Package 'PGSP'

September 2, 2020

Type Package

Title Progeno Genomic Selection Pipeline

Version 0.5.4

Author Steven Maenhout

Maintainer Steven Maenhout <Steven.Maenhout@progeno.net>

Description genomic selection pipeline that interconnects all tasks and computations that are required for a routine application of genomic selection in breeding programs

Depends R (>= 3.0.0)

Imports methods,
jsonlite (>= 1.5),
mongolite (== 1.4.9000),
data.table (>= 1.10.4),
Rcpp (>= 0.12.4),
openxlsx (>= 4.0.17),
R.utils (>= 2.6.0),
future (>= 1.10.0),
promises (>= 1.0.1),
pryr (>= 0.1.2),
progeno (>= 1.6.63),
pdpbuilder (>= 0.3.4),
randomForest (>= 4.6-14),
parallel (>= 3.3.1),
processx (>= 3.4.1),
bigmemory (>= 4.5.33),
rrBLUP (>= 4.6.1),
ggplot2 (>= 3.2.0),
plotly (>= 4.7.0),
rrBLUP (>= 4.6.1)

LinkingTo BH (>= 1.62.0-1),
Rcpp (>= 0.12.4)

Suggests testthat

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Collate 'init.R'
'DataMongoBase.R'
'DataEBVFactory.R'
'DataFile.R'
'DataGenoFactory.R'
'DataPhenoFactory.R'
'DataQuality.R'
'GenoImputation.R'
'GenomicPrediction.R'
'GenomicSelection.R'
'JobQueue.R'
'LogRegistry.R'
'ModelFactory.R'
'PDPFactory.R'
'PhenoAnalysis.R'
'RcppExports.R'
'zzz.R'

R topics documented:

`dataEBVFactory` *DataEBVFactory constructor*

Description

Creates an instance of class `DataEBVFactory`

Usage

```
dataEBVFactory()
```

Value

an instance of class DataEBVFactory

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

`DataEBVFactory_getBreedingValueEstimates`
Get breeding value estimates from the database.

Description

Retrieves breeding value estimates from a specific PDP.

Arguments

`pdpID`	integer containing the ID of the PDP from which breeding value estimates are to be retrieved
`traitNames`	character vector containing the names of the requested breeding values, missing or empty implies all available traits
`targetNames`	character vector containing the targets of the requested breeding values, missing or empty implies all available targets
`predictionTypes`	character vector containing the prediction types of the requested breeding values, missing or empty implies all available prediction types
`segmentNames`	character vector containing the segments of the requested breeding values, missing or empty implies all available segments
`indNames`	optional character vector containing the names of individuals for wich breeding values are to be returned
`collectionName`	character vector containing the name of the collection, defaults to BreedingValueEstimates.

Value

data.frame containing the requested breeding value estimates

DataEBVFactory_getPDPInfo *Get available PDPs from the database.*

Description

Retrieves the available PDPs and their metadata

Arguments

collectionName	character vector containing the name of the collection, defaults to PDPs

Value

data.frame containing information on the available PDPs

dataGenoFactory *DataGenoFactory constructor*

Description

Creates an instance of class DataPhenoFactory

Usage

```
dataGenoFactory()
```

Value

an instance of class DataPhenoFactory

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

`DataGenoFactory_getGenotypedIndividuals` *Lists individuals that have genotypic scores/loadings in the databse*

Description

Lists all indivdiuals that have marker scores/loadings in the databse

Arguments

`markerIDs`	optional integer vector containing the markerIDs that should be considered
`useMIPs`	logical, indicates if MIPs marker scores should be used instead of SNP markers, default is FALSE
`allowImputed`	logical, indicates if imputed marker scores should be considered, default is FALSE

Value

data.frame containing the names of individuals and their number of genotypic scores/loadings in the database

`DataGenoFactory_getGenotypedMarkers` *Lists markerIDs for which scores/loadings are available in the databse*

Description

Lists all markerIDs that have scores/loadings in the databse

Arguments

`indNames`	optional character vector containing the names of individuals that should be considered
`useMIPs`	logical, indicates if MIPs marker scores should be used instead of SNP markers, default is FALSE
`allowImputed`	logical, indicates if imputed marker scores should be considered, default is FALSE

Value

data.frame containing the markerIDs and their number of scores/loadings in the database

DataGenoFactory_getMarkerLoadings
Retrieve loading matrix from the registry

Description

Retrieve a matrix of marker loadings for a subset of individuals and marker numbers

Arguments

indNames	character vector containing the names of individuals that should be considered
markerIDs	optional integer vector containing the markerIDs that should be considered
reqPercentComplete	numeric indicating the minimum percentage (a number between 0 and 1) of individuals that each marker needs to be scored for, defaults to 0.7
reqMAF	numeric indicating the minimum MAF (minor allele frequency as a number between 0 and 1) that each marker needs to have, defaults to 0.01
useMIPs	logical, indicates if MIPs marker scores should be used instead of SNP markers, default is FALSE
allowImputed	logical, indicates if imputed marker scores can be included in the score matrix, default is FALSE

Value

numeric matrix of marker allele loadings (i.e. frequencies) with indivdiuals in rows and markerIDs in columns

DataGenoFactory_getPDPGenotypedIndividuals
Lists individuals that have been considered as genotyped in a PDP

Description

Lists all individuals that have been considered genotyped in a specific PDP

Arguments

pdpID	integer containing the ID of the PDP for which genotyped individuals should be retrieved

Value

character vector containing the names of individuals

DataMongoBase-class *Class offering basic MongoDB functionality.*

Description

Class offering basic MongoDB functionality.

Fields

`collectionName` character vector holding the connected MongoDB collection

`connection` mongolite connection object

DataMongoBase_changeCollection *Change the collection that the DataMongoBase is pointing to*

Description

Change the collection that the DataMongoBase is pointing to

Arguments

`collectionName` character vector holding the desired collection name

Value

nothing

DataMongoBase_downloadFile *download a file from GridFS.*

Description

Download a file from GridFS by id

Arguments

`fileID` character vector containing the file id.

Value

raw vector containing the file content (length = 0 if file was not found)

`DataMongoBase_extractMongoClientPtr`
Extract the mongo client pointer from mongolite

Description

Returns the mongo client pointer from mongolite

Value

externalptr pointing to a mongo_client_t C struct instance

`DataMongoBase_getDistinctKeys`
Get the distinct keys in a collection.

Description

Retrieves the unique set of keys that appear in the documents of a collection.

Arguments

`collectionName`	character vector containing the name of the collection.

Value

vector containing the distinct key names

`DataMongoBase_getDistinctValues`
Get the distinct values of a document key.

Description

Retrieves the unique set of values for the provided document key.

Arguments

`collectionName`	character vector containing the name of the collection.
`keyName`	character vector containing the name of the document key.
`queryList`	list defining the query to filter the documents

Value

vector containing the distinct values of the key

DataMongoBase_getFileList *Get the list of files available in GridFS.*

Description

Retrieves the a list of files that have been stored in GridFS.

Arguments

queryList	optional list defining the query to filter the files

Value

data.frame containing the files in GridFS and their properties

DataMongoBase_removeFile *Remove a file from GridFS.*

Description

Remove a file from GridFS by id

Arguments

fileID	character vector containing the file id.

