes to run the function many times with a random seed and decide the outcome by voting, set 'predict_by_vote' to true instead. The default setting is NULL.

`plot_cv_results`
: A logical variable which determines if the user want the cross validation results to be printed as a 'ggplot' image. Default is TRUE

Value

This function returns two types of list-of-list objects. If 'predict_by_vote' is set to FALSE (the default setting), then the list will contain four objects: method, cross_validation_results, prediction_model, and predictions. The method object is a character which denotes what method was used. The cross_validation_results object contains the results of the cross-validation. The prediction model contains the information about the prediction model, and predictions contains the predictions for the testing_genotypes. If 'predict_by_vote' is set to TRUE, then the list will contain four objects: method, cross_validation_results, predictions, and consensus_predictions. Both method and cross_validation_results are the same for this option. Predictions contains the prediction of each model ran to vote. The object consensus_predictions contains a consensus prediction of the alleleic state of a genotype based on majority rule and a tablular summary of how many votes were taken per catigory.

Examples

```
#refer to vignette for an in depth look at the auto_locus function
vignette("An_Intro_to_HaploCatcher", package = "HaploCatcher")
```

---

gene_comp *Model Gene Compendium Data Set*

Description

A data frame which contains information from 1345 unique wheat lines on the Sst1 solid stem locus.

Usage

```
gene_comp
```

Format

A data frame with 1345 rows and 7 columns:

Trait A short discription of the phenotype associated with the gene

Chromosome The chromosome where the gene resides

Gene The name of the gene

Nursery The program which produced the gene call for the genotype

Line A breeder assigned line designation

FullSampleName A designation unique to the line found in the genotypic matrix

Call A 'call' given for the allelic state. For this package, it is best to format the non desiarable allele as "non_gene" and the heterozygous state as "het_gene".

Source

Generated by Zachary James Winn for the CSU breeding program via USDA-ARS gene reports and in-house gene assays

Examples

```
data("gene_comp") #lazy loads the dataset for use in the package
```

`geno_mat` *Model Gene Compendium Data Set*

Description

A numeric matrix which contains molecular marker information on 1345 unique genotypes for 2271 SNP markers located on wheat chromosome 3B. This data set corresponds to the information found in the "gene_comp" and "marker_info" data sets.

Usage

```
geno_mat
```

Format

A numeric matrix with 1345 rows and 2271 columns:

Source

Generated by Zachary James Winn for the CSU breeding program via historical in-house GBS data

Examples

```
data("geno_mat") #lazy loads the dataset for use in the package
```

`locus_cv` *Haplotype Prediction: Cross Validation of KNN and RF Models*

Description

This function performs the analysis featured in Winn et al 2022 where genome wide markers are used to train machine learning models to identify if genotypes have or do not have specific alleles of a QTL/gene. This function is used to perform cross validation where a random partition of the total available data is used to train a model and a reserved testing partition is used to validate. This function is used for a single round of cross validation.

Usage

```
locus_cv(
  geno_mat,
  gene_file,
  gene_name,
  marker_info,
  chromosome,
  ncor_markers = 50,
  n_neighbors = 50,
  percent_testing = 0.2,
  percent_training = 0.8,
  include_hets = FALSE,
  include_models = FALSE,
  verbose = TRUE,
  graph = FALSE
)
```

Arguments

`geno_mat` An imputed, number-coded, genotypic matrix which has n rows of individuals and m columns of markers. Row names of the matrix should be representative of genotypic IDs and column names should be representative of marker IDs. Missing data is not allowed. Numeric coding of genotypes can vary as long as it remains consistant among markers.

`gene_file` A dataframe containing at least three columns labeled as follows: 'Gene', 'FullSampleName', and 'Call'. The 'Gene' column contains the name of the gene for which the observation belongs to. The 'FullSampleName' column contains the genotypic ID which corresponds exactly to the column name in the genotypic matrix. The 'Call' column contains the marker call which corresponds to the gene for that genotype. Other information may be present in this dataframe beyond these columns, but the three listed columns above are obligatory.

`gene_name` A character string which matches the name of the gene which you are trying to perform cross validation for. This character string must be present in your gene_file 'Gene' column.

