Package ‘PhenStat’

March 28, 2024

Type Package
Title Statistical analysis of phenotypic data
Version 2.38.0
Date 2019-07-09
Author Natalja Kurbatova, Natasha Karp, Jeremy Mason, Hamed Haselimashhadi
Maintainer Hamed Haselimashhadi <hamedhm@ebi.ac.uk>
Depends R (>= 3.5.0)
Imports SmoothWin, methods, car, nlme, nortest, MASS, msgps, logistf,
        knitr, tools, pingr, ggplot2, reshape, corrplot, graph, lme4,
        graphics, grDevices, utils, stats
Suggests RUnit, BiocGenerics
Description Package contains methods for statistical analysis of phenotypic data.
License file LICENSE
NeedsCompilation no
biocViews StatisticalMethod
git_url https://git.bioconductor.org/packages/PhenStat
git_branch RELEASE_3_18
git_last_commit 806485e
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-03-27

R topics documented:

analysedDataset ................................................................. 4
analysedDatasetPhenList ..................................................... 4
analysedSubset ................................................................. 5
analysisResults ................................................................. 5
batchIn ......................................................................... 6
boxplotResidualBatch ........................................................ 6
R topics documented:

boxplotSexGenotype .................................................. 7
boxplotSexGenotypeBatch ........................................... 8
boxplotSexGenotypeBatchAdjusted ................................. 10
boxplotSexGenotypeResult ........................................... 11
boxplotSexGenotypeWeightBatchAdjusted ......................... 12
categoricalBarplot ................................................... 13
checkDataset .......................................................... 14
classificationTag ...................................................... 16
columnChecks .......................................................... 17
columnLevels ........................................................... 18
comparison ............................................................. 19
dataset ................................................................. 19
determiningLambda .................................................... 19
dim ........................................................................... 20
finalLRModel ............................................................ 21
finalModel ............................................................... 22
finalTFModel ............................................................ 24
FisherExactTest .......................................................... 25
formulaDeterminingLambda ........................................... 27
generateGraphs .......................................................... 27
getColumn ............................................................... 28
getColumnBatchAdjusted ............................................... 29
getColumnView .......................................................... 29
getColumnWeightBatchAdjusted ..................................... 29
getCountMatrices ......................................................... 30
getDataset ............................................................... 30
getGenotypeEffect ....................................................... 30
getMatrix ............................................................... 31
getPercentageMatrix .................................................... 31
getStat ................................................................. 31
getVariable ............................................................. 32
getVariables ............................................................ 32
hemiGenotype ........................................................... 32
htest-class .............................................................. 32
htestPhenStat-class ..................................................... 33
JSONOutput ............................................................. 35
LargeDataObject-class ................................................ 36
LRDataset ............................................................... 37
matrixCount ............................................................. 38
method ................................................................. 38
methodText ............................................................. 39
modelFormula ........................................................... 39
modelFormulaLR ........................................................ 40
multipleBatches ........................................................ 41
noSexes ................................................................. 42
parameters ............................................................. 42
parserOutputSummary .................................................. 42
parserOutputSummaryLR ............................................... 43
R topics documented:

- parserOutputTFSummary .......................................................... 44
- performReverseTransformation ................................................. 45
- performTransformation ............................................................ 46
- PhenList .................................................................................. 47
- PhenList-class ................................................................. 49
- PhenStatReport ................................................................. 51
- PhenTestResult ................................................................. 53
- PhenTestResult-class ......................................................... 55
- plot.PhenList ........................................................................ 58
- plot.PhenTestResult .............................................................. 59
- plotResidualPredicted .............................................................. 61
- printLROutput ..................................................................... 62
- printTabStyle ...................................................................... 63
- pvalue .............................................................................. 64
- qqplotGenotype .................................................................... 64
- qqplotRandomEffects .............................................................. 65
- qqplotRotatedResiduals ............................................................. 66
- recommendMethod ................................................................. 67
- refGenotype ....................................................................... 68
- RRTest .............................................................................. 69
- scatterplotGenotypeWeight ...................................................... 70
- scatterplotGenotypeWeightResult ........................................... 71
- scatterplotSexGenotypeBatch .................................................. 73
- scatterplotSexGenotypeBatchResult ....................................... 74
- setBatch ........................................................................... 75
- setGenotype ...................................................................... 76
- setMissingValue ................................................................. 76
- setSex ............................................................................ 76
- setWeight ....................................................................... 77
- startLRModel ................................................................. 77
- startModel ....................................................................... 79
- startTFModel ................................................................. 81
- subSetText ........................................................................ 83
- summary.PhenTestResult ........................................................ 83
- summaryOutput .............................................................. 84
- testDataset ................................................................. 85
- testFinalLRModel ............................................................... 88
- testFinalModel ............................................................... 90
- testGenotype ................................................................... 91
- TFDataset .................................................................... 91
- transformation ................................................................. 92
- transformationJSON .............................................................. 93
- transformationText .............................................................. 93
- vectorOutput ................................................................. 93
- vectorOutputMatrices ............................................................. 95
- weightIn ....................................................................... 96

Index ........................................................................ 97
analysedDataset PhenList

analysedDataset  

Method "analysedDataset"

Description

Method analysedDataset returns subset of the original dataset that was analysed.

Value

This function returns the analysed dataset.

Methods

signature(obj = "PhenTestResult")

Examples

# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
depVariable="Lean.Mass")
head(PhenStat:::analysedDataset(obj = result))

analysedDatasetPhenList

Method "analysedDatasetPhenList"

Description

Method analysedDatasetPhenList returns subset of the original dataset that will be analysed.

Value

The function returns the analysed dataset.

Methods

signature(obj = "PhenList")
**analysedSubset**

*Method "analysedSubset"*

**Description**

Method `analysedSubset` returns name of the subset that was analysed by RR or FE frameworks.

**Value**

The function returns name of the subset that was analysed by RR or FE frameworks.

**Methods**

```r
signature(obj = "htestPhenStat")
```

**analysisResults**

*Method "analysisResults"*

**Description**

Method `analysisResults` returns analysis results.

**Value**

The function returns the analysis result.

**Methods**

```r
signature(obj = "PhenTestResult")
```

**Examples**

```r
# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                           testGenotype="Sparc/Sparc")

result <- PhenStat:::testDataset(test, depVariable="Lean.Mass")
r = PhenStat:::analysisResults(obj = result)
```
**Method "batchIn"**

**Description**

Method `batchIn` returns TRUE/FALSE values depending on the batch column presence/absence in the dataset.

**Value**

TRUE/FALSE

**Methods**

- `signature(obj = "PhenList")`
- `signature(obj = "PhenTestResult")`

---

**Method "boxplotResidualBatch"**

**Description**

Graph function for the Mixed Model framework’s results. Creates a box plot with residue versus batch split by genotype.

**Usage**

```r
boxplotResidualBatch(phenTestResult, outputMessages=TRUE)
```

**Arguments**

- `phenTestResult` instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Empty.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenList and PhenTestResult

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file, na.strings = '-'),
                        testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
                        depVariable="Lean.Mass")
PhenStat:::boxplotResidualBatch(result)
```

Description

Graph function for the phenotypic dataset. Creates a box plot split by sex and genotype.

Usage

```r
boxplotSexGenotype(phenList, 
                    depVariable=NULL, 
                    graphingName=NULL, 
                    outputMessages=TRUE)
```

Arguments

- **phenList** instance of the PhenList class; mandatory argument
- **depVariable** a character string defining the dependent variable of interest; mandatory argument
- **graphingName** a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- **outputMessages** flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value

Empty.
Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
testGenotype="Sparc/Sparc")

# box plot for dataset with two sexes: males and females
PhenStat:::boxplotSexGenotype(test,
  depVariable="Bone.Mineral.Content",
  graphingName="BMC")

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
  testGenotype="Mysm1/+")

# box plot for females only dataset
PhenStat:::boxplotSexGenotype(test_1sex,
  depVariable="Lean.Mass",
  graphingName="Lean Mass (g)")
```

---

**Method "boxplotSexGenotypeBatch"**

**Description**

NB! Function is deprecated, please use scatterplotSexGenotypeBatch function instead. Graph function for the phenotypic dataset. Creates a box plot split by sex, genotype and batch.

Note: the batches are not ordered with time but allow assessment of how the treatment groups lie relative to the normal control variation.

**Usage**

```r
boxplotSexGenotypeBatch(phenList,
  depVariable=NULL,
  graphingName=NULL,
  outputMessages=TRUE)
```
boxplotSexGenotypeBatch

Arguments

- **phenList**: instance of the `PhenList` class; mandatory argument
- **depVariable**: a character string defining the dependent variable of interest; mandatory argument
- **graphingName**: a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- **outputMessages**: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value

Empty.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

`PhenList`

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Sparc/Sparc")
# box plot for dataset with two sexes: males and females
PhenStat:::boxplotSexGenotypeBatch(test,
                           depVariable="Bone.Mineral.Content",
                           graphingName="BMC")

file <- system.file("extdata", "test4.csv", package="PhenStat")
test1sex <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Mysm1/+")
# box plot for females only dataset
PhenStat:::boxplotSexGenotypeBatch(test1sex,depVariable="Lean.Mass",
                           graphingName="Lean Mass")
```
Method "boxplotSexGenotypeBatchAdjusted"

Description
Graph function for the phenotypic dataset. Creates a box plot split by sex and genotype after accounting for batch.

Usage
```r
boxplotSexGenotypeBatchAdjusted(
    phenList,
    depVariable=NULL,
    graphingName=NULL,
    outputMessages=TRUE)
```

Arguments
- `phenList`: instance of the `PhenList` class; mandatory argument
- `depVariable`: a character string defining the dependent variable of interest; mandatory argument
- `graphingName`: a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value
Empty.

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References

See Also
- `PhenList`
**boxplotSexGenotypeResult**

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
    testGenotype="Sparc/Sparc")
# box plot for dataset with two sexes: males and females
PhenStat:::boxplotSexGenotypeBatchAdjusted(test,
    depVariable="Bone.Mineral.Content",
    graphingName="BMC")
```

---

**Description**

Graph function for the phenotypic dataset. Creates a box plot split by sex and genotype. Is based on PhenTestResult object.

**Usage**

```r
boxplotSexGenotypeResult(phenTestResult,
    graphingName=NULL,
    outputMessages=TRUE)
```

**Arguments**

- `phenTestResult`: instance of the `PhenTestResult` class; mandatory argument
- `graphingName`: a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Empty.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
  testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
  depVariable="Lean.Mass")
# box plot for dataset with two sexes: males and females
PhenStat:::boxplotSexGenotypeResult(result,
  graphingName="BMC")

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
  testGenotype="Mysm1/+")
result_1sex <- PhenStat:::testDataset(test_1sex,
  depVariable="Lean.Mass")
# box plot for females only dataset
PhenStat:::boxplotSexGenotypeResult(result_1sex,
  graphingName="Lean Mass (g)")
```

boxplotSexGenotypeWeightBatchAdjusted

*Method "boxplotSexGenotypeWeightBatchAdjusted"

Description

Graph function for the phenotypic dataset. Creates a box plot split by sex and genotype after accounting for batch and weight.

Usage

```r
boxplotSexGenotypeWeightBatchAdjusted(
  phenList,
  depVariable=NULL,
  graphingName=NULL,
  outputMessages=TRUE)
```

Arguments

- `phenList`: instance of the `PhenList` class; mandatory argument
- `depVariable`: a character string defining the dependent variable of interest; mandatory argument
- `graphingName`: a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
Value
Empty.

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References

See Also
PhenList

Examples
```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Sparc/Sparc")
# box plot for dataset with two sexes: males and females
PhenStat:::boxplotSexGenotypeWeightBatchAdjusted(test,
                                              depVariable="Bone.Mineral.Content",
                                              graphingName="BMC")
```

categoricalBarplot  Method "categoricalBarplot"

Description
Graph function for the Fisher Exact Test framework’s results. Creates stacked bar plot(s) to compare proportions seen in a categorical variable between different genotypes. Graphs are created for all data and also for each sex individually.

Usage
categoricalBarplot(phenTestResult,
outputMessages=TRUE)

Arguments
phenTestResult  instance of the PhenTestResult class that comes from the function testDataset; mandatory argument
outputMessages  flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
checkDataset

Value
Empty.

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References

See Also
PhenList

Examples
```r
cat <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
testGenotype="Aff3/Aff3")
result <- PhenStat:::testDataset(test,
depVariable="Thoracic.Processes",
method="FE")
PhenStat:::categoricalBarplot(result)
```

checkDataset    Method "checkDataset"

Description
Checks dataset for the minimum required info:
1. Column names should be present
2. Genotype column should be present
3. Sex column should be present
4. Two data points for each Genotype/Sex combination
5. Number of Genotype levels should be 2
6. Number of Sex levels should be 1 or 2
7. Sex levels have to be "Female" and/or "Male"
8. Records with reference genotype should be in the dataset
9. Records with test genotype should be in the dataset
Perform the following additional checks:
- presence of Weight column,
- presence of Batch column (Assay.Date).
Warning given in case of missed data indicating that you can only fit a glm or to use MM equation "withoutWeight".
Function `checkDataset` is called from `PhenList` function.

Usage