Value

raw vector containing the file content

DataMongoBase_uploadFile *upload file and metadata to GridFS.*

Description

Uploads a file and associated metadata in GridFS

Arguments

filepath	character vector containing the file.
metadata	list withe named elements such as description

Value

character vector containing the ID of the stored file

dataPhenoFactory *DataPhenoFactory constructor*

Description

Creates an instance of class DataPhenoFactory

Usage

```
dataPhenoFactory()
```

Value

an instance of class DataPhenoFactory

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

`DataPhenoFactory_getBreedingValueInfo`
Get available breeding value types.

Description

Retrieves the available breeding values for a PDP

Arguments

`pdpID`	integer containing the ID of the PDP from which breeding value types are to be retrieved
`collectionName`	character vector containing the name of the collection, defaults to BreedingValues

Value

data.frame containing the available breeding value types

`DataPhenoFactory_getDistinctTraits`
Get the distinct observation traits in a collection.

Description

Retrieves the unique set of traits that appear in the TrialData array of the documents in a collection.

Arguments

`queryList`	list defining the query to filter the documents
`collectionName`	character vector containing the name of the collection, defaults to TrialData

Value

vector containing the distinct trait names

DataPhenoFactory_getMultiTrialAnalysisData

Get trait observations for multiple trials

Description

Retrieves the observations for a single trial and trait in a suitable form for analysis

Arguments

pdpTypeID	numeric vector containing the pdpTypeID for which the observations are to be retrieved
traitName	character vector containing the name of the trait
trialIDs	optional numeric vector containing the trialIDs for which the observations are to be retrieved, if missing, all available and connected trials are returned
includeOutliers	logical indicating if previously detected outliers should be included in the returned observations, defaults to FALSE

Value

data.frame containing the analysis data

DataPhenoFactory_getObservationKeys

Get observation keys.

Description

Retrieves the available keys that allow to identify observations.

Arguments

collectionName	character vector containing the name of the collection, defaults to TrialDataKeys.

Value

data.frame containing the available observation keys

`DataPhenoFactory_getObservationKeyValues` *Get observation key values.*

Description

Retrieves the available values for an observaion key.

Arguments

`key`	character containing the name of the key
`queryList`	optional list defining the query to filter the observations from which the key values are listed
`collectionName`	character vector containing the name of the collection, defaults to TrialData.

Value

data.frame containing the unique values for an observation key and their frequency

`DataPhenoFactory_getObservations` *Get observations from the database*

Description

Retrieves the observations for selected traits.

Arguments

`traitNames`	character vector containing the names of the requested traits, missing or empty implies all observed traits
`queryList`	list defining the query to filter the documents
`returnFields`	character vector containing the document keys to return besides the trait observations
`forceNumeric`	logical, if TRUE the observations are converted to numeric
`aggregate`	logical, only relevant if forceNumeric=TRUE, if TRUE the numeric observations are averaged over the levels of the document keys in returnFields
`collectionName`	character vector containing the name of the collection, defaults to TrialData.

Value

data.frame containing the requested observations

`DataPhenoFactory_getPedigree`
Get pedigree records from a collection.

Description

Retrieves the pedigree for selected breeding pool members.

Arguments

`names`	character vector containing the names of the accessions who's pedigree is to be retrieved
`collectionName`	character vector containing the name of the collection, defaults to Individuals.

Value

data.frame containing the full pedigree

`DataPhenoFactory_getSingleTrialAnalysisData`
Get observations of a particular trial and trait

Description

Retrieves the observations for a single trial and trait in a suitable form for analysis

Arguments

`pdpTypeID`	numeric vector containing the pdpTypeID for which the observations are to be retrieved, defaults to 1
`traitName`	character vector containing the name of the trait
`trialID`	numeric vector containing the trialID for which the observations are to be retrieved

Value

data.frame containing the analysis data

DataPhenoFactory_getTraitInfo

Get trait information.

Description

Retrieves information on traits

Arguments

traitNames	optional character vector containing the names of the traits for which information is to be retrieved
collectionName	character vector containing the name of the collection, defaults to TraitDefinitions

Value

data.frame containing information on the requested traits

DataPhenoFactory_getTrialInfo

Get trial information.

Description

Retrieves information on trials.

Arguments

collectionName	character vector containing the name of the collection, defaults to Trials
queryList	optional list defining the query to filter the returned trials

Value

data.frame containing information on requested trials

DataPhenoFactory_getTrialObservations

Get observations from specific trials.

Description

Retrieves the observations for selected trials and traits.

Arguments

trialIDs	character or numeric vector containing the trialIDs for which the observations are to be retrieved
traitNames	optional character vector containing the names of the requested traits, missing or empty implies all observed traits
forceNumeric	logical, if TRUE the observations are converted to numeric
collectionName	character vector containing the name of the collection, defaults to TrialData.

Value

data.frame containing the trial observations

dataQuality

DataQuality constructor

Description

Creates an instance of class DataQuality

Usage

```
dataQuality(phenoData, genoData, pedigreeData)
```

Arguments

phenoData	optional data.frame where the first column is of type character and contains identifiers of the individuals and one or more numeric columns containing the phenotypic observations of one or more traits
genoData	optional numeric matrix containing marker scores, individuals in rows and markers in columns
pedigreeData	optional data.frame where the first three columns are of type character and hold the identifiers for individuals, their mothers and their fathers (in that order)

Value

an instance of class DataQuality

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

`DataQuality_verify` *Verifies the quality of the provided phenotypic, genotypic and pedigree data*

Description

Verifies the quality of the provided data

Arguments

`phenoData`	optional data.frame where the first column is of type character and contains individual identifiers and one or more numeric columns containing the phenotypic observations
`genoData`	optional numeric matrix containing marker scores, individuals in rows and markers in columns
`pedigreeData`	optional data.frame where the first three columns are of type character and hold the identifiers for individuals, their mothers and their fathers (in that order)
`timeColumn`	optional identifier of a column in the phenoData data.frame that holds a unit of time for the observation. If phenoData is complemented with pedigreeData an animal model is fitted to the observations and the timeColumn is used to estimate the rate of genetic progress. Parameter timeColumn is either of type numeric indicating the column number or of type character indicating the column name. The column to which timeColumn refers should be either of type POSIXct or type numeric.

Value

list containing data quality measures

`DataQuality_verifyGenoDataQuality` *Verifies the quality of the provided genotypic data*

Description

Verifies the quality of the provided genotypic data

Arguments

`genoData`	numeric matrix containing marker scores, individuals in rows and markers in columns

Value

list containing genotypic data quality statistics

`DataQuality_verifyPedigreeConsistency`
Verifies the consistency of the provided pedigree data

Description

Verifies the consistency of the provided pedigree data

Arguments

`pedigreeData`	data.frame where the first three columns are of type character and hold the identifiers for individuals, their mothers and their fathers (in that order)

Value

list containing pedigree consistency status

`DataQuality_verifyPhenoDataQuality`
Verifies the quality of the provided phenotypic data

Description

Verifies the quality of the provided phenotypic data by calculating various quality statistics

Arguments

`phenoData`	data.frame where the first column is of type character and contains identifiers of the individuals and one or more numeric columns containing the phenotypic observations of one or more traits

Value

list containing phenotypic data quality statistics for each trait

genoImputation *GenoImputation constructor*

Description

Creates an instance of class GenoImputation

Usage

```
genoImputation(loadingScoreMatrix, markerPositions)
```

Arguments

loadingScoreMatrix	matrix of marker scores / loadings with individuals in rows and markers in columns. In case loadingScoreMatrix is of type character each cell in the matrix should be a string of size 2 containing the two alleles of the marker. In case loadingScoreMatrix is of type numeric each cell in the matrix is the frequency of the reference allele as a number between 0 and 2. optionally the matrix can have refAlleles and altAlleles attributes providing the reference and alternative alleles for each marker
markerPositions	data.frame having three columns in the following order: a character column holding the names of the markers which should match the column names of the scoreMatrix or loadingMatrix argument an numeric column holding the linkage group (i.e. chromosome) of each marker a numeric column holding the physical position on the linkage group for each marker