`marker_info` A dataframe containing the following three columns: 'Marker', 'Chromosome', and 'BP_Position'. The 'Marker' column contains the names of the marker which are present in the genotypic matrix. The 'Chromosome' column contains the corresponding chromosome/linkage group to which the marker belongs. The 'Position' column contains the physical or centimorgan position of the marker. All markers present in the genotypic matrix must be listed in this dataframe. If physical or centimorgan positions are unavailable for the listed markers, a numeric dummy variable ranging from one to n number of markers may be provided instead.

`chromosome` A character string which matches the name of the chromosome upon which the gene resides. This chromosome name must be present in the marker_info file.

`ncor_markers` A numeric variable which represents the number of markers the user want to use in model training. Correlation among markers to the gene call is calculated and

the top n markers specified are retained for training. The default setting is 50 markers.

`n_neighbors` A numeric variable which represents the number of neighbors to use in KNN. Default is 50.

`percent_testing` A numeric variable which ranges such that x|0<x<1. This means that this number can be neither zero nor one. This number represents the percent of the total data available the user wants to retain to validate the model. The default setting is 0.20.

`percent_training` A numeric variable which ranges such that x|0<x<1. This means that the number can be neither zero nor one. This number represents the percent of the total data available the user wants to retain for training of the model.The default setting is 0.80.

`include_hets` A logical variable which determines if the user wishes to include heterozygous calls or not. Default is FALSE.

`include_models` A logical variable which determines if the user wishes to include the trained models in the results object for further testing. Warning: the models are quite large and running this will result in a very large results object. Default is FALSE.

`verbose` A logical variable which determines if the user wants text feedback. Default is TRUE.

`graph` A logical variable which determines if the user wants plots displayed. Default is FALSE.

Value

This function returns a list of list which contains the following list objects: 'confu', 'preds', 'models', and 'data'. The 'confu' list contains the confusion matrix objects for both the random forest and k-nearest neighbors models. The 'preds' list contains the predictions made by the separate models. The 'models' contains the two caret model objects for both the random forest and k-nearest neighbors models. The 'data' list contains the training and test data frames made by the function.

Examples

```
#read in the genotypic data matrix
data("geno_mat")

#read in the marker information
data("marker_info")

#read in the gene compendium file
data("gene_comp")

#run the function without hets for a very limited number of markers and neighbors
#due to requirements by cran, this must be commented out
#to run, place this code in the console and remove comments
#fit<-locus_cv(geno_mat=geno_mat, #the genotypic matrix
#             gene_file=gene_comp, #the gene compendium file
#             gene_name="sst1_solid_stem", #the name of the gene
```

```
#             marker_info=marker_info, #the marker information file
#             chromosome="3B", #name of the chromosome
#             ncor_markers=2, #number of markers to retain
#             n_neighbors=1, #number of neighbors
#             percent_testing=0.2, #percentage of genotypes in the validation set
#             percent_training=0.8, #percentage of genotypes in the training set
#             include_hets=FALSE, #include hets in the model
#             include_models=TRUE, #include models in the final results
#             verbose=TRUE, #allows text output
#             graph=TRUE) #allows graph output
```

`locus_perm_cv` *Haplotype Prediction: Permutation Cross Validation of KNN and RF Models*

Description

This function performs the analysis featured in Winn et al 2022 where genome wide markers are used to train machine learning models to identify if genotypes have or do not have specific alleles of a QTL/gene. This function is used to perform cross validation where a random partition of the total available data is used to train a model and a reserved testing partition is used to validate. This function is used for permutation based cross validation.

