Package ‘PhenStat’

December 22, 2016

Type Package
Title Statistical analysis of phenotypic data
Version 2.8.0
Date 2016-07-22
Author Natalja Kurbatova, Natasha Karp, Jeremy Mason
Maintainer Jeremy Mason <jmason@ebi.ac.uk>
Depends R (>= 2.3.0)
Imports methods, car, nlme, nortest, MASS, logistf
Suggests RUnit, BiocGenerics
Description Package contains methods for statistical analysis of phenotypic data.
License file LICENSE
NeedsCompilation no

R topics documented:

analysedDataset ..................................................... 3
analysedDatasetPhenList ........................................... 3
analysedSubset ....................................................... 3
analysisResults ..................................................... 4
batchIn ............................................................. 4
boxplotResidualBatch .............................................. 4
boxplotSexGenotype ............................................... 5
boxplotSexGenotypeBatch ......................................... 6
boxplotSexGenotypeBatchAdjusted ................................. 7
boxplotSexGenotypeResult ......................................... 8
boxplotSexGenotypeWeightBatchAdjusted ......................... 9
categoricalBarplot ................................................ 10
checkDataset ....................................................... 11
classificationTag ................................................... 12
comparison .......................................................... 14
dataset .............................................................. 14
finalLRModel ......................................................... 14
finalModel ........................................................... 15
finalTFModel ......................................................... 17
FisherExactTest ...................................................... 18
generateGraphs ...................................................... 19
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getColumn</code></td>
<td>20</td>
</tr>
<tr>
<td><code>getColumnBatchAdjusted</code></td>
<td>20</td>
</tr>
<tr>
<td><code>getColumnView</code></td>
<td>20</td>
</tr>
<tr>
<td><code>getColumnWeightBatchAdjusted</code></td>
<td>21</td>
</tr>
<tr>
<td><code>getCountMatrices</code></td>
<td>21</td>
</tr>
<tr>
<td><code>getDataset</code></td>
<td>21</td>
</tr>
<tr>
<td><code>getGenotypeEffect</code></td>
<td>21</td>
</tr>
<tr>
<td><code>getMatrix</code></td>
<td>22</td>
</tr>
<tr>
<td><code>getPercentageMatrix</code></td>
<td>22</td>
</tr>
<tr>
<td><code>getStat</code></td>
<td>22</td>
</tr>
<tr>
<td><code>getVariable</code></td>
<td>22</td>
</tr>
<tr>
<td><code>getVariables</code></td>
<td>23</td>
</tr>
<tr>
<td><code>hemiGenotype</code></td>
<td>23</td>
</tr>
<tr>
<td><code>JSONOutput</code></td>
<td>23</td>
</tr>
<tr>
<td><code>LRDataset</code></td>
<td>24</td>
</tr>
<tr>
<td><code>matrixCount</code></td>
<td>25</td>
</tr>
<tr>
<td><code>method</code></td>
<td>25</td>
</tr>
<tr>
<td><code>methodText</code></td>
<td>25</td>
</tr>
<tr>
<td><code>multipleBatches</code></td>
<td>26</td>
</tr>
<tr>
<td><code>noSexes</code></td>
<td>26</td>
</tr>
<tr>
<td><code>parameters</code></td>
<td>26</td>
</tr>
<tr>
<td><code>parserOutputSummary</code></td>
<td>27</td>
</tr>
<tr>
<td><code>parserOutputSummaryLR</code></td>
<td>28</td>
</tr>
<tr>
<td><code>parserOutputTFSummary</code></td>
<td>29</td>
</tr>
<tr>
<td><code>PhenList</code></td>
<td>30</td>
</tr>
<tr>
<td><code>PhenList-class</code></td>
<td>31</td>
</tr>
<tr>
<td><code>PhenTestResult</code></td>
<td>34</td>
</tr>
<tr>
<td><code>PhenTestResult-class</code></td>
<td>36</td>
</tr>
<tr>
<td><code>plotResidualPredicted</code></td>
<td>38</td>
</tr>
<tr>
<td><code>pvalue</code></td>
<td>39</td>
</tr>
<tr>
<td><code>qqplotGenotype</code></td>
<td>40</td>
</tr>
<tr>
<td><code>qqplotRandomEffects</code></td>
<td>41</td>
</tr>
<tr>
<td><code>qqplotRotatedResiduals</code></td>
<td>42</td>
</tr>
<tr>
<td><code>recommendMethod</code></td>
<td>43</td>
</tr>
<tr>
<td><code>refGenotype</code></td>
<td>44</td>
</tr>
<tr>
<td><code>RRTest</code></td>
<td>44</td>
</tr>
<tr>
<td><code>scatterplotGenotypeWeight</code></td>
<td>45</td>
</tr>
<tr>
<td><code>scatterplotGenotypeWeightResult</code></td>
<td>46</td>
</tr>
<tr>
<td><code>scatterplotSexGenotypeBatch</code></td>
<td>47</td>
</tr>
<tr>
<td><code>scatterplotSexGenotypeBatchResult</code></td>
<td>48</td>
</tr>
<tr>
<td><code>setBatch</code></td>
<td>49</td>
</tr>
<tr>
<td><code>setGenotype</code></td>
<td>50</td>
</tr>
<tr>
<td><code>setMissingValue</code></td>
<td>50</td>
</tr>
<tr>
<td><code>setSex</code></td>
<td>50</td>
</tr>
<tr>
<td><code>setWeight</code></td>
<td>50</td>
</tr>
<tr>
<td><code>startLRModel</code></td>
<td>51</td>
</tr>
<tr>
<td><code>startModel</code></td>
<td>52</td>
</tr>
<tr>
<td><code>startTFModel</code></td>
<td>54</td>
</tr>
<tr>
<td><code>subsetText</code></td>
<td>55</td>
</tr>
<tr>
<td><code>summaryOutput</code></td>
<td>56</td>
</tr>
<tr>
<td><code>testDataset</code></td>
<td>57</td>
</tr>
<tr>
<td><code>testFinalLRModel</code></td>
<td>59</td>
</tr>
</tbody>
</table>
analysedDataset