Value

an instance of class GenoImputation

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

---

genoImputation_getImputedLoadingMatrix
Retrieve imputation result as a numeric matrix of marker allele scores

Description

Retrieve imputation result as a numeric matrix of marker allele scores

Value

matrix of type numeric containing imputed marker allele loadings (i.e. frequency of the reference allele as a number between 0 and 2)

genoImputation_getImputedScoresInTripleFormat
Retrieve imputed marker scores as a data.frame of three columns (generalID, markerNr, score)

Description

Retrieve imputed marker scores as a data.frame of three columns (generalID, markerNr, score)

Value

data.frame of three columns

genoImputation_impute *Impute missing marker scores*

Description

Imputes missing marker scores by means of FImpute

Arguments

loadingScoreMatrix	matrix of marker scores / loadings with individuals in rows and markers in columns. In case loadingScoreMatrix is of type character each cell in the matrix should be a string of size 2 containing the two alleles of the marker. In case loadingScoreMatrix is of type numeric each cell in the matrix is the frequency of the reference allele as a number between 0 and 2. optionally the matrix can have refAlleles and altAlleles attributes providing the reference and alternative alleles for each marker
markerPositions	data.frame of three columns containing the physical position of each marker
method	character vector containing the name of the imputation engine, either 'fimpute', 'fastphase' or 'beagle'
threads	number of processing threads used for calculations
accuracyIterations	number of iterations used to estimate imputation error rate and accuracy. At each iteration scores of randomly selected positions of the imputation result are masked in a way that the mimmics the pattern of missing scores in the input matrix. If this parameter is set to 0, imputation error and accuracy are not estimated.
detailedIterationOutput	logical indicating if the results of all iterations of the imputation accuracy estimation procedure need to be returned.
keepOutput	logical indicating if the output of the imputation software should be returned

Value

in case accuracyIterations=0, the scoreMatrix with missing scores replaced by imputed scores. In case accuracyIterations>0, a list containing the imputed matrix, the estimated imputation error rate and accuracy

---

genoImputation_transcodeLoadingsToScores

Transcode numeric marker allele loadings to character marker scores

---

Description

Transcode marker scores from numeric to character representation

Arguments

loadingMatrix	numeric matrix containing marker allele loadings (i.e. frequency of the reference allele as a number between 0 and 2), individuals in rows and markers in columns reference alleles and alternative alleles can be provided as attributes refAlleles and altAlleles respectively

Value

matrix of type character containing marker scores

---

genoImputation_transcodeScoresToLoadings

Transcode character marker allele scores to numeric marker allele loadings

---

Description

Transcode character marker scores fto numeric marker allele loadings

Arguments

scoreMatrix	character matrix containing marker allele scores (i.e. two-character representation of diploid score), individuals in rows and markers in columns reference alleles and alternative alleles can be provided as attributes refAlleles and altAlleles respectively

Value

matrix of type numeric containing marker allele loadings (i.e. frequency of the reference allele as a number between 0 and 2)

genomicPrediction *GenomicPrediction constructor*

Description

Creates an instance of class GenomicPrediction

Usage

```
genomicPrediction(phenoData, genoData)
```

Arguments

phenoData	optional data.frame where one column is of type character and contains identifiers of the individuals and one or more numeric columns contain the phenotypic observations of one or more traits
genoData	optional numeric matrix containing marker scores, individuals in rows and markers in columns

Value

an instance of class PhenoAnalysis

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

GenomicPrediction_consensusPredict *Make consensus predictions using two or more genomic prediction models*

Description

Make genomic predictions by means two or more trained prediction models

Arguments

genoData	numeric matrix containing marker scores, individuals in rows and markers in columns
predictionModels	list containing two or more trained genomic prediction models as produced by GenomicPrediction$trainGenomicPredictionModel
weights	optional numeric vector containing a weight for each prediction model in predictionModels argument
fixedMarkers	optional numeric matrix, containing scores for a limited set of markers that will be fitted as fixed effects, only used when predictionMethod equals 'GBLUP'
threads	number of processing threads

Value

numeric vector containing genomic predictions of the individuals in genoData

---

`GenomicPrediction_extractScalingData`

Extracts the input phenotypes and matching genomic predictions from a trained genomic prediction model

---

Description

Generates a data.frame that contains the input phenotypic observations and matching genomic predictions to assess scaling differences

Arguments

`predictionModel`

list containing a trained genomic prediction model created by the function GenomicPrediction$trainGenomicPredictionModel

Value

data.frame containing the input phenotypic observations and matching genomic predictions

---

`GenomicPrediction_getPredictionModelStats`

Retrieves some basic statistics for a genomic proediction models

---

Description

Retrieves basic statistics such as training set size, number of markers, ... for a genomic prediction model

Arguments

`predictionModel`

list containing a trained genomic prediction model created by the function GenomicPrediction$trainGenomicPredictionModel

Value

list containing statistics

GenomicPrediction_isValidPredictionModel

Verifies if the provided genomic prediction model is structurally valid

Description

Verifies if the provided genomic prediction model is structurally valid

Arguments

predictionModel	list containing a trained genomic prediction model created by the function GenomicPrediction$trainGenomicPredictionModel

Value

logical indicating if the genomic prediction model is structurally sane

GenomicPrediction_makeManhattanPlot

Creates a Manhattan plot from a GWAS result

Description

Generates a Manhattan plot from a GWAS result

Arguments

gwasResult	data.frame with four columns: 1. character column containing the names of the markers, these marker names should appear as column names of the genoData matrix, 2. integer column containing the chromosome of the markers 3. numeric column containing the position on the chromosome (in bp or cM) of the markers 4. numeric column containing the GWAS scores for the trait
fdrLevel	numeric containing the false discovery rate at which the markers effects should be tested

Value

the plotly-based Manhattan plot

GenomicPrediction_makeQQPlot

Creates a QQ plot from a GWAS result

Description

Generates a QQ plot from a GWAS result

Arguments

gwasResult	data.frame with four columns: 1. character column containing the names of the markers, these marker names should appear as column names of the genoData matrix, 2. integer column containing the chromosome of the markers 3. numeric column containing the position on the chromosome (in bp or cM) of the markers 4. numeric column containing the GWAS scores for the trait

Value

the plotly-based QQ plot

GenomicPrediction_performGWAS

Performs a GWAS analysis

Description

Performs a GWAS analysis using the QK linear mixed model approach described by Yu et al. (2006)

Arguments

phenoData	optional data.frame where the first column is of type character and contains identifiers of the individuals and one or more numeric columns containing the phenotypic observations of one or more traits
genoData	optional numeric matrix containing marker frequencies between 0 and 1, individuals in rows and markers in columns
markerInfo	optional data.frame with three columns: 1. character column containing the names of the markers, these marker names should appear as column names of the genoData matrix, 2. integer column containing the chromosome of the markers 3. numeric column containin the position on the chromosome (in bp or cM) of the markers
indColumn	optional character or integer vector of length 1 indicating the name or position of the column that contains the identifiers of the individuals. If missing it is assumed that the first character column contains the individuals.