```r
checkDataset(dataset, testGenotype, refGenotype="+/+", outputMessages=TRUE, dataset.clean=TRUE)
```

Arguments

- `dataset`: data frame created from file or from another source; mandatory argument
- `testGenotype`: defines the test genotype to be compared to the reference genotype; mandatory argument
- `refGenotype`: defines the reference genotype; assigned default value is "+/+"
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; assigned default value is TRUE
- `dataset.clean`: flag: "FALSE" value for no modification of the dataset; "TRUE" value to clean dataset if needed; assigned default value is TRUE

Value

Returns an instance of the `PhenList` class.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

`PhenTestResult`
classificationTag

Examples

# "checkDataset" is called internally from "PhenList" function
classificationTag
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
  testGenotype="Sparc/Sparc")

Description

Returns a classification tag to assign a sexual dimorphism assessment of the phenotypic change.

Usage

classificationTag(phenTestResult,
  userMode = "summaryOutput",
  phenotypeThreshold = 0.01,
  outputMessages=TRUE)

Arguments

phenTestResult instance of the PhenTestResult class that comes from the function testDataset; mandatory argument
userMode flag: "vectorOutput" a sexual dimorphic classification is assigned with a proviso of later assessing the overall statistical significance; "summaryOutput" the phenotypeThreshold is used to assess the overall statistical significance and then if significant the sexual dimorphic classification determined; defaults to summaryOutput
phenotypeThreshold a numerical value defining the threshold to use in classificationTag in determining whether the genotype effect is classed as significant or not; default value 0.01
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages ; default value TRUE

Value

Returns a classification tag to assign a sexual dimorphism assessment of the phenotypic change.

If you are working interactively with the data, the argument "userMode" set to the value "summaryOutput" will use the "phenotypeThreshold" argument’s value to assess statistical significance of the genotype effect and if significant then assign a sexual dimorphic classification. Alternatively, if the "userMode" set to the value "vectorMode", a sexual dimorphic classification will be returned with the MM framework where later you can globally assess whether the variable had a significant genotype effect. With the FE framework and the vectorMode, a NA is returned as the type of the effect cannot be assessed without assessing the statistical significance of the genotype effect.
**columnChecks**

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

PhenTestResult

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Sparc/Sparc")
result <- PhenStat::testDataset(test,
                                 depVariable="Lean.Mass")
PhenStat::classificationTag(result,
                          userMode="summaryOutput",
                          phenotypeThreshold=0.001)
PhenStat::classificationTag(result,
                          userMode="vectorOutput")
```

---

**columnChecks**

*Method "columnChecks"*

**Description**

Checks the particular column of the dataset for eligibility. Returns: presence of column, all data are numeric, number of levels that passed check (number of data points for each genotype/sex combination is at least equals to threshold).

**Usage**

```r
columnChecks(dataset,
              columnName,
              dataPointsThreshold=4)
```

**Arguments**

- **dataset**: data frame created from file or from another source; mandatory argument
- **columnName**: name of the column to check; mandatory argument
- **dataPointsThreshold**: number of data points for each genotype/sex combination; default value is set to 4
columnLevels

Value

Boolean vector of length 3: presence of column, all data are numeric, number of levels that passed check (number of data points for each genotype/sex combination is at least equals to threshold).

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


---

columnLevels Method "columnLevels"

Description

Method removes null values from the column and calculates column basic stat. Returns: No of data points, No of levels, No of Genotype/Sex combinations, No of data points for each one combination.

Usage

columnLevels(dataset, columnName)

Arguments

dataset data frame created from file or from another source; mandatory argument

colName name of the column to check; mandatory argument

Value

Numeric vector of length 6 or 7: No of data points, No of levels, No of Genotype/Sex combinations, No of data points for each one combination.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


**comparison**  

*Method "comparison"*

**Description**

Method `comparison` returns name of the comparison that was performed by RR framework: "High vs Normal/Low" or "Low vs Normal/High"

**Value**

returns name of the comparison that was performed by RR framework.

**Methods**

`signature(obj = "htestPhenStat")`

---

**dataset**  

*Method "dataset"*

**Description**

Method `dataset` returns data frame stored in the object

**Value**

returns data frame stored in the object

**Methods**

`signature(obj = "PhenList")`

---

**determiningLambda**  

*Method "determiningLambda"*

**Description**

Function role 1. Rescale data if needed (Box-Cox transformation can only be applied on data >0) 2. Calculate lambda value for a Box-Cox transformation 3. Assess whether transformation is required (if confidence interval includes 1 then no transformation required) 4. Return output to allow application of the transformation if necessary using the returned lambda value and any associated rescaling that was needed. 5. Requires a function that will be fitted determined from `formulaAssessingBoxCox`
determiningLambda(phenList, depVariable, equation="withWeight")

Arguments

phenList instance of the PhenList class; mandatory argument
depVariable a character string defining the dependent variable of interest; mandatory argument
equation a character string defining the equation to use. Possible values "withWeight" (default),"withoutWeight"

Value


Author(s)

Natasha Karp

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"), testGenotype="Sparc/Sparc")
PhenStat:::determiningLambda(test,"Lean.Mass",equation="withoutWeight")
```

Method "dim"

Description

Retrieve the number of rows (measures) and columns (parameters) for a PhenList object.

Usage

```r
## S3 method for class 'PhenList'
dim(x)
```

Arguments

x an object of class PhenList
Details

These methods allow one to extract the size of phenotypic data objects in the same way that one would do for ordinary matrices.
A consequence is that row and column commands `nrow(x)`, `ncol(x)` and so on also work.

Value

Numeric vector of length 2. The first element is the number of rows (measures) and the second is the number of columns (parameters).

Author(s)

Natalja Kurbatova

See Also

dim in the base package.

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
testGenotype="Sparc/Sparc")
PhenStat:::dim.PhenList(test)
```

finalLRModel

Method "finalLRModel"

Description

This is an internal function run within LR framework. It completes the final stage of the LR framework, which builds the final model and estimates effects. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.

Works with `PhenTestResult` object created by `startLRModel` function. The creation of LR final model is based on the significance of different fixed effects, depVariable and equation values stored in `PhenTestResult` object.

Usage

```r
finalLRModel(phenTestResult,
outputMessages = TRUE)
```

Arguments

- `phenTestResult`: instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
Value

Returns results stored in instance of the `PhenTestResult` class.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

`PhenTestResult` and `testDataset`

Examples

```r
file <- system.file("extdata", "testLR.csv", package="PhenStat")
testLR <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Mock")
testLR2 <- PhenStat:::LRDataset(testLR,"V2")

# when "testDataset" function's argument "callAll" is set to FALSE
# only "startLRModel" function is called - the first step of LR framework
resultLR <- PhenStat:::testDataset(testLR2,
                          depVariable="V2",
                          callAll=FALSE,
                          method="LR")

# print out formula that has been created
PhenStat:::analysisResults(resultLR)$model.formula.genotype

# print out batch effect's significance
PhenStat:::analysisResults(resultLR)$model.effect.batch

resultLR <- PhenStat:::finalLRModel(resultLR)
```

---

### finalModel

**Method "finalModel"**

**Description**

This is an internal function run within MM framework. It completes the final stage of the MM framework, which builds the final model and estimates effects. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.

Works with `PhenTestResult` object created by `startModel` function. The creation of MM final model is based on the significance of different fixed effects, depVariable and equation values stored in `PhenTestResult` object.
finalModel

Usage

finalModel(phenTestResult,
outputMessages = TRUE,
modelWeight = NULL)

Arguments

phenTestResult instance of the PhenTestResult class that comes from the function testDataset; mandatory argument
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
modelWeight A vector of positive values for weights mainly used for applying windowing to the data

Value

Returns results stored in instance of the PhenTestResult class

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenTestResult and testDataset

Examples

file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
testGenotype="Sparc/Sparc")
# when "testDataset" function's argument "callAll" is set to FALSE
# only "startModel" function is called - the first step of MM framework
result <- PhenStat:::testDataset(test,
   depVariable="Lean.Mass",
   equation="withoutWeight",
   callAll=FALSE)
# print out formula that has been created
# result$model.formula.genotype
# print out batch effect's significance
# result$model.effect.batch
# change the model
# result <- PhenStat:::testDataset(test,
Method "finalTFModel"

Description

This is an internal function run within TF framework. It completes the final stage of the TF framework, which builds the final model and estimates effects. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.

Works with PhenTestResult object created by startTFModel function. The creation of TF final model is based on the significance of different fixed effects, depVariable and equation values stored in PhenTestResult object.

Usage

```r
finalTFModel(phenTestResult, outputMessages = TRUE)
```

Arguments

- `phenTestResult` instance of the PhenTestResult class that comes from the function testDataset; mandatory argument
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value

Returns results stored in instance of the PhenTestResult class

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


FisherExactTest

See Also

PhenTestResult and testDataset

Examples

```r
file <- system.file("extdata", "test7_TFE.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file,na.strings = '-'),
                 testGenotype="het",
                 refGenotype = "WT",
                 dataset.colname.sex="sex",
                 dataset.colname.genotype="Genotype",
                 dataset.values.female="f",
                 dataset.values.male = "m",
                 dataset.colname.weight="body.weight",
                 dataset.colname.batch="Date_of_procedure_start")

test_TF <- PhenStat:::TFDataset(test,depVariable="Cholesterol")

# when "testDataset" function's argument "callAll" is set to FALSE
# only "startTFModel" function is called - the first step of TFE framework
result <- PhenStat:::testDataset(test_TF,
                                   depVariable="Cholesterol",
                                   callAll=FALSE,
                                   method="TF")

# print out formula that has been created
PhenStat:::analysisResults(result)$model.formula.genotype

# print out batch effect's significance
PhenStat:::analysisResults(result)$model.effect.batch

result <- PhenStat:::finalTFModel(result)
```

FisherExactTest Method "FisherExactTest"

Description

The main function of the Fisher Exact Test framework. Creates n times 2 matrices with record counts, where n rows represent dependent variable levels and two columns represent genotype levels. Performs Fisher Exact Tests on calculated count matrices.

Three matrices can be created and three tests can be potentially perform depending on the dataset:
- all records together regardless the sex values - combined dataset,
- records where sex value is "Male" (if such exists) - males only dataset,
- records where sex value is "Female" (if such exists) - females only dataset.


The results (matrices, statistics and Fisher Exact Test outputs) are stored in PhenTestResult object. The function is called from "testDataset" function when "method" argument is set to "FE" meaning "Fisher Exact Test".
FisherExactTest

Usage

FisherExactTest(phenList, depVariable, outputMessages=TRUE)

Arguments

phenList instance of the PhenList class; mandatory argument
depVariable a character string defining the dependent variable of interest; mandatory argument
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value

Returns results stored in instance of the PhenTestResult class

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenTestResult

Examples

file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
             testGenotype="Aff3/Aff3")
# "FisherExactTest" function is called from "testDataset" function.
result <- PhenStat:::testDataset(test,depVariable="Thoracic.Processes",method="FE")
# Fisher Exact Test results can be printed out using function "summaryOutput"
summary(result)
**formulaDeterminingLambda**

*Method "formulaDeterminingLambda"*

**Description**

Returns the starting formula that we wish to interrogate the data with in estimating the lambda for a Box-Cox transformation

**Usage**

```r
formulaDeterminingLambda(noSexes, depVariable, multipleBatches, equation)
```

**Arguments**

- `noSexes`: number of sexes in the data; mandatory argument
- `depVariable`: a character string defining the dependent variable of interest; mandatory argument
- `multipleBatches`: TRUE if there are multiple batch values in the data, FALSE otherwise
- `equation`: a character string defining the equation to use. Possible values "withWeight" (default), "withoutWeight"

**Value**

Returns a formula

**Author(s)**

Natasha Karp

---

**generateGraphs**

*Method "generateGraphs"*

**Description**

Function generates graphs for the Mixed Model and Time as Fixed Effect frameworks' results and stores generated graphs in the defined directory.