Method "analysedDataset"

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

Methods
signature(obj = "PhenTestResult")

analysedDatasetPhenList

Method "analysedDatasetPhenList"

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

Methods
signature(obj = "PhenList")

analysedSubset

Method "analysedSubset"

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

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

**Description**
Method `analysisResults` returns analysis results

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

---

**batchIn**  
*Method "batchIn"*

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

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

---

**boxplotResidualBatch**  
*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

**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 <- PhenList(dataset=read.csv(file),
testGenotype="Sparc/Sparc")
result <- testDataset(test,
    depVariable="Lean.Mass")
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

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


See Also

PhenList

Examples

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

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

---

**boxplotSexGenotypeBatch**

*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)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>phenList</code></td>
<td>instance of the <code>PhenList</code> class; mandatory argument</td>
</tr>
<tr>
<td><code>depVariable</code></td>
<td>a character string defining the dependent variable of interest; mandatory argument</td>
</tr>
<tr>
<td><code>graphingName</code></td>
<td>a character string defining the name to be used on the graph for the dependent variable; mandatory argument</td>
</tr>
<tr>
<td><code>outputMessages</code></td>
<td>flag: &quot;FALSE&quot; value to suppress output messages; &quot;TRUE&quot; value to show output messages; default value TRUE</td>
</tr>
</tbody>
</table>

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenList

Examples

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

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenList(dataset=read.csv(file),
                      testGenotype="Mysm1/+")
# box plot for females only dataset
boxplotSexGenotypeBatch(test_1sex,depVariable="Lean.Mass",
                         graphingName="Lean Mass")
```

`boxplotSexGenotypeBatchAdjusted`

*Method "boxplotSexGenotypeBatchAdjusted"*

Description

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

Usage

`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
Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
  testGenotype="Sparc/Sparc")
# box plot for dataset with two sexes: males and females
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

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenList

Examples

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

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

```
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

`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
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

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList

Examples

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

References


See Also

PhenList

Examples

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

description

Method "checkDataset"