traitColumns	optional character or integer vector indicating the columns that will be used as dependent variables in the GWAS analysis. If traitColumns is numeric the dependent columns are identified by their position in the phenoData data.frame. If traitColumns is of type character it is assumed to hold the names of the trait columns in the phenoData data.frame. If traitColumns is not provided, all numeric columns in phenoData will be analyzed.
fixedEffectColumns	optional character or integer vector indicating the columns that will be fitted as fixed effects (e.g. population structure) in the linear mixed model. If fixedEffectColumns is numeric the dependent columns are identified by their position in the phenoData data.frame. If fixedEffectColumns is of type character it is assumed to hold the names of the columns in the phenoData data.frame. If fixedEffectColumns is not provided, no fixed effects are fitted in the model.
minMAF	Minor Allele Frequency threshold, defaults to 0.05
reuseVar	logical indicating if the required genetic variance parameter should be estimated once (TRUE) or for each marker seperately (FALSE, slower)
threads	number of processing threads

Value

data.frame containing the scores (-log10 p-values) for each combination of marker and trait

---

GenomicPrediction_predict

Make genomic predictions

Description

Make genomic predictions by means of a trained prediction model

Arguments

genoData	numeric matrix containing marker scores, individuals in rows and markers in columns
predictionModel	list containing a trained genomic prediction model as produced by GenomicPrediction$trainGenomicPredictionModel
fixedMarkers	optional numeric matrix, containing scores for a limited set of markers that will be fitted as fixed effects, only used when predictionMethod equals 'GBLUP'
allowNA	logical indicating if NA's are allowed in the score matrix in which case they are replaced by the average marker score, defaults to TRUE
threads	number of processing threads

Value

numeric vector containing genomic predictions of the individuals in genoData

`GenomicPrediction_trainGenomicPredictionModel`
Train a genomic prediction model

Description

Train a genomic prediction model

Arguments

`phenoData`	optional data.frame where the first column is of type character and contains identifiers of the individuals and one or more numeric columns containing the phenotypic observations of one or more traits
`genoData`	optional numeric matrix containing marker scores, individuals in rows and markers in columns
`fixedMarkers`	optional numeric matrix, containing scores for a limited set of markers that will be fitted as fixed effects, only used when predictionMethod equals 'GBLUP'
`indColumn`	optional character or integer vector of length 1 indicating the name or position of the column that contains the identifiers of the individuals. If missing it is assumed that the first character column contains the individuals.
`traitColumns`	optional character or integer vector indicating the columns that will be used as dependent variables to train genomic prediction models. If traitColumns is numeric the dependent columns are identified by their position in the phenoData data.frame. If traitColumns is of type character it is assumed to hold the names of the trait columns in the phenoData data.frame. If traitColumns is not provided, all numeric columns in phenoData will be analyzed.
`predictionMethod`	character vector indiciating the genomic prediction method, one of 'RRBLUP', 'GBLUP' or 'RandomForest', defaults to 'RRBLUP'
`threads`	number of processing threads
`accuracyIterations`	number of iterations used to estimate genomic prediction accuracy.
`detailedIterationOutput`	logical indicating if the results of all iterations of the prediction accuracy estimation procedure need to be returned.
`removeRedundantMarkers`	logical indicating if redundant columns in the genoMatrix should be removed prior to analysis
`...`	other arguments passed to the genomic prediction methods

Value

list containing genomic prediction results

genomicSelection *GenomicSelection constructor*

Description

Creates an instance of class GenomicSelection

Usage

```
genomicSelection(
  predictionModels,
  weights,
  goals,
  descriptors,
  genoData,
  geneticMap
)
```

Arguments

predictionModels	optional list containing one or more trained genomic prediction models
weights	optional numeric vector containing weights that indicate the relative importance of each of the genomic prediction models
goals	optional vector containing for each prediction model the desired numerical target value or the character values 'min' or 'max' indicating that the trait is to be minimized or maximized respectively
descriptors	character vector containing a description for each of the genomic prediction models
genoData	optional numeric matrix containing marker loadings fro the selection candidates, individuals in rows and markers in columns
geneticMap	data.frame containing genetic map information. The first column should contain marker numbers, the second linkage group of each marker and the third column should contain the position of the marker (in cM)

Value

an instance of class PhenoAnalysis

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

GenomicSelection_makeComplementaryLinePredictions

Identifies inbred line combinations that after after a process of intercrossing and consecutive selfing have maximum probability of producing hybrids with superior phenotypic performance

Description

Inbred lines are pairwise crossed after which a sample of the possible inbred lines that can originate from consecutive selfing of the F1 is generated. For each initial cross the top performing simulated inbred lines are identified by means of genomic prediction models that are trained to predict line performance (i.e. general combining abilities). These top performing lines are intermated to create hybrids for which the phenotypic performance is predicted with genomic prediction models that are trained to predict hybrid performance.

Arguments

linePredictionModels	list containing one or more genomic prediction models that are trained to predict line performances (i.e. general combining abilities)
hybridPredictionModels	list containing one or more genomic prediction models that are trained to predict hybrid performances. The number and order of these hybrid prediction models should match the order of the prediction models of the linePredictionModels argument (i.e. line and hybrid prediction models should predict the same traits)
weights	optional numeric vector containing weights that indicate the relative importance of each of the genomic prediction models, if not provided weights are assumed equal. These weights are used to calculate index values for both line and hybrid predictions.
goals	vector containing for each prediction model the desired numerical target value or the character values 'min' or 'max' indicating that the trait is to be minimized or maximized respectively. These goals are are used to calculate index values for both line and hybrid predictions.
descriptors	optional character vector containing a description for each of the genomic prediction models (shared by line and hybrid prediction models)
genoData	optional numeric matrix containing marker loadings for the parental individuals, individuals in rows and markers in columns
parentPairs	optional data.frame containing two integer columns or two character columns.Each row represents a pair of parents that will be crossed to create an inbred line population. If the two columns are numeric they refer to the row indexes of the genoData argument. If they are of type character the two columns should contain generalIDs
excludeSelfings	logical indicating if selfing of parents is to be excluded as an initial cross. This situation could occur if the entries in the two columns of the parentPairs data.frame overlap. Defaults to TRUE

`geneticMap` data.frame containing genetic map information. The first column should contain marker numbers, the second linkage group of each marker and the third column should contain the position of the marker (in cM)

`selfingGenerations` integer number indicating the number of consecutive selfings that is used to obtain each inbred line population. Defaults to 4 (i.e. creates F5 lines)

`popSize` integer number indicating the size of each inbred line population. Defaults to 1000, should be at least 100

`selectionPercentage` numeric between 5 and 95 indicating the percentage of the simulated line population that is selected as the top performers. Defaults to 10 implying that if the popSize is 1000 the top 100 performing inbred lines will be selected for virtual hybrid creation.

`threads` numeric indicating the number of processing threads used for the calculations, defaults to 1

Value

data.frame containing the average rank for each four-way parental line combination. A lower rank implies that the virtual hybrids that originate from these four initial parents have a better performance.