**Usage**

```r
generateGraphs(phenTestResult, dir, graphingName=NULL, type="Xlib")
```
**Arguments**

- `phenTestResult`: instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `dir`: directory where to store generated graphs; mandatory argument
- `graphingName`: a character string defining the name to be used on the graph for the dependent variable; defaults to NULL then the depVariable name will be plotted
- `type`: character string, one of "Xlib" or "quartz" (some OS X builds) or "cairo", or "window", or cairo-png"; mandatory argument

**Value**

Empty.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

`PhenTestResult`

---

**Description**

Method `getColumn` returns column of interest

**Value**

This function returns column of interest

**Methods**

- `signature(obj = "PhenList")`
**getColumnBatchAdjusted**

*Method "getColumnBatchAdjusted"*

**Description**

Method `getColumnBatchAdjusted` returns column of interest adjusted for batch

**Value**

This function returns column of interest

**Methods**

signature(obj = "PhenList")

---

**getColumnView**

*Method "getColumnView"*

**Description**

Method `getColumnView` returns RR or FE frameworks results in a column format

**Value**

This function returns RR or FE frameworks results in a column format

**Methods**

signature(obj = "htestPhenStat")

---

**getColumnWeightBatchAdjusted**

*Method "getColumnWeightBatchAdjusted"*

**Description**

Method `getColumnWeightBatchAdjusted` returns column of interest adjusted for weight and batch

**Value**

This function returns column of interest

**Methods**

signature(obj = "PhenList")
**getCountMatrices**

**Method "getCountMatrices"**

**Description**
Method `getCountMatrices` returns count matrices for FE and RR frameworks

**Value**
This function returns count matrices for FE and RR frameworks

**Methods**
signature(obj = "PhenTestResult")

---

**getDataset**

**Method "getDataset"**

**Description**
Method `getDataset` returns data frame stored in the object

**Value**
This function returns data frame stored in the object.

**Methods**
signature(obj = "PhenList")

---

**getGenotypeEffect**

**Method "getGenotypeEffect"**

**Description**
Method `getGenotypeEffect` returns genotype effect and standard error for the linear regression frameworks

**Value**
This function returns genotype effect and standard error for the linear regression frameworks

**Methods**
signature(obj = "PhenTestResult")
### getMatrix

**Method** "getMatrix"

**Description**
Method `getMatrix` returns RR or FE frameworks results in a matrix format.

**Value**
This function returns RR or FE frameworks results in a matrix format.

**Methods**
signature(obj = "htestPhenStat")

### getPercentageMatrix

**Method** "getPercentageMatrix"

**Description**
Method `getPercentageMatrix` returns percentage matrix calculated out of counts.

**Value**
This function returns percentage matrix calculated out of counts.

**Methods**
signature(obj = "htestPhenStat")

### getStat

**Method** "getStat"

**Description**
Method `getStat` returns simple statistics about variables in the dataset.

**Value**
This function returns simple statistics about variables in the dataset.

**Methods**
signature(obj = "PhenList")
### getVariable

**Method** `getVariable`

**Description**
Method `getVariable` returns dependent variable name

**Value**
This function returns dependent variable name

**Methods**
- `signature(obj = "PhenTestResult")`

### getVariables

**Method** `getVariables`

**Description**
Method `getVariables` returns names of variables in the dataset

**Value**
This function returns names of variables in the dataset

**Methods**
- `signature(obj = "PhenList")`

### hemiGenotype

**Method** `hemiGenotype`

**Description**
Method `hemiGenotype` returns hemi genotype if defined

**Value**
This function returns hemi genotype if defined

**Methods**
- `signature(obj = "PhenList")`
- `signature(obj = "PhenTestResult")`
Description

An old S3 class from package stats rewritten into S4 style in order to add functionality needed for PhenStat.

Value

S3 class

Explore htest object

A list with class "htest" containing the following components:

1. p.value: the p-value of the test.
2. conf.int: a confidence interval for the odds ratio. Only present in the 2 by 2 case and if argument conf.int = TRUE.
3. estimate: an estimate of the odds ratio. Note that the conditional Maximum Likelihood Estimate (MLE) rather than the unconditional MLE (the sample odds ratio) is used. Only present in the 2 by 2 case.
4. null.value: the odds ratio under the null, or. Only present in the 2 by 2 case.
5. alternative: a character string describing the alternative hypothesis.
6. method: the character string "Fisher’s Exact Test for Count Data".
7. data.name: a character string giving the names of the data.

Author(s)

Natalja Kurbatova

See Also

htestPhenStat-class
Description

Extension of htest class for the PhenStat needs

Value

Empty.

Explore htestPhenStat object

A list with "htest" components in modelOutput slot and the additional PhenStat specific slots:
1. modelOutput: htest object with Fisher Exact test (fisher.test) output
2. analysedSubset: the subset analysed with possible values "all", "males", "females"
3. comparison: for the RR method with possible values "High vs Normal/Low", "Low vs Normal/High"
4. ES: effect size value
5. matrixCount: matrix of counts used for the test

Slots

modelOutput: Object of class "htest" ~
analysedSubset: Object of class "character" ~
comparison: Object of class "character" ~
ES: Object of class "numeric" ~
matrixCount: Object of class "matrix" ~

Methods

columnView signature(obj = "htestPhenStat"): ...
gMatrix signature(obj = "htestPhenStat"): ...
gPercentageMatrix signature(obj = "htestPhenStat"): ...
subsetText signature(object = "htestPhenStat"): ...
comparison signature(object = "htestPhenStat"): ...
show signature(object = "htestPhenStat"): ...

Author(s)

Natalja Kurbatova
See Also

htest-class

Examples

```r
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
    testGenotype="Aff3/Aff3")
# "FisherExactTest" function is called from "testDataset" function.
result <- PhenStat:::testDataset(test,depVariable="Thoracic.Processes",method="FE")
# Fisher Exact Test results can be printed out using function "summaryOutput"
val <- PhenStat:::analysisResults(result)[[1]]
print(val)
class(val)
```

---

**JSONOutput Method "JSONOutput"**

Description

Wrapper for the model fitting results. Returns model fitting and testing results in a JSON format. Assumes that modelling results are stored in the `PhenTestResult` object (output from function `testDataset`).

Usage

```r
JSONOutput(phenTestResult, phenotypeThreshold = 0.01)
```

Arguments

- **phenTestResult**: instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- **phenotypeThreshold**: a numerical value defining the threshold to use in `classificationTag`; default value 0.01

Value

Returns a vector with model fitting results in JSON format: model output summary and other values

Author(s)

Natalja Kurbatova

See Also

PhenTestResult
Examples

## Not run:

```r
# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
                                  depVariable="Lean.Mass")
PhenStat:::JSONOutput(result)

# Fishe Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test2 <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                            testGenotype="Aff3/Aff3")
result2 <- PhenStat:::testDataset(test2,
                                  depVariable="Thoracic.Processes",
                                  method="FE")
PhenStat:::JSONOutput(result2)
```

## End(Not run)

---

**LargeDataObject-class**

```
Class "LargeDataObject"
```

### Description

Internal class. A list-based S4 class for storing large data objects.

### Value

Empty.

### Methods

- **show**
  
  signature(object = "LargeDataObject"): Prints out the object head using common method `show`

- **printHead**
  
  signature(object = "LargeDataObject"): Print leading 5 elements or rows of atomic object

### Author(s)

Gordon Smyth

### References

**LRDataset**  

*Method "LRDataset"*

**Description**

Prepares dataset for the LR framework - maps values of dependent variable to 0/1, where 1 is modeled.

**Usage**

```r
LRDataset(
  phenList = NULL,
  depVariable = NULL,
  abnormalValues = c("abnormal", "Abnormal", "TRUE", "deviant"),
  outputMessages = TRUE
)
```

**Arguments**

- `phenList` instance of the `PhenList` class; mandatory argument
- `depVariable` a character string defining the dependent variable of interest; mandatory argument
- `abnormalValues` a list of abnormal values that will be mapped to 0; mandatory argument
- `outputMessages` flag: “FALSE” value to suppress output messages; “TRUE” value to show output messages; default value TRUE

**Value**

Returns dataset suitable for LR framework, where provided abnormal values are mapped to 1 and all other values of dependent variable are assumed to be normal and are mapped to 0.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


Examples

```r
file <- system.file("extdata", "testLR.csv", package="PhenStat")
testLR <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
testGenotype="Mock")
testLR2 <- PhenStat:::LRDataset(testLR,"V2")
```

---

**matrixCount**  
*Method "matrixCount"*

**Description**

Method matrixCount returns matrix of counts that was created by RR or FE framework.

**Value**

This function returns matrix of counts that was created by RR or FE framework.

**Methods**

```r
signature(obj = "htestPhenStat")
```

---

**method**  
*Method "method"*

**Description**

Method method returns method abbreviation, for example: "RR", "MM".

**Value**

This function returns returns method abbreviation.

**Methods**

```r
signature(obj = "PhenTestResult")
```
**Method Text**

**Method Text** returns method's name, for example: "Reference Range Plus", "Linear Mixed Model".

**Value**

This function returns method's name.

**Methods**

signature(obj = "PhenTestResult")

**Description**

Creates formula for the start Mixed Model based on equation and number of sexes in the data.

**Usage**

modelFormula(equation, numberofsexes, depVariable)

**Arguments**

- **equation**: a character string defining the equation to use (withWeight or withoutWeight); mandatory argument
- **numberofsexes**: a numerical value for levels of sex in the dataset (1 or 2); mandatory argument
- **depVariable**: a character string defining the dependent variable of interest; mandatory argument

**Value**

Returns formula to use.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenTestResult

Examples

```r
formula <- PhenStat:::modelFormula(equation="withWeight",
        numberofsexes=2,
        depVariable="Lean.Mass")
```

### Description

Creates formula for the start Mixed Model based on equation and number of sexes in the data.

**Usage**

```r
modelFormulaLR(numberofsexes, depVariable, sexIncluded, dimorphismIncluded, IncludeBatch)
```

**Arguments**

- `numberofsexes`: a numerical value for levels of sex in the dataset (1 or 2); mandatory argument
- `depVariable`: a character string defining the dependent variable of interest; mandatory argument
- `sexIncluded`: include sex effect into formula or not; mandatory argument
- `dimorphismIncluded`: include interaction effect of Genotype/Sex into formula or not; mandatory argument
- `IncludeBatch`: include batch effect into formula or not; mandatory argument

**Value**

Returns formula to use.
multipleBatches

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also
PhenTestResult

Examples

```r
formula <- PhenStat:::modelFormulaLR(numberofsexes=2,
         depVariable="Lean.Mass",TRUE,TRUE,"No")
```

\[
\text{multipleBatches} \quad \text{Method "multipleBatches"}
\]

Description
Method multipleBatches returns TRUE if there are multiple batches in the dataset, FALSE otherwise.

Value
This function returns TRUE if there are multiple batches in the dataset.

Methods
signature(obj = "PhenList")
**Method "noSexes"**

**Description**

Method noSexes returns number of sex levels: 1/2

**Value**

This function returns number of sex levels.

**Methods**

signature(obj = "PhenList")
signature(obj = "PhenTestResult")

---

**Method "parameters"**

**Description**

Method parameters returns parameters specific for applied method/dataset combination

**Value**

This function returns parameters specific for applied method/dataset combination.