Usage

```
locus_perm_cv(
  n_perms = 30,
  geno_mat,
  gene_file,
  gene_name,
  marker_info,
  chromosome,
  ncor_markers = 50,
  n_neighbors = 50,
  percent_testing = 0.2,
  percent_training = 0.8,
  include_hets = FALSE,
  include_models = FALSE,
  verbose = FALSE,
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`n_perms`	A numeric variable defining the number of permutations to perform. This value may range from one to infinity. Default is 30.

`geno_mat` An imputed, number-coded, genotypic matrix which has n rows of individuals and m columns of markers. Row names of the matrix should be representative of genotypic IDs and column names should be representative of marker IDs. Missing data is not allowed. Numeric coding of genotypes can vary as long as it remains consistent among markers.

`gene_file` A dataframe containing at least three columns labeled as follows: 'Gene', 'FullSampleName', and 'Call'. The 'Gene' column contains the name of the gene for which the observation belongs to. The 'FullSampleName' column contains the genotypic ID which corresponds exactly to the column name in the genotypic matrix. The 'Call' column contains the marker call which corresponds to the gene for that genotype. Other information may be present in this dataframe beyond these columns, but the three listed columns above are obligatory.

`gene_name` A character string which matches the name of the gene which you are trying to perform cross validation for. This character string must be present in your gene_file 'Gene' column.

`marker_info` A dataframe containing the following three columns: 'Marker', 'Chromosome', and 'BP_Position'. The 'Marker' column contains the names of the marker which are present in the genotypic matrix. The 'Chromosome' column contains the corresponding chromosome/linkage group to which the marker belongs. The 'Position' column contains the physical or centimorgan position of the marker. All markers present in the genotypic matrix must be listed in this dataframe. If physical or centimorgan positions are unavailable for the listed markers, a numeric dummy variable ranging from one to n number of markers may be provided instead.

`chromosome` A character string which matches the name of the chromosome upon which the gene resides. This chromosome name must be present in the marker_info file.

`ncor_markers` A numeric variable which represents the number of markers the user want to use in model training. Correlation among markers to the gene call is calculated and the top n markers specified are retained for training. The default setting is 50 markers.

`n_neighbors` A numeric variable which represents the number of neighbors to use in KNN. Default is 50.

`percent_testing`
A numeric variable which ranges such that x|0<x<1. This means that this number can be neither zero nor one. This number represents the percent of the total data available the user wants to retain to validate the model. The default setting is 0.20.

`percent_training`
A numeric variable which ranges such that x|0<x<1. This means that the number can be neither zero nor one. This number represents the percent of the total data available the user wants to retain for training of the model.The default setting is 0.80.

`include_hets` A logical variable which determines if the user wishes to include heterozygous calls or not. The default setting is FALSE.

`include_models` A logical variable which determines if the user wishes to include the trained models in the results object for further testing. Warning: the models are quite

large and running this will result in a very large results object. The default setting is FALSE.

`verbose` A logical variable which determines if the user wants plots displayed and text feedback from each permutation. Regardless of this parameter, the function will display the name of the gene which is being cross validated and the current progress of the permutations. Default setting is FALSE.

`parallel` A logical variable which determines if the user wants the cross validation performed in parallel. Default is FALSE. If the user defines that parallel is TRUE, all visual and textual feedback will not be rendered.

`n_cores` A numerical vector which denotes the number of cores used for parallel processor. If "parallel" option is TRUE and n_cores is not specified, then the number of available cores minus one will be assigned to processing.

Value

This function returns a list of list with the following objects: "Overall_Parameters", "By_Class_Parameters", "Overall_Summary", "By_Class_Summary", and "Raw_Permutation_Info". The "Overall_Parameters" data frame contains all the relevant parameters for each permutation overall. The "By_Class_Parameters" data frame contains all the relevant parameters for each permutation by class.The "Overall_Summary" data frame contains all the relevant parameters overall summarized across permutations. The "By_Class_Summary" data frame contains all the relevant parameters by class summarized across permutations. The "Raw_Permutation_Info" is a list of list which contains each permutations model info as described in the "locus_cv" function.

Examples

```
#read in the genotypic data matrix
data("geno_mat")

#read in the marker information
data("marker_info")

#read in the gene compendium file
data("gene_comp")

#run permutational analysis - commented out for package specifications
#to run, copy and paste without '#' into the console

#fit<-locus_perm_cv(n_perms = 10, #the number of permutations
#                   geno_mat=geno_mat, #the genotypic matrix
#                   gene_file=gene_comp, #the gene compendium file
#                   gene_name="sst1_solid_stem", #the name of the gene
#                   marker_info=marker_info, #the marker information file
#                   chromosome="3B", #name of the chromosome
#                   ncor_markers= 25, #number of markers to retain
#                   n_neighbors = 25, #number of nearest-neighbors
#                   percent_testing=0.2, #percentage of genotypes in the validation set
#                   percent_training=0.8, #percentage of genotypes in the training set
#                   include_hets=FALSE, #excludes hets in the model
#                   include_models=FALSE, #excludes models in results object
```

```
#                   verbose = FALSE) #excludes text
```

---

locus_pred *Haplotype Prediction: Using Trained Models to Make Predictions*

Description

Haplotype Prediction: Using Trained Models to Make Predictions

Usage