GenomicSelection_makeHybridPredictions

Predicts the phenotypic performance of hybrids from the marker loadings of parental inbred lines

---

Description

Predicts the phenotypic performance of hybrids for one or more traits and calculates a selection index

Arguments

`predictionModels` list containing one or more genomic prediction models that are trained to predict hybrid performances

`weights` optional numeric vector containing weights that indicate the relative importance of each of the genomic prediction models, if not provided weights are assumed equal

`goals` vector containing for each prediction model the desired numerical target value or the character values 'min' or 'max' indicating that the trait is to be minimized or maximized respectively

`descriptors` optional character vector containing a description for each of the genomic prediction models

genoData — optional numeric matrix containing marker loadings for the parental individuals, individuals in rows and markers in columns

parentPairs — optional data.frame containing two integer columns or two character columns.Each row represents a pair of parents that will be crossed to create a hybrid. If the two columns are numeric they refer to the row indexes of the genoData argument. If they are of type character the two columns should contain generalIDs of the parental lines

excludeSelfings — logical indicating if selfing of parents is to be excluded. This situation could occur if the entries in the two columns of the parentPairs data.frame overlap. Defaults to TRUE

threads — numeric indicating the number of processing threads used for the calculations, defaults to 1

Value

data.frame containing the requested quantiles of the distributions of genomic predictions for each hybrid and a selection index in case there is more than one prediction model

---

GenomicSelection_makeInitialCrossPredictions

Predicts the phenotypic performance of inbred line populations that originate from recurrent selfing of parent pair crosses

---

Description

Predicts the phenotypic performance of inbred line populations for one or more traits and calculates a selection index

Arguments

predictionModels — list containing one or more genomic prediction models that are trained to predict line performances (i.e. general combining abilities)

weights — optional numeric vector containing weights that indicate the relative importance of each of the genomic prediction models, if not provided weights are assumed equal

goals — vector containing for each prediction model the desired numerical target value or the character values 'min' or 'max' indicating that the trait is to be minimized or maximized respectively

descriptors — optional character vector containing a description for each of the genomic prediction models

genoData — optional numeric matrix containing marker loadings for the parental individuals, individuals in rows and markers in columns

`parentPairs` optional data.frame containing two integer columns or two character columns.Each row represents a pair of parents that will be crossed to create an inbred line population. If the two columns are numeric they refer to the row indexes of the genoData argument. If they are of type character the two columns should contain generalIDs

`excludeSelfings` logical indicating if selfing of parents is to be excluded as an initial cross. This situation could occur if the entries in the two columns of the parentPairs data.frame overlap. Defaults to TRUE

`geneticMap` data.frame containing genetic map information. The first column should contain marker numbers, the second linkage group of each marker and the third column should contain the position of the marker (in cM)

`selfingGenerations` integer number indicating the number of consecutive selfings that is used to obtain each inbred line population. Defaults to 4 (i.e. creates F5 lines)

`popSize` integer number indicating the size of each inbred line population. Defaults to 1000, should be at least 100

`popPercentile` numeric between 0 and 100 indicating the percentile of the population that will be used to calculate the selection index. Defaults to 20 implying that 80 percent of the population will have a performance that is better than the predicted value.

`threads` numeric indicating the number of processing threads used for the calculations, defaults to 1

Value

data.frame containing the requested quantiles of the distributions of genomic predictions for each parent pair and a selection index in case there is more than one prediction model

---

`GenomicSelection_makePredictions`

Predicts the phenotypic performance of selection candidates for one or more traits and calculates a selection index

---

Description

Predicts the phenotypic performance of selection candidates for one or more traits and calculates a selection index

Arguments

`predictionModels` list containing one or more trained genomic prediction models

`weights` optional numeric vector containing weights that indicate the relative importance of each of the genomic prediction models, if not provided weights are assumed equal

goals	vector containing for each prediction model the desired numerical target value or the character values 'min' or 'max' indicating that the trait is to be minimized or maximized respectively
descriptors	optional character vector containing a description for each of the genomic prediction models
genoData	optional numeric matrix containing marker loading for the selection candidates, individuals in rows and markers in columns
threads	numeric indicating the number of processing threads used for the calculations, defaults to 1

Value

data.frame containing the genomic predictions for each individual in genoData and a selection index in case there is more than one prediction model

---

GenomicSelection_makeSegregatingLinePredictions

Predicts the phenotypic performance of inbred line populations that originate from recurrent selfing of segregating lines (e.g. F3)

Description

Predicts the phenotypic performance of inbred line populations for one or more traits and calculates a selection index. To sample a population of inbred lines from a start individual (e.g. an F3) the phase of this start individual has to be known. this means that for each herozygous locus of the start individual it should be known which allele has been inherited from the maternal parent and which allele has been inherited from the paternal parent. The phase of each start individual can often be deduced from the genotypic scores of the parents. If these parental genotypes are unknown or the allelic configurations of the parents are ambigious (e.g. both parents are heterozygous at the given locus), the two possible phase configurations are sampled at equal probability.

Arguments

predictionModels	list containing one or more trained genomic prediction models
weights	optional numeric vector containing weights that indicate the relative importance of each of the genomic prediction models, if not provided weights are assumed equal
goals	vector containing for each prediction model the desired numerical target value or the character values 'min' or 'max' indicating that the trait is to be minimized or maximized respectively
descriptors	optional character vector containing a description for each of the genomic prediction models

genoData	optional numeric matrix containing marker loadings for the start individuals and, if available, the loadings of the parental individuals (i.e. the two parents of the initial cross from which a start individual has been derived). Individuals are in rows and markers in columns
pedigree	data.frame containing three character columns. The first column contains the generalIDs of the start individuals for which inbred line populations will be sampled. The second column contains the generalID of the maternal parent of the initial cross from which the start individual has been derived. The third column contains the generalID of the paternal parent. Entries in the second and third columns should be NA if the parents are unknown or do not appear in the genoData matrix.
geneticMap	data.frame containing genetic map information. The first column should contain marker numbers, the second linkage group of each marker and the third column should contain the position of the marker (in cM)
selfingGenerations	integer number indicating the number of consecutive selfings that is used to obtain each inbred line population. Defaults to 3 (e.g. if the start individuals are F3's the simulated inbred line populations will be F6)
popSize	integer number indicating the size of each inbred line population. Defaults to 1000, should be at least 100
popPercentile	numeric between 0 and 100 indicating the percentile of the population that will be used to calculate the selection index. Defaults to 20 implying that 80 percent of the population will have a performance that is better than the predicted value.
threads	numeric indicating the number of processing threads used for the calculations, defaults to 1

Value

data.frame containing the genomic predictions for each individual in the first column of the pedigree argument and a selection index in case there is more than one prediction model

---

importXlsx *XLSX file import*

Description

Imports data from an XLSX file

Usage

```
importXlsx(filePath, sheetNr)
```

Arguments

filePath	character variable holding the path to the XLSX file
sheetNr	optional numeric indicating the sheet to import

Value

a data.frame of the sheet or a list of data.frames in case the file has multiple sheets and no sheetNr was provided

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

---

jobFailed *Internal function that is called when a job fails*

Description

Internal function that is called when a job fails

Usage

```
jobFailed(errorValue, jobFuture, jobId, logId, outFile)
```

---

jobQueue *JobQueue constructor*

Description

Creates an instance of class JobQueue

Usage

```
jobQueue()
```

Value

an instance of class JobQueue

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

JobQueue_getJobLogEntries *Get the log entries of a job*

Description

Returns the available entries in the registry log that are related to a particular job

Arguments

jobId	character vector containing the ID of the job

Value

a data.frame with rows ordered in ascending log timestamp

JobQueue_getJobResult *Get the job result*

Description

Returns the output and result of a job that has been processed

Arguments

jobId	character vector containing the ID of the job

Value

list containing an output character vector and an R object that is the result of the job

JobQueue_getJobStatus *Get the status of a job*

Description

Returns the current status of a job

Arguments

jobId	character vector containing the ID of the job

Value

character vector containing one of "submitted", queued", "processing", "ready", "failed" if the jobID is found, NULL otherwise