**Methods**

signature(obj = "PhenTestResult")

---

**Method "parserOutputSummary"**

**Description**

Parsers model output summary and returns it in readable named vector format

**Usage**

parserOutputSummary(linearRegressionOutput)
**parserOutputSummaryLR**

**Arguments**

- **linearRegressionOutput**
  
  linear regression output that comes from the MM methods; mandatory argument

**Value**

Returns a named vector with linear regression model output summary results

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

*PhenTestResult*

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file, na.strings = "-"),
  testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
  depVariable="Lean.Mass")
linearRegressionOutput <- PhenStat:::analysisResults(result)
PhenStat:::parserOutputSummary(linearRegressionOutput)
```

---

**parserOutputSummaryLR Method "parserOutputSummaryLR"**

**Description**

Parsers model output summary and returns it in readable named vector format

**Usage**

```r
parserOutputSummaryLR(linearRegressionOutput)
```

**Arguments**

- **linearRegressionOutput**
  
  linear regression output that comes from the MM methods; mandatory argument
ParserOutputTFSummary

Description

Parsers model output summary from TF framework and returns it in readable named vector format.

Usage

ParserOutputTFSummary(linearRegressionOutput)

Arguments

linearRegressionOutput

Linear regression output that comes from the TF method; mandatory argument.

Value

Returns a named vector with model output summary results.
Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References

See Also
PhenTestResult

Examples
```r
file <- system.file("extdata", "test6_RR.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-')
                           testGenotype="Oxr1/Oxr1")
result <- PhenStat:::testDataset(test,
                                  depVariable="Ca",
                                  method="TF",
                                  dataPointsThreshold=2)
linearRegressionOutput <- PhenStat:::analysisResults(result)
PhenStat:::parserOutputTFSummary(linearRegressionOutput)
```

performReverseTransformation

Method "performReverseTransformation"

Description
Reverse back the transformed values according to the lambda value: exponential transformation if the lambda is 0, fractional power transformation otherwise

Usage
```r
performReverseTransformation(values, lambda, scaleShift)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>vector of values to reverse</td>
</tr>
<tr>
<td>lambda</td>
<td>lambda value of Box-Cox transformation</td>
</tr>
<tr>
<td>scaleShift</td>
<td>scale shift value of Box-Cox transformation</td>
</tr>
</tbody>
</table>

Value
Returns values transformed back to original scale.
**Author(s)**

Natalja Kurbatova

**Examples**

```r
PhenStat:::performReverseTransformation(c(-5.7312462,1.3166139,-0.8921497),-0.7,0)
```

---

**Method "performTransformation"**

**Description**

Transforms the given vector of values according to the lambda value: log transformation if the lambda is 0, power transformation otherwise.

**Usage**

```r
performTransformation(values, lambda, scaleShift)
```

**Arguments**

- `values` vector of values to reverse
- `lambda` lambda value of Box-Cox transformation
- `scaleShift` scale shift value of Box-Cox transformation

**Value**

Returns values transformed according to Box-Cox rules.

**Author(s)**

Natalja Kurbatova

**Examples**

```r
PhenStat:::performTransformation(c(0.1,38,0.5),-0.7,0)
```
PhenList

Method "PhenList"

Description

Function to create data object from the data frame.

In addition to dependent variable column (the variable of interest) mandatory columns that should
present in the data frame are "Genotype" and "Sex". The "Assay.Date" column is used to model
"Batch" effect if not specified differently. "Weight" column is used to model body weight effect.

Function creates PhenList object, checks data integrity, renames columns when requested, provides
simple statistics about dataset.

Returns an instance of the PhenList object created from the data file.

Dataset is cleaned to ensure there is a maximum two sex levels and there are exactly two levels for
genotype. Data cleaning can be switched off by defining the argument "dataset.clean" as FALSE.

Usage

PhenList(dataset, testGenotype, refGenotype="+/+", hemiGenotype=NULL,
outputMessages=TRUE, dataset.clean=TRUE,
dataset.colname.batch=NULL, dataset.colname.genotype=NULL,
dataset.colname.sex=NULL, dataset.colname.weight=NULL,
dataset.values.missingValue=" ", dataset.values.male=NULL,
dataset.values.female=NULL)

Arguments

dataset data frame created from file or from another source; mandatory argument
testGenotype defines the test genotype to be compared to the reference genotype; mandatory
argument
refGenotype defines the reference genotype; assigned default value is "+/+"
hemiGenotype defines the genotype value for hemizygous that will be changed to test genotype
value
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output
messages; assigned default value is TRUE
dataset.clean flag: "FALSE" value for no modification of the dataset; "TRUE" value to clean
dataset if needed; assigned default value is TRUE
dataset.colname.batch column name within dataset for the batch effect
dataset.colname.genotype column name within dataset for the genotype info
dataset.colname.sex column name within dataset for the sex info
dataset.colname.weight
column name within dataset for the weight info
dataset.values.missingValue
value used as missing value in the dataset
dataset.values.male
value used to label "males" in the dataset
dataset.values.female
value used to label "females" in the dataset

Value
Returns an instance of the PhenList class.

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References

See Also
PhenList

Examples
```r
defile <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file, na.strings = '-'),
testGenotype="Sparc/Sparc")
defile <- system.file("extdata", "test2.csv", package="PhenStat")
test2 <- PhenStat::PhenList(dataset=read.csv(file, na.strings = '-'),
testGenotype="Arid4a/Arid4a",
refGenotype="+/+",
hemiGenotype="Arid4a/+",
dataset.colname.weight="Weight.Value")
defile <- system.file("extdata", "test3.csv", package="PhenStat")
test3 <- PhenStat::PhenList(dataset=read.csv(file, na.strings = '-'),
dataset.clean=TRUE,
dataset.values.female=1,
dataset.values.male=2,
testGenotype="Mysm1/+")
```
PhenList-class

Class "PhenList"

Description

A list-based S4 class for storing phenotypic data. Helps to support data integrity checks and statistics calculation. The PhenList object can be created by using function `PhenList`.

Explore PhenList object

PhenList object instance contains the following slots:

1. "datasetPL" where cleaned and checked dataset is stored: `getDataset(phenList)`
2. "testGenotype" where the genotype level to test is stored: `testGenotype(phenList)`
3. "refGenotype" where reference genotype value is stored with default value set to "+/+": `refGenotype(phenList)`
4. "hemiGenotype" if defined contains the genotype value for hemizygous: `hemiGenotype(phenList)`
6. "dataset.clean" flag value is stored, see PhenList for more details: `phenList@dataset.clean`
7. "dataset.colname" if defined contains dataset column names that have been renamed:
   - `phenList@dataset.colname.batch` column name for batch values
   - `phenList@dataset.colname.genotype` column name for genotype values
   - `phenList@dataset.colname.sex` column name for sex values
   - `phenList@dataset.colname.weight` column name for weight values
8. "dataset.values" if defined contains dataset values that have been changed during dataset cleaning process:
   - `phenList@dataset.values.missingValue` value used as missing value in the original dataset
   - `phenList@dataset.values.male` value used to label "males" in the original dataset
   - `phenList@dataset.values.female` value used to label "females" in the original dataset

Slots

datasetPL: Object of class "data.frame" ~ dataset to work with
refGenotype: Object of class "character" ~ reference genotype
testGenotype: Object of class "character" ~ test genotype
hemiGenotype: Object of class "character" ~ hemi genotype
dataset.colname.batch: Object of class "character" ~ column name for batch values
dataset.colname.genotype: Object of class "character" ~ column name for genotype values
dataset.colname.sex: Object of class "character" ~ column name for sex values
dataset.colname.weight: Object of class "character" ~ column name for weight values
dataset.values.missingValue: Object of class "character" -- value used as missing value in the original dataset

dataset.values.male: Object of class "character" -- value used to label "males" in the original dataset

dataset.values.female: Object of class "character" -- value used to label "females" in the original dataset

dataset.clean: Object of class "logical" -- flag value is stored

datasetUNF: Object of class "data.frame" -- unfiltered dataset

Methods

getDataset (accessor): Returns dataset

refGenotype (accessor): Returns reference genotype

testGenotype (accessor): Returns test genotype

hemiGenotype (accessor): Returns hemi genotype if specified

getColumn Returns specified column if exists

getColumnBatchAdjusted Returns specified column adjusted for batch effect

getColumnWeightBatchAdjusted Returns specified column adjusted for batch and weight effects

getStat Returns simple dataset statistics

getVariables Returns dataset column names

batchIn Returns TRUE if the batch is in the dataset, FALSE otherwise

weightIn Returns TRUE if the weight is in the dataset, FALSE otherwise

multipleBatches Returns TRUE if the batches are variable in the dataset, FALSE otherwise

noSexes Returns number of sexes in the dataset

setBatch Sets the batch column - change the column names

setGenotype Sets the genotype column - change the column names

setMissingValue Sets the missing value

setSex Sets the sex column - change the column names

setWeight Sets the weight column - change the column names

Author(s)

Natalja Kurbatova

See Also

PhenList
Examples

```r
file <- system.file("extdata","test1.csv",package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = "-"),
    testGenotype="Sparc/Sparc")
class(test)

file <- system.file("extdata","test2.csv",package="PhenStat")
test2 <- PhenStat::PhenList(dataset=read.csv(file,na.strings = "-"),
    testGenotype="Arid4a/Arid4a",
    refGenotype="+/+",
    hemiGenotype="Arid4a/+",
    dataset.colname.weight="Weight.Value")
PhenStat::getStat(test2)
PhenStat::testGenotype(test2)
PhenStat::refGenotype(test2)
PhenStat::hemiGenotype(test2)
```

---

PhenStatReport  Generating a pdf report from a PhenList object

### Description

This function takes a Phenlist object and generates a pdf report containing several statistical methods.

### Usage

```r
PhenStatReport(
    PhenlistObject ,
    depVariable = NULL ,
    other.response = NULL ,
    update = TRUE ,
    Gene.Symbol = NULL ,
    Response.name = NULL ,
    destination = NULL ,
    reportTitle = "Extended Statistical Report",
    DataRelease = NULL ,
    Showsource = FALSE ,
    open = FALSE ,
    clean = TRUE ,
    verbos = FALSE ,
    ...
)
```

### Arguments

- **PhenlistObject**  A phenlist object that is already created by PhenList() function in PhenStat.
- **depVariable**  String. Name of the dependent variable.
other.response  The vector of strings. A vector of names containing the other dependent variables in the data set. Default NULL.

update  Logical flag. Set to TRUE to get the latest version of the report on the fly. Default is TRUE.

Gene.Symbol  Optional string. Gene symbol. Default NULL.

Response.name  Optional string. Name of the dependent variable. Default NULL.

destination  Location of the final report file. The default is the working directory of R.

reportTitle  The title of the report that is printed on the top of the first page of the report.

DataRelease  Optional flag. Data release version. Default NULL.

Showsource  Logical flag. Set to TRUE to see the actual R codes that generate each section of the report. Default FALSE.

open  Logical flag. Set to TRUE to open the report after it is generated. Default FALSE.

clean  Logical flag. Set to TRUE to remove the auxiliary files after successfully generating the report. Default TRUE.

verbos  Logical flag. Setting to TRUE shows the details and progress of the report generating function on screen. Default FALSE.

...  List of other parameters that can be passed to the functions.

Details

See PhenStatReport page on the IMPC website.

Value

PhenlistObject  The input object

depVariable  Dependent variable

texfile  The location of the .tex file. given Clean=FALSE

pdffile  The location of the final pdf file

...  The same as the input parameters

Author(s)

Hamed Haselimashhadi <chamedhm@ebi.ac.uk>

See Also

PhenList
Examples

```r
file <- system.file("extdata", "test1.csv", package = "PhenStat")
test = PhenStat:::PhenList(dataset = read.csv(file, na.strings = '-'),
                           testGenotype = "Sparc/Sparc")

## Not run:
PhenStatReport(test,
               depVariable = 'Bone.Area',
               open = TRUE)

## End(Not run)
```

PhenTestResult

**Method "PhenTestResult"**

Description

Creates `PhenTestResult` object based on test results or model building results. Internal function for the package. Not build for users to directly call.