```
locus_pred(locus_train_results, geno_mat, genotypes_to_predict)
```

Arguments

`locus_train_results`
: This object is a the results object from the "locus_train" function.

`geno_mat`
: This is a genotypic matrix with genotypes of individuals you have genotyped and characterized for the locus/QTL/gene of interest and individuals that have only been genotyped with a genome wide marker panel. It is important to note that the markers in the genome wide panel *must* be shared between training population and the population you wish to forward predict. In the case of genotyping-by-sequencing markers, the training and test populations should be discovered and produced together into a genotype file. All marker platforms, however, are compatable, as long as the training and forward prediction population share the same markers genome-wide.

`genotypes_to_predict`
: This is a character vector of genotypes in the geno_mat which the user wishes to predict. If this object contains names in the training population, they will be omitted in the results to avoid bias.

Value

a data frame with two columns: genotype names and predictions.

Examples

```
#set seed for reproducible sampling
set.seed(022294)

#read in the genotypic data matrix
data("geno_mat")

#read in the marker information
data("marker_info")
```

```
#read in the gene compendium file
data("gene_comp")

#Note: in practice you would have something like a gene file
#that does not contain any lines you are trying to predict.
#However, this is for illustrative purposes on how to run the function

#sample data in the gene_comp file to make a traning population
train<-gene_comp[gene_comp$FullSampleName %in%
                   sample(gene_comp$FullSampleName,
                          round(length(gene_comp$FullSampleName)*0.8),0),]

#pull vector of names, not in the train, for forward prediction
test<-gene_comp[!gene_comp$FullSampleName
                %in% train$FullSampleName,
                "FullSampleName"]

#run the function with hets
fit<-locus_train(geno_mat=geno_mat, #the genotypic matrix
                 gene_file=train, #the gene compendium file
                 gene_name="sst1_solid_stem", #the name of the gene
                 marker_info=marker_info, #the marker information file
                 chromosome="3B", #name of the chromosome
                 ncor_markers=2, #number of markers to retain
                 n_neighbors=3, #number of neighbors
                 include_hets=FALSE, #include hets in the model
                 verbose = FALSE, #allows for text and graph output
                 set_seed = 022294, #sets a seed for reproduction of results
                 models = "knn") #sets what models are requested

#predict the lines in the test population
pred<-locus_pred(locus_train_results=fit,
                 geno_mat=geno_mat,
                 genotypes_to_predict=test)

#see predictions
head(pred)
```

`locus_train` *Haplotype Prediction: Training Models for Use in Forward Prediction*

Description

This function is used to train a model for use in forward prediction of lines which have no record

Usage

```
locus_train(
  geno_mat,
```

```
  gene_file,
  gene_name,
  marker_info,
  chromosome,
  ncor_markers = 50,
  n_neighbors = 50,
  include_hets = FALSE,
  verbose = FALSE,
  set_seed = NULL,
  models_request = "all",
  graph = FALSE
)
```

Arguments

`geno_mat` An imputed, number-coded, genotypic matrix which has n rows of individuals and m columns of markers. Row names of the matrix should be representative of genotypic IDs and column names should be representative of marker IDs. Missing data is not allowed. Numeric coding of genotypes can vary as long as it remains consistant among markers.

`gene_file` A dataframe containing at least three columns labeled as follows: 'Gene', 'FullSampleName', and 'Call'. The 'Gene' column contains the name of the gene for which the observation belongs to. The 'FullSampleName' column contains the genotypic ID which corresponds exactly to the column name in the genotypic matrix. The 'Call' column contains the marker call which corresponds to the gene for that genotype. Other information may be present in this dataframe beyond these columns, but the three listed columns above are obligatory.

`gene_name` A character string which matches the name of the gene which you are trying to perform cross validation for. This character string must be present in your gene_file 'Gene' column.

`marker_info` A dataframe containing the following three columns: 'Marker', 'Chromosome', and 'BP_Position'. The 'Marker' column contains the names of the marker which are present in the genotypic matrix. The 'Chromosome' column contains the corresponding chromosome/linkage group to which the marker belongs. The 'Position' column contains the physical or centimorgan position of the marker. All markers present in the genotypic matrix must be listed in this dataframe. If physical or centimorgan positions are unavailable for the listed markers, a numeric dummy variable ranging from one to n number of markers may be provided instead.

`chromosome` A character string which matches the name of the chromosome upon which the gene resides. This chromosome name must be present in the marker_info file.

`ncor_markers` A numeric variable which represents the number of markers the user want to use in model training. Correlation among markers to the gene call is calculated and the top n markers specified are retained for training. The default setting is 50 markers.

`n_neighbors` A numeric variable which represents the number of neighbors to use in KNN. Default is 50.

include_hets	A logical variable which determines if the user wishes to include heterozygous calls or not. The default setting is FALSE.
verbose	A logical variable which determines if the user wants text feedback. Default setting is TRUE.
set_seed	A numeric variable that is used to set a seed for reproducible results if the user is running the function once for use in the "locus_pred" function. If the user wishes to run the function many times with a random seed and decide the outcome by voting, use the function "locus_voting" instead. The default setting is NULL.
models_request	A character string which defines what models are to be ran. K-nearest neighbors is abbreviated as "knn" and random forest is "rf". If both models are desired, use the text string "all". Default setting is "all".
graph	A logical variable which determines if the user wants graphs displayed. default setting is FALSE.

Value

This function returns a list of list which contains: "seed", "models_request" ,"models", and "data". The "seed" object is the seed set by the user. If no seed was provided this will appear as a character stating "no_seed_set". The "models_request" item hold the models requested. The "models" object contains the trained models. The "data" object contains the data used to train the models.

Examples