JobQueue_getJobType *Get the type of a job*

Description

Returns the job type

Arguments

jobId	character vector containing the ID of the job

Value

character vector containing the type if the jobID is found, NULL otherwise

JobQueue_getLastJobStatus *Get the status of the last submitted job*

Description

Returns the status of the last submitted job of a particular type

Arguments

jobType	character vector containing the type of the job

Value

list containing the last jobID and its status ("submitted", queued", "processing", "ready", "failed") if a job is found, NULL otherwise

JobQueue_listJobs *Lists the jobs in the queue*

Description

Returns a list of queued jobs

Arguments

query	optional list specifying a query on the metadata. Example list(metadata.status="queued")

Value

a data.frame containing the metadata of the tasks in the queue

JobQueue_processJobs *Processes jobs in the queue*

Description

Checks the queue for pending jobs and processes them

Arguments

strategy	character vector containing the queuing strategy. Used by the future::plan function. Defaults to "multiprocess"
workers	integer indicating the number of simultaneous worker threads, defaults to 1. In case of strategy="cluster" workers can also be a character vector of computation node adresses.
returnWhenDone	logical indicating if the function should return when all jobs have been processed or wait for other jobs, defaults to TRUE
queryInterval	numeric only used when returnWhenDone=FALSE, gives the time in seconds between registry requests for new jobs, defaults to 5 seconds
restartActiveJobs	logical indicating if at the start of the routine jobs with status "processing" (i.e. jobs that were not completed in a previous run of the routine) should be restarted. Defaults to TRUE

JobQueue_removeJob *Remove a job from the queue*

Description

Removes a job from the queue

Arguments

jobId	character vector containing the ID of the job

Value

logical indicating if removal was successful or not

JobQueue_submitJob *Submit a job to the queue for ansynchronous processing*

Description

Adds a job to the processing queue

Arguments

expr	expression containing the R code that is to be executed as a separate, asynchronuous process.
exprStr	optional character vector containing the R code as a character vector, if provided the expr parameter will be ignored
type	optional character vector containing the job type, defaults to 'job'
envir	environment in which the variables that are referred to in expr can be found, defaults to the calling environment.
variables	optional character vector containing the names of the variables that need to be retrieved from envir or a named list containing these variables. This parameter is usually not required as variables are identified automatically from the provided expr.
packages	optional character vector containing the names of the packages that are required to run expr. This parameter is usually not required as packages are identified automatically from the provided expr.
description	optional character vector containing a description of the task

Value

character vector containing the jobID if the job has been successfully submitted or NULL otherwise

jobSuccess *Internal function that is called when a job finishes successfully*

Description

Internal function that is called when a job finishes successfully

Usage

```
jobSuccess(result, jobFuture, jobId, logId, outFile)
```

logRegistry *LogRegistry constructor*

Description

Creates an instance of class LogRegistry

Usage

```
logRegistry(user, logDescription, logID)
```

Arguments

user	character variable holding the user's name
logDescription	optional character vector describing the logging instance
logID	optional character vector providing the ID of the log

Value

an instance of class LogRegistry#'

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

LogRegistry_getFunctionStatistics *Get PGSP usage statistics for one or more users*

Description

Get usage statistics like the number of users, warnings and errors for one or more PGSP functions

Arguments

functions	optional character vector containing one or more function name for which usage statistics are requested, all logged functions will be reported when missing

Value

a data frame containing the usage statistics

LogRegistry_getFunctionUsage *Get PGSP functions that have been used by a particual user*

Description

Get PGSP functions that have been used by a particual user

Arguments

username	optional character vector containing the user for which function usage is requested

Value

a data frame listing the PGSP functions called by the user

LogRegistry_getLogEntries *Retrieve log entries*

Description

Retrieves the log entries as a data.frame with rows ordered according to the log timestamp

Value

a data.frame with rows ordered in ascending log timestamp

LogRegistry_getLogID *Get the ID of the log*

Description

Get the ID of the log

Value

a character vector containing the ID of the log

`LogRegistry_getLogList`

Lists the available logIDs and their description

Description

Retrieves a list of logIDs and their description

Arguments

`allUsers`	logical, if TRUE logs from all users will be retrieved, defaults to FALSE

Value

a data.frame containing logID, description and initial log entry timestamp

---

`LogRegistry_getUserStatistics`

Get PGSP usage statistics for one or more users

Description

Get usage statistics like the last login, number of sessions, warnings, errors

Arguments

`usernames`	optional character vector containing one or more usernames for which usage statistics are requested, all users will be reported when missing

Value

a data frame containing the usage statistics

LogRegistry_log *Add an entry to the log*

Description

Adds a log entry

Arguments

message	character vector containing the log message
level	log level, character vector containing one of INFO, WARN, ERROR, CRITICAL, defaults to INFO
fnName	optional character vector naming the function to which the log entry refers

Value

logical indicating if the log message has been successfully written to the database backend

LogRegistry_setLogID *Set the ID of the log*

Description

Set the ID of the log

Arguments

logID	character vector containing the ID of the log

modelFactory *ModelFactory constructor*

Description

Creates an instance of class ModelFactory

Usage

```
modelFactory()
```

Value

an instance of class ModelFactory

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

`ModelFactory_getMultiTrialModel`

Get the model definition for a multi-trial analysis

Description

Retrieves the linear mixed model model definition for an analysis over multiple trials

Arguments

`pdpTypeID`	integer indicating the pdpTypeID for which the model definition is to be retrieved
`traitName`	character vector containing the names of the trait for which the model definition is to be retrieved
`trialIDs`	integer containing the IDs of the trials for which the combined model definition is to be retrieved

Value

charactar vector containing the linear mixed model definition as an XML

`ModelFactory_getSingleTrialModel`

Get the model definition for a trial and trait

Description

Retrieves the linear mixed model model definition for a trial / trait combination

Arguments

`pdpTypeID`	integer indicating the pdpTypeID for which the model definition is to be retrieved
`traitName`	character vector containing the names of the trait for which the model definition is to be retrieved
`trialID`	integer containing the IDs of the trial for which the model definition is to be retrieved

Value

charactar vector containing the linear mixed model definition as an XML

`ModelFactory_getTrialModelInfo`

Get information on available trial models

Description

Retrieves information on the available trial models

Arguments

`pdpTypeIDs`	optional integer vector containing the pdpTypeIDs for which the model information is to be retrieved
`trialIDs`	optional integer vector containing the IDs of the trials for which the model information is to be retrieved
`traitNames`	optional character vector containing the names of the traits for which the model information is to be retrieved
`collectionName`	character vector containing the name of the collection, defaults to TrialModels

Value

data.frame containing information on the requested trial models

`ModelFactory_uploadMultiTrialModel`

Upload the model definition for a multi-trial analysis

Description

Takes a model definition for the analysis of a trait over multiple trials, extracts the base model and single trial models and updates these models in the database

Arguments

`pdpTypeID`	integer indicating the pdpTypeID for which the model definition is to be retrieved
`traitName`	character vector containing the names of the trait for which the model definition is to be retrieved
`modelDefinition`	character vector containing the multi trial linear mixed model definition as an XML