Usage

```r
PhenTestResult(model.output=NULL, model.dataset=NULL,
                depVariable=NULL,
                refGenotype=NULL,
                equation="withWeight",
                method="MM", model.effect.batch=NULL,
                model.effect.variance=NULL, model.effect.sex=NULL,
                model.effect.interaction=NULL, model.output.interaction=NULL,
                model.effect.weight=NULL, numberSexes=NULL,pThreshold=0.05,
                model.formula.null=NULL,model.formula.genotype=NULL,
                model.output.genotype.nulltest.pVal=NULL,
                model.output.quality=NULL,model.output.summary=NULL,
                model.output.averageRefGenotype = NULL,
                model.output.percentageChanges = NULL)
```

Arguments

- `model.output` representing the model fit gls object or lme object
- `model.dataset` dataset used for modeling
- `depVariable` depending variable in the model
- `refGenotype` reference genotype in the model
- `equation` possible values: "withWeight" to include weight effect into model, "withoutWeight" to exclude weight effect from the model
- `method` possible values: "MM" to work with Mixed Model framework, "FE" to work with Fisher Exact Test framework
Value

Returns an instance of the `PhenTestResult` class.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

`PhenTestResult`
Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
    testGenotype="Sparc/Sparc")
# "testDataset" function calls "PhenTestResult" function internally
result <- PhenStat:::testDataset(test,
    equation="withoutWeight",
    depVariable="Lean.Mass")
class(result)
```

**PhenTestResult-class**  
*Class "PhenTestResult"*

**Description**

A list-based S4 class for storing of model fitting results and other information like dependent variable, method used to build a model, etc. Helps to support operations needed for model fitting process and modelling/testing results storage. In the package PhenTestResult object is created by function `testDataset`.

**Explore PhenTestResult object**

PhenTestResults object instance contains the following slots:

1. "analysedDataset" contains subdataset that was used for the modelling/testing:
   ```r
   analysedDataset(phenTestResult)
   ```
2. "depVariable" contains dependent variable that was tested:
   ```r
   getVariable(phenTestResult)
   ```
3. "refGenotype" contains reference genotype which is usually is wildtype:
   ```r
   refGenotype(phenTestResult)
   ```
4. "testGenotype" contains test genotype:
   ```r
   testGenotype(phenTestResult)
   ```
5. "method" contains method name that was used for modelling/testing:
   ```r
   method(phenTestResult)
   ```
6. "transformationRequired" contains TRUE/FALSE value indicating transofrmation of dependent variable:
   ```r
   transformationText(phenTestResult) or phenTestResult@transformationRequired
   ```
7. "lambdaValue" contains lambda value for the Box-Cox transformation:
   ```r
   transformationText(phenTestResult) or phenTestResult$lambdaValue
   ```
8. "scaleShift" contains the value of scale shift for Box-Cox transformation:
   ```r
   transformationText(phenTestResult) or phenTestResult@scaleShift
   ```
9. "transformationCode" contains the code of transformation. Possible values:
    0 - transformation is not applicable (methods "FE", "RR", "LR") or switched off by user
    1 - transformation is not needed (1 is within the 95
    2 - log transformation
    3 - power transformation
    4 - transformation is not performed since optimal lambda value is not found (-5 > lambda > 5)
   ```r
   transformationText(phenTestResult) or phenTestResult@transformationCode
   ```
10. "parameters" contains the parameters used during method application, e.g. pThreshold for MM and TF:

\[ \text{parameters(phenTestResult)} \]

11. Modelling/testing results are stored in the sections "analysisResults": \[ \text{analysisResults(phenTestResult)} \]

The contents of the analysisResults slot depend on framework that was used.

The results of MM and TF frameworks:
1. Equation used during linear modeling: \[ \text{analysisResults(phenTestResult)$equation} \]
2. Batch effect significance: \[ \text{analysisResults(phenTestResult)$model.effect.batch} \]
3. Variance equality: \[ \text{analysisResults(phenTestResult)$model.effect.variance} \]
4. Weight effect significance: \[ \text{analysisResults(phenTestResult)$model.effect.weight} \]
5. Sex effect significance: \[ \text{analysisResults(phenTestResult)$model.effect.interaction} \]
6. Evidence of sex effect (p-value): \[ \text{analysisResults(phenTestResult)$model.output.interaction} \]
7. Evidence of genotype effect (p-value): \[ \text{analysisResults(phenTestResult)$model.output.genotype.nulltest.pVal} \]
8. Formula for the final genotype model: \[ \text{analysisResults(phenTestResult)$model.formula.genotype} \]
9. Formula for the final null model: \[ \text{analysisResults(phenTestResult)$model.formula.null} \]
10. Model fitting output: \[ \text{analysisResults(phenTestResult)$model.output} \]
11. Model fitting summary: \[ \text{summary(analysisResults(phenTestResult)$model.output)$tTable} \]

The results of FE and RR frameworks are stored as a list of htestPhenStat S4 objects: \[ \text{analysisResults(phenTestResult)[[1]} \]

Each one htestPhenStat object contains:
1. Output of Fisher Exact Test: \[ \text{pvalue(analysisResults(phenTestResult)[[1]])} \]
2. Effect size: \[ \text{ES(analysisResults(phenTestResult)[[1]])} \]
3. Name of the table analysed (all, males, females): \[ \text{analysedSubset(analysisResults(phenTestResult)[[1]])} \]
4. Comparison, used for RR only (High vs Normal/Low, Low vs High/Normal): \[ \text{comparison(analysisResults(phenTestResult)[[1]])} \]
5. Matrix of counts: \[ \text{matrixCount(analysisResults(phenTestResult)[[1]])} \]

Slots

- \[ \text{analysedDataset: Object of class "data.frame" -- analysed dataset} \]
- \[ \text{transformationRequired: Object of class "logical" -- flag indicating was or not transformation performed} \]
- \[ \text{lambdaValue: Object of class "numeric" -- Box-Cox transform lambda value} \]
- \[ \text{scaleShift: Object of class "numeric" -- Box-Cox transform scale shift} \]
- \[ \text{transformationCode: Object of class "numeric" -- code explaining the transformation output} \]
- \[ \text{depVariable: Object of class "character" -- dependent variable} \]
- \[ \text{refGenotype: Object of class "character" -- reference genotype} \]
- \[ \text{testGenotype: Object of class "character" -- test genotype} \]
- \[ \text{method: Object of class "character" -- analysis method} \]
- \[ \text{parameters: Object of class "matrix" -- parameters of method} \]
- \[ \text{analysisResults: Object of class "list" -- results of the analysis} \]
Methods

getVariable (accessor): Returns dependent variable
refGenotype (accessor): Returns reference genotype
testGenotype (accessor): Returns test genotype
method (accessor): Returns method used
methodText (accessor): Returns full name of the method used
parameters (accessor): Returns parameters used during the method application
analysisResults (accessor): Returns analysis results depending on method
analysedDataset (accessor): Returns analysed dataset
transformationText (accessor): Returns the info about transformation
batchIn Returns TRUE if the batch is in the dataset, FALSE otherwise
weightIn Returns TRUE if the weight is in the dataset, FALSE otherwise
noSexes Returns number of sexes in the dataset
show Prints out the PhenTestResult object
getCountMatrices Returns count matrices if they are present (for methods "FE" and "RR"), NULL otherwise.

Author(s)
Natalja Kurbatova

Examples

# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file, na.strings = '-'),
testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
equation="withoutWeight",
depVariable="Bone.Area")
PhenStat:::getVariable(result)
PhenStat:::method(result)
# Batch effect is significant
PhenStat:::analysisResults(result)$model.effect.batch
# Variance homogeneity
PhenStat:::analysisResults(result)$model.effect.variance
# Weight effect is significant
PhenStat:::analysisResults(result)$model.effect.weight
# Sex effect is significant
PhenStat:::analysisResults(result)$model.effect.interaction
# Sex effect p-value - the result of the test
PhenStat:::analysisResults(result)$model.output.interaction
# Genotype effect p-value
PhenStat:::analysisResults(result)$model.output.genotype.nulltest.pVal
# Final model formula with genotype
PhenStat:::analysisResults(result)$model.formula.genotype
# Final model formula without genotype
PhenStat:::analysisResults(result)$model.formula.null
# Final model fitting output
# result$model.output
# Final model fitting summary
# summary(result$model.output)$tTable

# Fisher Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
testGenotype="Aff3/Aff3")
result <- PhenStat:::testDataset(test,
    depVariable="Thoracic.Processes",
    method="FE")
PhenStat:::getVariable(result)
PhenStat:::method(result)
for (i in seq_along(analysisResults(result))) {
    val <- PhenStat:::analysisResults(result)[[i]]
    val
}

---

**plot.PhenList**  
*Plot Phenlist object*

**Description**

Plot method for objects of class "PhenList".

**Usage**

```r
## S3 method for class 'PhenList'
plot(
    x,
    depVariable = 'Value',
    graphingName = NULL,
    outputMessages = TRUE,
    type = NULL,
    ...
)
```

**Arguments**

- `x`  
  instance of the PhenList class mandatory argument
- `depVariable`  
  a character string defining the dependent variable of interest mandatory argument
- `graphingName`  
  character string defining the name to be used on the graph for the dependent variable
outputMessages
flag: "FALSE" value to suppress output messages "TRUE" value to show output messages default value TRUE

type
a vector of names. Select one or more from the list below to only get those plots.

- boxplotSexGenotype. See documentations for boxplotSexGenotype
- boxplotSexGenotypeBatchAdjusted. See documentations for boxplotSexGenotypeBatchAdjusted
- boxplotSexGenotypeWeightBatchAdjusted. See documentations for boxplotSexGenotypeWeightBatchAdjusted
- scatterplotSexGenotypeBatch. See documentations for scatterplotSexGenotypeBatch
- scatterplotGenotypeWeight. See documentations for scatterplotGenotypeWeight
- boxplotSexGenotypeBatch. See documentations for boxplotSexGenotypeBatch

Value
Empty.

Author(s)
Hamed Haselimashhadi < hamedhm@eb.ac.uk >

See Also
PhenList.

Examples
example(testDataset)

---

plot.PhenTestResult  Plot phenTestResult object

Description
Plot method for objects of class "phenTestResult".
Usage

```r
## S3 method for class 'PhenTestResult'
plot(
  x,
  graphingName = NULL,
  outputMessages = TRUE,
  type = NULL,
  ...
)
```

Arguments

- `x` instance of the PhenTestResult class mandatory argument
- `graphingName` character string defining the name to be used on the graph for the dependent variable
- `outputMessages` flag: "FALSE" value to suppress output messages "TRUE" value to show output messages default value TRUE
- `type` a vector of names. Select one or more from the list below to only get those plots.
  - boxplotSexGenotypeResult. See documentations for boxplotSexGenotypeResult
  - scatterplotSexGenotypeBatchResult. See documentations for scatterplotSexGenotypeBatchResult
  - scatterplotGenotypeWeightResult. See documentations for scatterplotGenotypeWeightResult
  - plotResidualPredicted. See documentations for plotResidualPredicted
  - qqplotRandomEffects. See documentations for qqplotRandomEffects
  - boxplotResidualBatch. See documentations for boxplotResidualBatch
  - qqplotRotatedResiduals. See documentations for qqplotRotatedResiduals
  - qqplotGenotype. See documentations for qqplotGenotype
  - categoricalBarplot. See documentations for categoricalBarplot

Value

Empty.

Author(s)

Hamed Haselimashhadi < hamedhm@eb.ac.uk >
See Also
  testDataset

Examples
  example(testDataset)

---

plotResidualPredicted Method "plotResidualPredicted"

Description
  Graph function for the Mixed Model framework’s results. Creates predicted versus residual values plots split by genotype.

Usage
  plotResidualPredicted(phenTestResult, outputMessages=TRUE)

Arguments
  phenTestResult  instance of the PhenTestResult class that comes from the function testDataset; mandatory argument
  outputMessages  flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value
  Empty.

Author(s)
  Natalja Kurbatova, Natasha Karp, Jeremy Mason

References

See Also
  PhenList and PhenTestResult
printLROutput

### Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-'"),
  testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
  depVariable="Lean.Mass")
PhenStat:::plotResidualPredicted(result)
```

### Method “printLROutput”

#### Description

Prints out for linear regression output

#### Usage

```r
printLROutput(phenTestResult,phenotypeThreshold=0.01)
```

#### Arguments

- `phenTestResult`: instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `effectValues`: a vector with genotype effect and standard error for the linear regression; mandatory argument
- `phenotypeThreshold`: a numerical value defining the threshold to use in `classificationTag`; default value 0.01

#### Value

Prints out the common part of linear regression output.

#### Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

#### References


#### See Also

`PhenTestResult`
Examples

```r
file <- system.file("extdata", "testLR.csv", package="PhenStat")
testLR <- PhenStat::PhenList(dataset=read.csv(file,na.strings = '-'),
    testGenotype="Mock")
testLR2 <- PhenStat::LRDataset(testLR,"V2")
resultLR <- PhenStat::testDataset(testLR2,
    depVariable="V2",
    method="LR")
PhenStat::printLROutput(resultLR)
```

---

**printTabStyle**

Method "printTabStyle"

Description

Makes ready list of strings to be printed in tabular form, creates a table row from it. Adds empty positions if needed in order to make all strings the same by length and adds "tabSep" character between strings. Returns text that is ready to be printed (one row of a table).