```
#set seed for reproducible sampling
set.seed(022294)

#read in the genotypic data matrix
data("geno_mat")

#read in the marker information
data("marker_info")

#read in the gene compendium file
data("gene_comp")

#Note: in practice you would have something like a gene file
#that does not contain any lines you are trying to predict.
#However, this is for illustrative purposes on how to run the function

#sample data in the gene_comp file to make a traning population
train<-gene_comp[gene_comp$FullSampleName %in%
                  sample(gene_comp$FullSampleName,
                         round(length(gene_comp$FullSampleName)*0.8),0),]

#pull vector of names, not in the train, for forward prediction
test<-gene_comp[!gene_comp$FullSampleName
                %in% train$FullSampleName,
                "FullSampleName"]

#run the function with hets
```

```
fit<-locus_train(geno_mat=geno_mat, #the genotypic matrix
                 gene_file=train, #the gene compendium file
                 gene_name="sst1_solid_stem", #the name of the gene
                 marker_info=marker_info, #the marker information file
                 chromosome="3B", #name of the chromosome
                 ncor_markers=2, #number of markers to retain
                 n_neighbors=3, #number of neighbors
                 include_hets=FALSE, #include hets in the model
                 verbose = FALSE, #allows for text and graph output
                 set_seed = 022294, #sets a seed for reproduction of results
                 models = "knn") #sets what models are requested

#predict the lines in the test population
pred<-locus_pred(locus_train_results=fit,
                 geno_mat=geno_mat,
                 genotypes_to_predict=test)

#see predictions
head(pred)
```

`marker_info` *Model Gene Compendium Data Set*

Description

A data frame which contains marker information of GBS markers found on wheat chromosome 3B. This data pairs with the markers found in "geno_mat" data file associated with the HaploCatcher package.

Usage

```
marker_info
```

Format

A data frame with 2271 rows and 3 columns:

Marker The designation of the markers which are found in the genotypic matrix

Chromosome The chromosome where each marker resides

BP_Position The position of each marker in basepairs

Source

Generated by Zachary James Winn for the CSU breeding program via historical in-house GBS data

Examples

```
data("marker_info") #lazy loads the dataset for use in the package
```

plot_locus_perm_cv *Visualize Permutation CV Results*

Description

This function takes a list of list object from the "locus_perm_cv" function and creates a summary graphic of the accuracy, kappa, sensitivity, and specificity of the models ran. If heterozygous individuals were left in the cross validation scheme, by-class sensitivity and specificity will be displayed; otherwise, displayed parameters will be of the overall model.

Usage

```
plot_locus_perm_cv(results, individual_images = FALSE)
```

Arguments

`results` An object of class "list" which is derived from the "locus_perm_cv" function.

`individual_images` A logical argument that defines if the user wants both the composite image and the full image. Default setting is FALSE.

Value

Prints a ggplot2 image

Examples

```
#refer to vignette for an in depth look at the plot_locus_perm_cv function
vignette("An_Intro_to_HaploCatcher", package = "HaploCatcher")
```

Index

∗ **datasets**
    gene_comp, 5
    geno_mat, 6
    marker_info, 16

auto_locus, 2

gene_comp, 5
geno_mat, 6

locus_cv, 6
locus_perm_cv, 9
locus_pred, 12
locus_train, 13

marker_info, 16

plot_locus_perm_cv, 17