ModelFactory_uploadSingleTrialModel

Upload the model definition for a single trial and trait combination

Description

Replaces a trial model in the database

Arguments

pdpTypeID	integer indicating the pdpTypeID for which the model definition is to be retrieved
traitName	character vector containing the names of the trait for which the model definition is to be retrieved
trialID	integer the ID of the trial
modelDefinition	character vector containing the trial linear mixed model definition as an XML

pdpFactory

PDPFactory constructor

Description

Creates an instance of class PDPFactory

Usage

```
pdpFactory()
```

Value

an instance of class DataPhenoFactory

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

PDPFactory_addEstimatesToPDP
Add trait estimates to a PDP

Description

Adds breeding values and other estimates to a PDP that is under construction

Arguments

pdpID	numeric containing the ID of the PDP
traitName	character vector containing the name of the trait (e.g. yield)
traitDescription	optional character vector containing a description of the trait
traitUnit	optional character vector containing a description of the unit in which the trait is expressed
traitObjective	either a character vector containing 'minimize' or 'maximize' or a numeric target value
target	character vector describing the type of estimate, usually one of EBV (Estimated Breeding Value) or GV (Genetic Value)
prediction	character vector describing the prediction/estimation type, either pedigree or genomic
segment	optional character vector describing the part of the breeding pool for which the estimates are provided (e.g. early)
isBreedingValue	logical indicating if the provided estimates are breeding values and should be maide available for the Prediction&Advice module, deafults to FALSE
intercept	numeric value providing the intercept for the linear rescaling of the estimates (defaults to 0)
slope	numeric value providing the slope for the linear rescaling of the estimates (defaults to 1)
estimates	data.frame containing the estimates/predictions. This data.frame should have 2 or 3 columns. The first column is of type character and contains the identifiers of the individuals. The second column is of type numeric and contains the estimates/predictions. If a third numeric column is present it is assumed that it holds the reliabilities of the estimates.

Value

locagical indicating if the addition of estimates was successful

PDPFactory_addGenomicPredictionModelToPDP
Add genomic prediction model to a PDP

Description

Adds a genomic prediction model to a PDP that is under construction

Arguments

pdpID	numeric containing the ID of the PDP
traitName	character vector containing the name of the trait (e.g. yield)
traitDescription	optional character vector containing a description of the trait
traitUnit	optional character vector containing a description of the unit in which the trait is expressed
traitObjective	either a character vector containing 'minimize' or 'maximize' or a numeric target value
target	character vector describing the type of estimate, either EBV (Estimated Breeding Value) or GV (Genetic Value)
segment	optional character vector describing the part of the breeding pool for which the estimates are provided (e.g. early)
intercept	numeric value providing the intercept for the linear rescaling of the estimates (defaults to 0)
slope	numeric value providing the slope for the linear rescaling of the estimates (defaults to 1)
predictionModel	an RRBLUP genomic prediction model produced by as produced by the GenomicPrediction class
estimates	optional data.frame containing genomic predictions. This data.frame should have 2 or 3 columns. The first column is of type character and contains the identifiers of the individuals. The second column is of type numeric and contains the estimates/predictions. If a third numeric column is present it is assumed that it holds the reliabilities of the estimates.

Value

locagical indicating if the addition of estimates was successful

PDPFactory_analyzePDP *Performs the phenotypic and genotypic analysis for a PDP*

Description

Performs both phenotypic and genotypic analysis on the provided traits, trials and markers

Arguments

pdpID	numeric containing the ID of the PDP that is to be analyzed
traits	optional character vector containing the names of the traits that should be analysed
trialIDs	optional integer vector containing the trialIDs fthat should be considered
markerIDs	optional integer vector containing the markerIDs that should be considered
dropExisting	logical if TRUE traits that have already been analyzed will be analyzed again, defaults to FALSE

PDPFactory_createNewPDP
Create a new Processed Data Package (PDP) in Progeno

Description

Creates a processed data package (PDP) that can be viewed in Progeno

Arguments

name	character vector containing the name of the PDP
pdpTypeID	integer vector containing the type of the PDP, defaults to 1
traits	optional character vector containing the names of the traits that should be analysed, missing implies all available traits
trialIDs	optional integer vector containing the trialIDs fthat should be considered, if missing, all available and connected trials are used
markerIDs	optional integer vector containing the markerIDs that should be considered, mising implies all available markers will be used

Value

integer containing the unique ID of the PDP

PDPFactory_getPDPInfo *Get available PDPs from the database.*

Description

Retrieves the available PDPs and their metadata

Arguments

collectionName	character vector containing the name of the collection, defaults to PDPs

Value

data.frame containing information on the available PDPs

PDPFactory_listEstimates *List the available estimates for a PDP*

Description

Lists the estimates contained in a PDP

Arguments

pdpID	numeric containing the ID of the PDP

PDPFactory_listGenomicPredictionModels *List genomic prediction models of a PDP*

Description

Lists the genomic prediction models of a PDP

Arguments

pdpID	numeric containing the ID of the PDP

PDPFactory_publishPDP *Publish a PDP*

Description

make a PDP available for other Progeno users by publishing it, requires administrative privileges

Arguments

pdpID	numeric containing the ID of the PDP that is to be published

PDPFactory_removePDP *Removes a PDP from the Progeno database*

Description

Removes a PDP from the Progeno database

Arguments

pdpID	numeric containing the ID of the PDP that is to be removed

PDPFactory_retractPDP *Retract a published PDP*

Description

make a published PDP unavailable for other Progeno users by retracting its publication

Arguments

pdpID	numeric containing the ID of the PDP that is to be retracted

PGSPBase-class *Base class for all PGSP classes*

Description

Base class for all PGSP classes

phenoAnalysis *PhenoAnalysis constructor*

Description

Creates an instance of class PhenoAnalysis

Usage

```
phenoAnalysis(phenoData, pedigreeData)
```

Arguments

phenoData	optional data.frame containing at least one column of type character which contains identifiers of the individuals and one or more numeric columns containing the phenotypic observations of one or more traits
pedigreeData	pedigreeData optional data.frame where the first three columns are of type character and hold the identifiers for individuals, their mothers and their fathers (in that order)

Value

an instance of class PhenoAnalysis

Author(s)

Steven Maenhout, <Steven.Maenhout@progeno.net>

PhenoAnalysis_findMultiTrialModel
Find the best fitting linear mixed model for a set of trials

Description

For each trial all models in the provided metamodel are fitted to identify the best fitting model definition. The resulting models are combined in a single multitrial model

Arguments

metamodelDef	character vector containing the candidate model definitions in XML format
basemodelDef	character vector containing the base model definition for the multitrial model in XML format
trialData	data frame containing the trial data

`pedigreeData` data.frame where the first three columns are of type character and hold the identifiers for individuals (hybrids and parental lines), their mothers and their fathers (in that order) The pedigree data.frame can contain an optional fourth column containing the coefficient of inbreeding for each individual. This is a number between 0 and 1 where 0 implies that the parents of the individual are completely unrelated and 1 implies that the individual has been selfed for an infinite number of generations. Generally there is no need to provide this coefficient of inbreeding as it is derived from the pedigree structure (i.e. the first three columns of the data.frame). Only in the case where the coefficient of inbreeding can not be correctly deduced from the pedigree structure (e.g. in case of line creation by repeated selfing or doubled haploids) the coefficient of coancestry should be provided. The entries in this fourth column corresponding to individuals for which the coefficient of inbreeding should be deduced from the pedigree structure can be set to NA.

Value

character vector containing the best fitting model definition in XML format

---

`PhenoAnalysis_findTrialModel`
Find the best fitting linear mixed model for a trial

Description

Iteratively tries all models in the provided metamodel to identify the best fitting model definition for a single trial

Arguments

`metamodelDef` character vector containing the candidate model definitions in XML format

`trialData` data frame containing the trial data

`pedigreeData` data.frame where the first three columns are of type character and hold the identifiers for individuals (hybrids and parental lines), their mothers and their fathers (in that order) The pedigree data.frame can contain an optional fourth column containing the coefficient of inbreeding for each individual. This is a number between 0 and 1 where 0 implies that the parents of the individual are completely unrelated and 1 implies that the individual has been selfed for an infinite number of generations. Generally there is no need to provide this coefficient of inbreeding as it is derived from the pedigree structure (i.e. the first three columns of the data.frame). Only in the case where the coefficient of inbreeding can not be correctly deduced from the pedigree structure (e.g. in case of line creation by repeated selfing or doubled haploids) the coefficient of coancestry should be provided. The entries in this fourth column corresponding to individuals for which the coefficient of inbreeding should be deduced from the pedigree structure can be set to NA.