Usage

```r
printTabStyle(textList,positions,tabSep="|")
```

Arguments

- `textList`: list of strings to prepare
- `positions`: desired length of one cell in the table
- `tabSep`: character used to separate columns in the table; default value is "|

Value

Returns text that is ready to be printed (one row of a table).

Author(s)

Natalja Kurbatova

Examples

```r
row_sep <- rep("----------",3)
print(PhenStat:::printTabStyle(row_sep,11))
list1 <- c("Owner","Model","Cc")
print(PhenStat:::printTabStyle(list1,11))
print(PhenStat:::printTabStyle(row_sep,11))
list2 <- c("Natalja","Honda","125")
print(PhenStat:::printTabStyle(list2,11))
print(PhenStat:::printTabStyle(row_sep,11))
```
**pvalue** | Method "pvalue"
---|---

**Description**
Method `pvalue` returns p-value that was calculated by RR or FE framework.

**Value**
This function returns p-value that was calculated by RR or FE framework.

**Methods**
signature(obj = "htestPhenStat")

**qqplotGenotype** | Method "qqplotGenotype"
---|---

**Description**
Graph function for the Mixed Model framework’s results. Creates a Q-Q plot of residuals for each genotype.

**Usage**

```r
qqplotGenotype(phenTestResult, outputMessages=TRUE)
```

**Arguments**
- `phenTestResult` instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**
Empty.

**Author(s)**
Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenList and PhenTestResult

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                           testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
                                depVariable="Lean.Mass")
PhenStat:::qqplotGenotype(result)
```

**Description**

Graph function for the Mixed Model framework's results. Creates a Q-Q plot of blups (best linear unbiased predictions).

**Usage**

```r
qqplotRandomEffects(phenTestResult, outputMessages=TRUE)
```

**Arguments**

- **phenTestResult**: instance of the PhenTestResult class that comes from the function `testDataset`; mandatory argument
- **outputMessages**: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Empty.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

*PhenList* and *PhenTestResult*

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
  testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
  depVariable="Lean.Mass")
PhenStat:::qqplotRandomEffects(result)
```

```
qqplotRotatedResiduals

*Method "qqplotRotatedResiduals"*

Description

Graph function for the Mixed Model framework's results. Creates a Q-Q plot of rotated residuals.

Usage

```r
qqplotRotatedResiduals(phenTestResult,outputMessages=TRUE)
```

Arguments

- `phenTestResult` instance of the *PhenTestResult* class that comes from the function *testDataset*; mandatory argument
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

Value

Empty.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason
**recommendMethod**  

**Method** "recommendMethod"

**Description**

Checks the dependent variable and dataset to make choose the appropriate statistical method. Returns the method or the list of methods that are appropriate for statistical analysis.

**Usage**

```r
recommendMethod(phenList = NULL, depVariable = NULL, outputMessages = TRUE, upper = 5)
```

**Arguments**

- **phenList**: instance of the `PhenList` class; mandatory argument
- **depVariable**: a character string defining the dependent variable of interest; mandatory argument
- **outputMessages**: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
- **upper**: Single integer: maximum allowed number of levels for the Batch in the TF framework

**References**


**See Also**

`PhenList` and `PhenTestResult`

**Examples**

```r
file <- system.file("extdata", "test2.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
dataset.colname.weight="Weight.Value",
testGenotype="Arid4a/Arid4a",
hemiGenotype="Arid4a/+"
result <- PhenStat:::testDataset(test,
depVariable="Cl")
PhenStat:::qqplotRotatedResiduals(result)
```
refGenotype

**Value**

Returns the method or the list of methods that are appropriate for statistical analysis.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

PhenList

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
                           testGenotype="Sparc/Sparc")
PhenStat:::recommendMethod(test,"Lean.Mass")
```

---

**refGenotype**

*Method "refGenotype"*

**Description**

Method *refGenotype* returns reference genotype

**Value**

This function returns reference genotype.

**Methods**

```r
signature(obj = "PhenList")
signature(obj = "PhenTestResult")
```
RRTest

Method "RRTest"

Description
This is an internal function run within RR framework. It performs Reference Ranges Plus test and after that Fisher Exact test on calculated count matrices. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.
Works with PhenList object created by PhenList function.

Usage
RRTest(
phenList,
depVariable,
outputMessages = TRUE,
naturalVariation = 95,
controlPointsThreshold = 60
)

Arguments
phenList instance of the PhenList class; mandatory argument
depVariable a character string defining the dependent variable of interest; mandatory argument
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
naturalVariation number defining the natural variation range in percents for normal values; default value 95
controlPointsThreshold number defining how many control points is needed to perform RR plus method; default value 60

Value
Returns results stored in instance of the PhenTestResult class

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References
**scatterplotGenotypeWeight**

**Method "scatterplotGenotypeWeight"**

**Description**

Graph function for the phenotypic dataset. Creates a scatter plot body weight versus dependent variable. Both a regression line and a loess line (locally weighted line) is fitted for each genotype.

**Usage**

```r
scatterplotGenotypeWeight(
  phenList, 
  depVariable = NULL, 
  graphingName = NULL, 
  outputMessages = TRUE
)
```

**Arguments**

- `phenList` instance of the `PhenList` class; mandatory argument
- `depVariable` a character string defining the dependent variable of interest; mandatory argument
- `graphingName` a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Empty.

---

**See Also**

- `PhenList` and `testDataset`

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                           testGenotype="Sparc/Sparc")
# "RRTest" function is called from "testDataset" function
result <- PhenStat:::testDataset(test,
                                  depVariable="Lean.Mass",
                                  method="RR")
# RR test results can be printed out using function "summaryOutput"
summary(result)
```
scatterplotGenotypeWeightResult

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
                           testGenotype="Sparc/Sparc")
PhenStat:::scatterplotGenotypeWeight(test,
                                  depVariable="Bone.Mineral.Content",
                                  graphingName="BMC")
file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
                                  testGenotype="Mysm1/+")
PhenStat:::scatterplotGenotypeWeight(test_1sex,
                                  depVariable="Lean.Mass",
                                  graphingName="Lean Mass")
```

scatterplotGenotypeWeightResult

*Method "scatterplotGenotypeWeightResult"

Description

Graph function for the phenotypic dataset. Creates a scatter plot body weight versus dependent variable. It is based on PhenTestResult object. Both a regression line and a loess line (locally weighted line) is fitted for each genotype.

Usage

```r
scatterplotGenotypeWeightResult(phenTestResult,
                               graphingName = NULL,
                               outputMessages = TRUE)
```
**scatteredGenotypeWeightResult**

**Arguments**

- **phenTestResult**: instance of the `PhenTestResult` class; mandatory argument
- **graphingName**: a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- **outputMessages**: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Empty.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

`PhenList`

**Examples**

```r
define file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat::PhenList(dataset=read.csv(file,na.strings = "-"),
  testGenotype="Sparc/Sparc")
result <- PhenStat::testDataset(test,
  depVariable="Lean.Mass")
PhenStat:::scatteredplotGenotypeWeightResult(result,
  graphingName="BMC")
define file <- system.file("extdata", "test4.csv", package="PhenStat")
test_lsex <- PhenStat::PhenList(dataset=read.csv(file,na.strings = "-"),
  testGenotype="Mysm1/+")
result_lsex <- PhenStat::testDataset(test_lsex,
  depVariable="Lean.Mass")
PhenStat:::scatteredplotGenotypeWeightResult(result_lsex,
  graphingName="Lean Mass")
```
**Description**

Graph function for the phenotypic dataset. Creates a scatterplot split by sex, genotype and batch. refGenotype data points are shown in black and the testGenotype data points are shown in red.

Note: the batches are not ordered with time but allow assessment of how the testGenotype data lie relative to the variation within the refGenotype data.

**Usage**

```r
scatterplotSexGenotypeBatch(
  phenList,
  depVariable = NULL,
  graphingName = NULL,
  outputMessages = TRUE
)
```

**Arguments**

- `phenList`: instance of the `PhenList` class; mandatory argument
- `depVariable`: a character string defining the dependent variable of interest; mandatory argument
- `graphingName`: a character string defining the name to be used on the graph for the dependent variable; mandatory argument
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Empty.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


scatterplotSexGenotypeBatchResult

See Also
PhenList

Examples

```
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
    testGenotype="Sparc/Sparc")
# box plot for dataset with two sexes: males and females
PhenStat:::scatterplotSexGenotypeBatch(test,
    depVariable="Bone.Mineral.Content",
    graphingName="BMC" )

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
    testGenotype="Mysml/+")
# box plot for females only dataset
PhenStat:::scatterplotSexGenotypeBatch(test_1sex,depVariable="Lean.Mass",
    graphingName="Lean Mass")
```

scatterplotSexGenotypeBatchResult

*Method* "scatterplotSexGenotypeBatchResult"

Description
Graph function for the phenotypic dataset. Creates a scatterplot split by sex, genotype and batch. refGenotype data points are shown in black and the testGenotype data points are shown in red. It is based on PhenTestResult object.

Note: the batches are not ordered with time but allow assessment of how the testGenotype data lie relative to the variation within the refGenotype data.

Usage

```
scatterplotSexGenotypeBatchResult(phenTestResult,
    graphingName = NULL,
    outputMessages = TRUE)
```

Arguments

phenTestResult instance of the *PhenTestResult* class; mandatory argument
graphingName a character string defining the name to be used on the graph for the dependent variable; mandatory argument
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
setBatch

Value

Empty.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
                         testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
                         depVariable="Lean.Mass")
# box plot for dataset with two sexes: males and females
PhenStat:::scatterplotSexGenotypeBatchResult(result,
                         graphingName="BMC")

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '\-'),
                         testGenotype="Mysm1/+")
result_1sex <- PhenStat:::testDataset(test_1sex,
                         depVariable="Lean.Mass")
# box plot for females only dataset
PhenStat:::scatterplotSexGenotypeBatchResult(result_1sex,
                         graphingName="Lean Mass")
```

---

### setBatch

**Method "setBatch"**

**Description**

Method `setBatch` sets batch column in the dataset

**Value**

This function sets batch column in the dataset.
setSex

**Methods**

signature(obj = "PhenList")

---

**setGenotype**

*Method "setGenotype"*

**Description**

Method setGenotype sets genotype column in the dataset

**Value**

This function sets genotype column in the dataset.

**Methods**

signature(obj = "PhenList")

---

**setMissingValue**

*Method "setMissingValue"*

**Description**

Method setMissingValue sets missing value in the dataset

**Value**

This function sets missing value in the dataset.

**Methods**

signature(obj = "PhenList")

---

**setSex**

*Method "setSex"*

**Description**

Method setSex sets sex column in the dataset

**Value**

This function sets sex column in the dataset.

**Methods**

signature(obj = "PhenList")
**setWeight**

*Method "setWeight"*

**Description**

Method `setWeight` sets weight column in the dataset.

**Value**

This function sets weight column in the dataset.

**Methods**

`signature(obj = "PhenList")`

**startLRModel**

*Method "startLRModel"*

**Description**

This is an internal function run within LR framework. It completes the testing stage of which effects are significant. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.

It creates start model and modify it after testing of different hypothesis.

The tested effects are:
- batch effect (TRUE if batch variation is significant, FALSE if not), though it is never used in final LR model;
- interaction effect (TRUE if genotype by sex interaction is significant),
- sex effect (TRUE if sex is significant),
- weight effect and variance effect are not tested and used in LR model.

**Usage**