Value

character vector containing the best fitting model definition in XML format

---

`PhenoAnalysis_fitAnimalModel`

Fits an animal model to phenotypic and pedigree data

Description

Fits an animal model to one or more phenotypic traits and returns Estimated Breeding Values (EBVs) and variance component estimates

Arguments

`phenoData`	a data.frame where one column is a character vector containing the identifiers of the individuals and one or more numeric columns contain the phenotypic observations
`pedigreeData`	pedigreeData data.frame where the first three columns are of type character and hold the identifiers for individuals, their mothers and their fathers (in that order) The pedigree data.frame can contain an optional fourth column containing the coefficient of inbreeding for each individual. This is a number between 0 and 1 where 0 implies that the parents of the individual are completely unrelated and 1 implies that the individual has been selfed for an infinite number of generations. Generally there is no need to provide this coefficient of inbreeding as it is derived from the pedigree structure (i.e. the first three columns of the data.frame). Only in the case where the coefficient of inbreeding can not be correctly deduced from the pedigree structure (e.g. in case of line creation by repeated selfing or doubled haploids) the coefficient of coancestry should be provided. The entries in this fourth column correpsonding to individuals for which the coefficient of inbreeding should be deduced from the pedigree structure can be set to NA.
`indColumn`	optional character or integer vector of length 1 indicating the name or position of the column that contains the identifiers of the individuals. If missing it is assumed that the first character column contains the individuals.
`traitColumns`	optional character or integer vector indicating the columns that will be used as dependent variables in the linear mixed model analysis. If traitColumns is numeric the dependent columns are identified by their position in the phenoData data.frame. If traitColumns is of type character it is assumed to hold the names of the trait columns in the phenoData data.frame. If traitColumns is not provided, all numeric columns in phenoData will be analyzed.
`cofactorColumns`	optional character or integer vector indicating the columns that will be used as cofactors in the linear mixed model analysis. The columns in the phenoData data.frame that are referred to by the cofactorColumns argument will be converted into factors and fitted as fixed effects in the model.
`modelDefinition`	optional character vector containing the linear mixed model definition as an XML

Value

a list containing EBVs and estimated variance components

`PhenoAnalysis_fitHybridModel`

Fits a hybrid model to phenotypic and pedigree data

Description

Fits a hybrid model to one or more phenotypic traits and returns General Combining Abilities (GCAs), Specific Combining Abilities (SCAs) and variance component estimates

Arguments

`phenoData`	a data.frame where the first column is a character vector containing the identifiers of the hybrids and the remaining numeric columns containing the phenotypic observations
`pedigreeData`	pedigreeData data.frame where the first three columns are of type character and hold the identifiers for individuals (hybrids and parental lines), their mothers and their fathers (in that order) The pedigree data.frame can contain an optional fourth column containing the coefficient of inbreeding for each individual. This is a number between 0 and 1 where 0 implies that the parents of the individual are completely unrelated and 1 implies that the individual has been selfed for an infinite number of generations. Generally there is no need to provide this coefficient of inbreeding as it is derived from the pedigree structure (i.e. the first three columns of the data.frame). Only in the case where the coefficient of inbreeding can not be correctly deduced from the pedigree structure (e.g. in case of line creation by repeated selfing or doubled haploids) the coefficient of coancestry should be provided. The entries in this fourth column corresponding to individuals for which the coefficient of inbreeding should be deduced from the pedigree structure can be set to NA.
`traitColumns`	optional character or integer vector indicating the columns that will be used as dependent variables in the linear mixed model analysis. If traitColumns is numeric the dependent columns are identified by their position in the phenoData data.frame. If traitColumns is of type character it is assumed to hold the names of the trait columns in the phenoData data.frame. If traitColumns is not provided, all numeric columns in phenoData will be analyzed.
`cofactorColumns`	optional character or integer vector indicating the columns that will be used as cofactors in the linear mixed model analysis. The columns in the phenoData data.frame that are referred to by the cofactorColumns argument will be converted into factors and fitted as fixed effects in the model.
`modelDefinition`	optional character vector containing the linear mixed model definition as an XML

Value

a list containing EBVs and estimated variance components

`PhenoAnalysis_getLastModelDefinition`
Returns the mixed model definition of the last analysis run

Description

Returns the REML-optimized mixed model definition of the last analysis

Value

a character vector containing the model definition as an XML, character(0) is return if no model definition is available

---

`PhenoAnalysis_mixedModelAnalysis`
Analyzes a single trait by means of a linear mixed model

Description

Fits a linear mixed model to a single trait and returns BLUEs, BLUPs en and variance component estimates

Arguments

`phenoData`	a data.frame where the first column is a character vector containing the identifiers of the individuals and the remaining numeric columns containing the phenotypic observations
`pedigreeData`	pedigreeData data.frame where the first three columns are of type character and hold the identifiers for individuals, their mothers and their fathers (in that order) The pedigree data.frame can contain an optional fourth column containing the coefficient of inbreeding for each individual. This is a number between 0 and 1 where 0 implies that the parents of the individual are completely unrelated and 1 implies that the individual has been selfed for an infinite number of generations. Generally there is no need to provide this coefficient of inbreeding as it is derived from the pedigree structure (i.e. the first three columns of the data.frame). Only in the case where the coefficient of inbreeding can not be correctly deduced from the pedigree structure (e.g. in case of line creation by repeated selfing or doubled haploids) the coefficient of coancestry should be provided. The entries in this fourth column correpsonding to individuals for which the coefficient of inbreeding should be deduced from the pedigree structure can be set to NA.

indColumn	optional character or integer vector of length 1 indicating the name or position of the column that contains the identifiers of the individuals. If missing it is assumed that the first character column contains the individuals.
traitColumn	optional character or integer vector indicating the column that will be used as dependent variables in the linear mixed model analysis. If missing it is assumed that the first numeric column contains the trait values If traitColumns is numeric the dependent columns are identified by their position in the phenoData data.frame. If traitColumns is of type character it is assumed to hold the names of the trait columns in the phenoData data.frame.
modelDefinition	character vector containing the linear mixed model definition as an XML
returnFittedValues	logical indicating if the fitted values and residuals should be returned, defaults to false
threads	number of processing threads, defaults to 1
verbosity	integer vector containing a number from 1 to 3 expressing the desired level of verbosity of the analysis routines, defaults to 2

Value

a list containing BLUEs, BLUPs en and variance component estimates

---

PhenoAnalysis_showTrialConnectivity

Analyze connectivity of trials

Description

Show the connection between trials. Two trials are connected if they have one or more accessions in common

Arguments

trialData	data frame containing the trial data
trialColumn	character or integer vector of length 1 indicating the name or position of the column that contains the identifiers of the trials. If trialColumn is numeric the column is identified by its position in the phenoData data.frame. If trialColumn is of type character it is assumed to hold the name of the column in the phenoData data.frame.
indColumn	character or integer vector of length 1 indicating the name or position of the column that contains the identifiers of the individuals.

Value

list containing character vectors of connected trials

Index