```
startLRModel(phenList, depVariable, outputMessages = TRUE, pThreshold = 0.05)
```
Arguments

phenList instance of the PhenList class; mandatory argument

depVariable a character string defining the dependent variable of interest; mandatory argument

outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

pThreshold a numerical value for the p-value threshold used to determine which fixed effects to keep in the model, default value 0.05

Value

Returns results stored in instance of the PhenTestResult class

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

file <- system.file("extdata", "testLR.csv", package="PhenStat")
testLR <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'), testGenotype="Mock")
testLR2 <- PhenStat:::LRDataset(testLR,"V2")

# when "testDataset" function's argument "callAll" is set to FALSE
# only "startLRModel" function is called - the first step of LR framework
resultLR <- PhenStat:::testDataset(testLR2,
  depVariable="V2",
  callAll=FALSE,
  method="LR")

# print out formula that has been created
PhenStat:::analysisResults(resultLR)$model.formula.genotype
# print out batch effect's significance
PhenStat:::analysisResults(resultLR)$model.effect.batch
Method "startModel"

Description

This is an internal function run within MM framework. It completes the testing stage of which effects are significant. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.

It creates start model and modify it after testing of different hypothesis (the model fixed effects).

The model random effects are:
- batch effect (TRUE if batch variation is significant, FALSE if not)

The model fixed effects are:
- variance effect (TRUE if residual variances for genotype groups are homogeneous and FALSE if they are heterogeneous),
- interaction effect (TRUE if genotype by sex interaction is significant),
- sex effect (TRUE if sex is significant),
- weight effect (TRUE if weight is significant).

If user would like to assign other TRUE/FALSE values to the fixed effects of the model then he or she has to define keepList argument which is vector of TRUE/FALSE values.

If user has defined model fixed effects (keepList argument) then function prints out calculated and user defined effects (only when outputMessages argument is set to TRUE), checks user defined effects for consistency (for instance, if there are no "Weight" column in the dataset then weight effect can’t be assigned to TRUE, etc.) and modifies start model according to user defined effects.

As the result PhenTestResult object that contains calculated or user defined model fixed effects and MM start model is created.

Usage

```r
startModel(phenList, depVariable, 
  equation="withWeight", outputMessages=TRUE, 
  pThreshold=0.05, keepList=NULL, modelWeight = NULL, 
  threshold = 10^-18, 
  check = 1)
```

Arguments

- `phenList` instance of the `PhenList` class; mandatory argument
- `depVariable` a character string defining the dependent variable of interest; mandatory argument
- `equation` a character string defining the equation to use. Possible values "withWeight" (default),"withoutWeight"
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
$startModel$

- **pThreshold**: A numerical value for the p-value threshold used to determine which fixed effects to keep in the model, default value 0.05
- **keepList**: A logical vector defining the significance of different model effects: keep_batch, keep_equalvar, keep_weight, keep_sex, keep_interaction; default value NULL
- **modelWeight**: A vector of positive values for weights in the mixed model. The sum of the values must be one.
- **threshold**: A single positive value. The threshold for the ModelWeights to consider as zero (see modelWeight)
- **check**: Only useful when modelWeight is included. Select between 0, 1, 2 to impose different weighting strategies on the Linear Mixed model. check=1 (default) selects the weights that are greater than the threshold (above) and keeps the batches that include more than one single sample. check=2 keeps only the weights that are greater than the threshold but ignores the single sample batches. check=0 to disable the function.

**Value**

Returns results stored in instance of the **PhenTestResult** class

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

**PhenList**

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file, na.strings = '-'),
                           testGenotype="Sparc/Sparc")
# when "testDataset" function's argument "callAll" is set to FALSE
# only "startModel" function is called - the first step of MM framework
result <- PhenStat:::testDataset(test,
                                  depVariable="Lean.Mass",
                                  callAll=FALSE)
# print out formula that has been created
PhenStat:::analysisResults(result)$model.formula.genotype
# print out batch effect's significance
PhenStat:::analysisResults(result)$model.effect.batch
# change the model
```
**Description**

This is an internal function run within TF framework. It completes the testing stage of which effects are significant. As an internal function, it doesn’t include extensive error testing of inputs. Please use cautiously if calling directly.

It creates start model and modify it after testing of different hypothesis.

The tested fixed effects are:

- batch effect (TRUE if batch variation is significant, FALSE if not),
- variance effect (TRUE if residual variances for genotype groups are homogeneous and FALSE if they are heterogeneous),
- interaction effect (TRUE if genotype by sex interaction is significant),
- sex effect (TRUE if sex is significant),
- weight effect (TRUE if weight is significant).

**Usage**

```r
startTFModel(phenList, depVariable, 
edepVariable="Lean.Mass", 
equation="withWeight", 
callAll=FALSE)
```

**Arguments**

- `phenList` instance of the `PhenList` class; mandatory argument
- `depVariable` a character string defining the dependent variable of interest; mandatory argument
- `equation` a character string defining the equation to use. Possible values "withWeight" (default),"withoutWeight"
- `outputMessages` flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages ; default value TRUE
- `pThreshold` a numerical value for the p-value threshold used to determine which fixed effects to keep in the model, default value 0.05
- `keepList` a logical vector defining the significance of different model effects: keep_batch, keep_equalvar, keep_weight, keep_sex, keep_interaction; default value NULL
Value

Returns results stored in instance of the `PhenTestResult` class

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test7_TFE.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
                           testGenotype="het",
                           refGenotype = "WT",
                           dataset.colname.sex="sex",
                           dataset.colname.genotype="Genotype",
                           dataset.values.females="f",
                           dataset.values.male= "m",
                           dataset.colname.weight="body.weight",
                           dataset.colname.batch="Date_of_procedure_start")

test_TF <- PhenStat:::TFDataset(test,depVariable="Cholesterol")

# when "TFDataset" function's argument "callAll" is set to FALSE
# only "startTFModel" function is called - the first step of TFE framework
result <- PhenStat:::testDataset(test_TF,
                                  depVariable="Cholesterol",
                                  callAll=FALSE,
                                  method="TF")

# print out formula that has been created
PhenStat:::analysisResults(result)$model.formula.genotype

# print out batch effect's significance
PhenStat:::analysisResults(result)$model.effect.batch
```
**subsetText**

*Method "subsetText"*

**Description**

Method `subsetText` returns full name of subset that was analysed by RR or FE framework: Males only, Females only, All.

**Value**

This function returns full name of subset that was analysed by RR or FE framework.

**Methods**

signature(obj = "htestPhenStat")

**summary.PhenTestResult**

*summary*

**Description**

Wrapper for the model fitting results. Returns model fitting and testing results in a user friendly format.

**Usage**

```r
## S3 method for class 'PhenTestResult'
summary(object, phenotypeThreshold = 0.01,...)
```

**Arguments**

- `object` instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `phenotypeThreshold` a numerical value defining the threshold to use in `classificationTag`; default value 0.01
- ... Not applicable

**Value**

Returns model fitting results in a text format for the screen

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

*PhenTestResult*

Examples

```r
# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                           testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
                                  depVariable="Lean.Mass")
summary(result)

# Fishe Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test2 <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                            testGenotype="Aff3/Aff3")
result2 <- PhenStat:::testDataset(test2,
                                  depVariable="Thoracic.Processes",
                                  method="FE")
summary(result2)
```

summaryOutput

Method "summaryOutput"

Description

Wrapper for the model fitting results. Returns model fitting and testing results in a user friendly format.

Usage

```r
summaryOutput(phenTestResult, phenotypeThreshold = 0.01)
```

Arguments

- `phenTestResult` instance of the *PhenTestResult* class that comes from the function *testDataset*; mandatory argument
- `phenotypeThreshold` a numerical value defining the threshold to use in classificationTag; default value 0.01
Value

Returns model fitting results in a text format for the screen

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

*PhenTestResult*

Examples

```r
# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-',
    testGenotype="Sparc/Sparc"))
result <- PhenStat:::testDataset(test,
    depVariable="Lean.Mass")
PhenStat:::summaryOutput(result)

# Fishe Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test2 <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
    testGenotype="Aff3/Aff3")
result2 <- PhenStat:::testDataset(test2,
    depVariable="Thoracic.Processes",
    method="FE")
PhenStat:::summaryOutput(result2)
```

---

**Description**

Statistical analysis manager function in PhenStat package.

Firstly, it performs the checks for dependent variable in the dataset. Some checks are generic, some depends on selected statistical framework/method.

Secondly, if checks are passed it runs the stages of selected framework to analyse dependent variable.
For instance, runs startModel and finalModel for the MM framework if the argument "callAll" is set to TRUE. If framework contains only one stage like in Fisher Exact Test case then runs that one stage regardless "callAll" value.

Usage

testDataset(
    phenList = NULL,
    depVariable = NULL,
    equation = "withWeight",
    outputMessages = TRUE,
    pThreshold = 0.05,
    method = "MM",
    modelWeight = NULL,
    callAll = TRUE,
    keepList = NULL,
    dataPointsThreshold = 4,
    RR_naturalVariation = 95,
    RR_controlPointsThreshold = 60,
    transformValues = FALSE,
    useUnfiltered = FALSE,
    threshold = 10^-18,
    check = 1,
    upper = 5
)

Arguments

phenList instance of the PhenList class; mandatory argument
depVariable a character string defining the dependent variable of interest; mandatory argument
equation a character string defining the equation to use. Possible values "withWeight" (default), "withoutWeight"
outputMessages flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE
pThreshold a numerical value for the p-value threshold used to determine which fixed effects to keep in the model, default value 0.05
method a character string ("MM", "FE", "TF" or "RR") defining the method to use for model building; default value "MM" for mixed model
modelWeight a vector of possible weights (the same length as the response) that are imposed on the weighted MM. Only works when method="MM". The default is NULL that leads to normal (non-weighted) MM.
callAll flag: "FALSE" value to run only the first stage of the selected framework; "TRUE" value (default) to run all stages of framework one after another
keepList a logical vector defining the significance of different model effects: keep_batch, keep_equalvar, keep_weight, keep_sex, keep_interaction; default value NULL
dataPointsThreshold
threshold for the number of data points in the MM framework; default value is 4; minimal value is 2

RR_naturalVariation
threshold for the variation ranges in the RR framework; default value is 95; minimal value is 60

RR_controlPointsThreshold
threshold for the number of control data points in the RR framework; default value is 60; minimal value is 40

transformValues
flag: "FALSE" value to suppress transformation; "TRUE" value to perform transformation if needed; default value FALSE

useUnfiltered
flag: "FALSE" value to use filtered dataset; "TRUE" value to use unfiltered dataset; default value FALSE

threshold
The threshold for the model weights to be considered as zero. It only works when method="MM" and modelWeight is not null. Default is 10^-18.

check
Only useful when modelWeight is included. Select between 0, 1, 2 to impose different weighting strategies on the Linear Mixed model. check=1 (default) selects the weights that are greater than the threshold (above) and keeps the batches that include more than one single sample. check=2 keeps only the weights that are greater than the threshold but ignores the single sample batches. check=0 to disable the function.

upper
Single integer: maximum allowed number of levels for the Batch in the TF framework. Default 5

Value
Returns results stored in instance of the PhenTestResult class

Author(s)
Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also
PhenList, plot.PhenTestResult, plot.PhenList
Examples

# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
testGenotype="Sparc/Sparc")
plot(test,depVariable="Lean.Mass")

result <- PhenStat:::testDataset(test,
depVariable="Lean.Mass")
# print out formula that has been created
PhenStat:::analysisResults(result)$model.formula.genotype
summary(result)
plot(result)

# Mixed Model framework with user defined effects
user_defined_effects <- c(keep_batch=TRUE,
    keep_equalvar=TRUE,
    keep_weight=TRUE,
    keep_sex=TRUE,
    keep_interaction=TRUE)
result3 <- PhenStat:::testDataset(test,
depVariable="Lean.Mass",
    keepList=user_defined_effects)
# print out formula that has been created
PhenStat:::analysisResults(result3)$model.formula.genotype
summary(result3)
plot(result3)

# Fisher Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test2 <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
    testGenotype="Aff3/Aff3")
plot(test2,depVariable="Thoracic.Processes")
result2 <- PhenStat:::testDataset(test2,
    depVariable="Thoracic.Processes",
    method="FE")
summary(result2)
plot(result2)
Description

Performs diagnostic tests for Logistic Regression model quality of fit. There are no arguments checks assuming that function is called internally from the "finalLRModel" function. Otherwise should be used with precaution.

Usage

testFinalLRModel(phenTestResult)

Arguments

phenTestResult instance of the PhenTestResult class that comes from the function testDataset ; mandatory argument

Value

Returns a vector with model fitting results

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenTestResult

Examples

file <- system.file("extdata", "testLR.csv", package="PhenStat")
testLR <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '"'),
  testGenotype="Mock")
testLR2 <- PhenStat:::LRDataset(testLR,"V2")
resultLR <- PhenStat:::testDataset(testLR2,
  depVariable="V2",
  method="LR")
PhenStat:::testFinalLRModel(resultLR)
testFinalModel  

Method "testFinalModel"

Description

Performs diagnostic tests for Mixed Model quality of fit. There are no arguments checks assuming that function is called internally from the "finalModel" function. Otherwise should be used with precaution.

Usage

```
testFinalModel(phenTestResult)
```

Arguments

- `phenTestResult` instance of the `PhenTestResult` class that comes from the function testDataset; mandatory argument

Value

Returns a vector with model fitting results

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

`PhenTestResult`

Examples

```
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-")
 testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test, depVariable="Lean.Mass")
PhenStat:::testFinalModel(result)
```
**testGenotype**

*Method "testGenotype"

**Description**

Method testGenotype returns test genotype

**Value**

This function returns test genotype.

**Methods**

```r
signature(obj = "PhenList")
signature(obj = "PhenTestResult")
```

---

**TFDataset**

*Method "TFDataset"

**Description**

Removes all non-concurrent batches from the dataset. Returns dataset suitable for TF framework or original dataset if data removal was not successful.

**Usage**

```r
TFDataset(phenList, depVariable, outputMessages = TRUE, forDecisionTree = FALSE, upper = 5)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phenList</td>
<td>instance of the PhenList class; mandatory argument</td>
</tr>
<tr>
<td>depVariable</td>
<td>a character string defining the dependent variable of interest; mandatory argument</td>
</tr>
<tr>
<td>outputMessages</td>
<td>flag: &quot;FALSE&quot; value to suppress output messages; &quot;TRUE&quot; value to show output messages; default value TRUE</td>
</tr>
<tr>
<td>forDecisionTree</td>
<td>flag: &quot;FALSE&quot; value for normal procedure; &quot;TRUE&quot; value to indicate that is called form decisionTree function; default value FALSE</td>
</tr>
<tr>
<td>upper</td>
<td>Single integer: maximum allowed number of levels for the Batch in the TF framework. Default 5.</td>
</tr>
</tbody>
</table>
transformation

Value

Returns dataset suitable for TF framework or original dataset if data removal was not successful.

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file, na.strings = '-'),
   testGenotype="Sparc/Sparc")
test2 <- PhenStat:::TFDataset(test, "Lean.Mass")
```

---

transformation  Method "transformation"

Description

Method transformation returns transformation values: lamda=value, scaleShift=value

Value

This function returns transformation values.

Methods

signature(obj = "PhenTestResult")
transformationJSON

Method "transformationJSON"

Description
Method transformationJSON returns transformation values as JSON string.

Value
This function returns transformation values as JSON string.

Methods
signature(obj = "PhenTestResult")

transformationText

Method "transformationText"

Description
Method transformationText returns transformation values as one string.

Value
This function returns transformation values as one string.

Methods
signature(obj = "PhenTestResult")

vectorOutput

Method "vectorOutput"

Description
Wrapper for the model fitting results. Returns model fitting and testing results in a vector format. Assumes that modelling results are stored in the PhenTestResult object (output from function testDataset).
Usage

vectorOutput(
  phenTestResult,
  phenotypeThreshold = 0.01,
  othercolumns = NULL,
  quote = '
')

Arguments

phenTestResult instance of the PhenTestResult class that comes from the function testDataset; mandatory argument

phenotypeThreshold a numerical value defining the threshold to use in classificationTag; default value 0.01

othercolumns a vector of column names that needs to be included in the function output

quote The character that is used to encompass the JSON keys in summary statistics. The default is quotation

Value

Returns a vector with model fitting results: model output summary and other values

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenTestResult

Examples

# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = '-'),
                          testGenotype="Sparc/Sparc")
result <- PhenStat:::testDataset(test,
                                depVariable="Lean.Mass")
PhenStat:::vectorOutput(result)

# Fishe Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test2 <- PhenStat:::PhenList(dataset=read.csv(file,na.strings = "-"),
  testGenotype="Aff3/Aff3")
result2 <- PhenStat:::testDataset(test2,
  depVariable="Thoracic.Processes",
  method="FE")
PhenStat:::vectorOutput(result2)

---

**vectorOutputMatrices**  Method "vectorOutputMatrices"

**Description**

Returns count matrices from Fisher Exact Test framework in a vector form.

Note: Maximal supported number of dependent variable levels is 10.

**Usage**

`vectorOutputMatrices(phenTestResult, outputMessages=TRUE)`

**Arguments**

- `phenTestResult`: instance of the `PhenTestResult` class that comes from the function `testDataset`; mandatory argument
- `outputMessages`: flag: "FALSE" value to suppress output messages; "TRUE" value to show output messages; default value TRUE

**Value**

Returns a vector with count values.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

`PhenTestResult`
Examples

```r
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test <- PhenStat:::PhenList(dataset=read.csv(file, na.strings = '\-'),
                          testGenotype="Aff3/Aff3")
result <- PhenStat:::testDataset(test,
                               depVariable="Thoracic.Processes",
                               method="FE")
PhenStat:::vectorOutputMatrices(result)
```

Description

Returns TRUE if the weight is in the dataset, FALSE otherwise

Value

TRUE/FALSE

Methods

signature(obj = "PhenList")
signature(obj = "PhenTestResult")
Index

* classes
  htest-class, 33
  htestPhenStat-class, 34
  LargeDataObject-class, 36
  PhenList-class, 49
  PhenTestResult-class, 55

* internal
  columnChecks, 17
  columnLevels, 18
  determiningLambda, 19
  dim, 20
  formulaDeterminingLambda, 27
  htest-class, 33
  htestPhenStat-class, 34
  LargeDataObject-class, 36
  modelFormula, 39
  modelFormulaLR, 40
  performReverseTransformation, 45
  performTransformation, 46
  printLROutput, 62
  printTabStyle, 63

* methods
  analysedDataset, 4
  analysedDatasetPhenList, 4
  analysedSubset, 5
  analysisResults, 5
  batchIn, 6
  comparison, 19
  dataset, 19
  getColumn, 28
  getColumnBatchAdjusted, 29
  getColumnView, 29
  getColumnWeightBatchAdjusted, 29
  getCountMatrices, 30
  getDataset, 30
  getGenotypeEffect, 30
  getMatrix, 31
  getPercentageMatrix, 31
  getStat, 31
  getVariable, 32
  getVariables, 32
  hemiGenotype, 32
  htestPhenStat-class, 34
  matrixCount, 38
  method, 38
  methodText, 39
  multipleBatches, 41
  noSexes, 42
  parameters, 42
  PhenList-class, 49
  PhenTestResult-class, 55
  pvalue, 64
  refGenotype, 68
  setBatch, 75
  setGenotype, 76
  setMissingValue, 76
  setSex, 76
  setWeight, 77
  subsetText, 83
  testGenotype, 91
  transformation, 92
  transformationJSON, 93
  transformationText, 93
  weightIn, 96

analysedDataset, 4
analysedDataset,PhenTestResult-method
(PhenTestResult-class), 55
analysedDatasetPhenList, 4
analysedSubset, 5
analysedSubset,htestPhenStat-method
(htestPhenStat-class), 34
analysisResults, 5
analysisResults,PhenTestResult-method
(PhenTestResult-class), 55
batchIn, 6
batchIn,PhenList-method (batchIn), 6
batchIn, PhenTestResult-method
(batchIn), 6
batchIn-methods (batchIn), 6
boxplotResidualBatch, 6
boxplotSexGenotype, 7
boxplotSexGenotypeBatch, 8
boxplotSexGenotypeBatchAdjusted, 10
boxplotSexGenotypeResult, 11
boxplotSexGenotypeWeightBatchAdjusted, 12
categoricalBarplot, 13
checkDataset, 14, 15
classificationTag, 16
columnChecks, 17
columnLevels, 18
comparison, 19
comparison, htestPhenStat-method
(htestPhenStat-class), 34
dataset, 19
dataset, PhenList-method
(PhenList-class), 49
determiningLambda, 19
dim, 20, 21
finalLRModel, 21
finalModel, 22
finalTFModel, 24
FisherExactTest, 25
formulaDeterminingLambda, 27
generateGraphs, 27
columnView
getColumn, 28
columnView, htestPhenStat-method
(htestPhenStat-class), 34
columnWeightBatchAdjusted, 29
columnWeightBatchAdjusted, PhenList, character-method
(PhenList-class), 49
columnView, 29
columnView, htestPhenStat-method
(htestPhenStat-class), 34
columnWeightBatchAdjusted, 29
columnWeightBatchAdjusted, PhenList, character-method
(PhenList-class), 49
columnView, 29
getColumn, PhenList, character-method
(PhenList-class), 49
getColumnBatchAdjusted, 29
columnView, htestPhenStat-method
(htestPhenStat-class), 34
columnWeightBatchAdjusted, 29
columnWeightBatchAdjusted, PhenList, character-method
(PhenList-class), 49
columnView, 29
getCountMatrices, 30
columnView, htestPhenStat-method
(htestPhenStat-class), 34
countMatrices, PhenTestResult-method
(PhenTestResult-class), 55
columnView, 29
countMatrices, PhenTestResult-method
(PhenTestResult-class), 55
countMatrices, PhenTestResult-method
(PhenTestResult-class), 55
countMatrices, PhenTestResult-method
(PhenTestResult-class), 55
countMatrices, PhenTestResult-method
(PhenTestResult-class), 55
countMatrices, PhenTestResult-method
(PhenTestResult-class), 55
parameters, 42
parameters, PhenTestResult-method (PhenTestResult-class), 55
parserOutputSummary, 42
parserOutputSummaryLR, 43
parserOutputTFSummary, 44
performReverseTransformation, 45
performTransformation, 46
PhenList, 7–10, 12–15, 20, 26, 37, 47, 48–50, 52, 59, 61, 65–75, 78–82, 86, 87, 91, 92
PhenList-class, 49
PhenStatReport, 51
PhenTestResult, 6, 7, 11, 13, 15–17, 21–26, 28, 35, 40, 41, 43–45, 53, 53, 54, 61, 62, 64–67, 69, 72, 74, 78, 80, 82–85, 87, 89, 90, 93–95
PhenTestResult-class, 55
plot.PhenList, 58, 87
plot.PhenTestResult, 59, 87
plotResidualPredicted, 61
printHead, LargeDataObject-method (LargeDataObject-class), 36
printLROutput, 62
printTabStyle, 63
pvalue, 64
pvalue, htestPhenStat-method (htestPhenStat-class), 34
qqplotGenotype, 64
qqplotRandomEffects, 65
qqplotRotatedResiduals, 66
recommendMethod, 67
refGenotype, 68
refGenotype, PhenList-method (PhenList-class), 49
refGenotype, PhenTestResult-method (PhenTestResult-class), 55
RRTest, 69
scatterplotGenotypeWeight, 70
scatterplotGenotypeWeightResult, 71
scatterplotSexGenotypeBatch, 73
scatterplotSexGenotypeBatchResult, 74
setBatch, 75
setBatch, PhenList, character-method (PhenList-class), 49
setGenotype, 76
setGenotype, PhenList, character-method (PhenList-class), 49
setMissingValue, 76
setMissingValue, PhenList, character-method (PhenList-class), 49
setSex, 76
setSex, PhenList, character-method (PhenList-class), 49
setWeight, 77
setWeight, PhenList, character-method (PhenList-class), 49
show, htestPhenStat-method (htestPhenStat-class), 34
show, LargeDataObject-method (LargeDataObject-class), 36
show, PhenTestResult-method (PhenTestResult-class), 55
startLRModel, 21, 77
startModel, 22, 79
startTFModel, 24, 81
subsetText, 83
subsetText, htestPhenStat-method (htestPhenStat-class), 34
summary, PhenTestResult, 83
summaryOutput, 84
testDataset, 6, 16, 21–25, 28, 35, 55, 61, 64–66, 70, 83, 84, 85, 93–95
testFinalLRModel, 88
testFinalModel, 90
testGenotype, 91
testGenotype, PhenList-method (PhenList-class), 49
testGenotype, PhenTestResult-method (PhenTestResult-class), 55
TFDataset, 91
transformation, 92
transformationJSON, 93
transformationText, 93
transformationText, PhenTestResult-method (PhenTestResult-class), 55
vectorOutput, 93
vectorOutputMatrices, 95
weightIn, 96
weightIn, PhenList-method (weightIn), 96
weightIn, PhenTestResult-method (weightIn), 96
weightIn-methods (weightIn), 96