Checks dataset for the minimum required info:
1. Column names should present
2. Genotype column should present
3. Sex column should 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)
```
classificationTag

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

Examples

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

classificationTag

Method "classificationTag"

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)
classificationTag

**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.

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason

**References**


**See Also**

- `PhenTestResult`

**Examples**

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
testGenotype="Sparc/Sparc")
result <- testDataset(test, depVariable="Lean.Mass")
classificationTag(result, userMode="summaryOutput", phenotypeThreshold=0.01)
classificationTag(result, userMode="vectorOutput")
```
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"

**Methods**

signature(obj = "hTestPhenStat")

dataset

Method "dataset"

**Description**

Method `dataset` returns data frame stored in the object

**Methods**

signature(obj = "PhenList")

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 <- PhenList(dataset=read.csv(file),
                 testGenotype="Mock")
testLR2 <- 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 <- testDataset(testLR2,
                        depVariable="V2",
                        callAll=FALSE,
                        method="LR")

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

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

resultLR <- finalLRModel(resultLR)
```

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.

Usage

```r
finalModel(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", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
    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 <- 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 <- testDataset(test,
#     depVariable="Lean.Mass",
#     equation="withWeight",
#     callAll=FALSE)
# print out new formula
#result$model.formula.genotype
# run the final model fitting when satisfied with the model
result <- finalModel(result)
```
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

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


See Also

PhenTestResult and testDataset

Examples

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

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
generateGraphs

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenTestResult

Examples

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

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

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenTestResult

getColumn

Method "getColumn"

Description

Method getColumn returns column of interest

Methods

signature(obj = "PhenList")

getColumnBatchAdjusted

Method "getColumnBatchAdjusted"

Description

Method getColumnBatchAdjusted returns column of interest adjusted for batch

Methods

signature(obj = "PhenList")

generatedView

Method "getColumnView"

Description

Method getColumnView 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.

**Methods**

signature(obj = "PhenList")

---

**getCountMatrices**  
*Method* "getCountMatrices"

**Description**

Method `getCountMatrices` 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.

**Methods**

signature(obj = "PhenList")

---

**getGenotypeEffect**  
*Method* "getGenotypeEffect"

**Description**

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

**Methods**

signature(obj = "PhenTestResult")
Method "getMatrix"

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

Methods
signature(obj = "htestPhenStat")

Method "getPercentageMatrix"

Description
Method getPercentageMatrix returns percentage matrix calculated out of counts.

Methods
signature(obj = "htestPhenStat")

Method "getStat"

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

Methods
signature(obj = "PhenList")

Method "getVariable"

Description
Method getVariable returns dependent variable name.

Methods
signature(obj = "PhenTestResult")
getVariables

Method "getVariables"

Description
Method getVariables returns names of variables in the dataset

Methods
signature(obj = "PhenList")

hemiGenotype

Method "hemiGenotype"

Description
Method hemiGenotype returns hemi genotype if defined

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

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
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

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

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

LRDataset

Method "LRDataset"

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

Usage
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 <- PhenList(dataset=read.csv(file),
testGenotype="Mock")
testLR2 <- LRDataset(testLR,"V2")
```

<table>
<thead>
<tr>
<th>matrixCount</th>
<th>Method &quot;matrixCount&quot;</th>
</tr>
</thead>
</table>

Description

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

Methods

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

<table>
<thead>
<tr>
<th>method</th>
<th>Method &quot;method&quot;</th>
</tr>
</thead>
</table>

Description

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

Methods

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

<table>
<thead>
<tr>
<th>methodText</th>
<th>Method &quot;methodText&quot;</th>
</tr>
</thead>
</table>

Description

Method `methodText` returns method’s name, for example: "Reference Range Plus", "Linear Mixed Model".

Methods

```r
signature(obj = "PhenTestResult")
```
**multipleBatches Method “multipleBatches”**

**Description**

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

**Methods**

signature(obj = "PhenList")

---

**noSexes Method “noSexes”**

**Description**

Method `noSexes` returns number of sex levels: 1/2

**Methods**

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

---

**parameters Method “parameters”**

**Description**

Method `parameters` returns parameters specific for applied method/dataset combination

**Methods**

signature(obj = "PhenTestResult")
parserOutputSummary  Method "parserOutputSummary"

Description

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

Usage

parserOutputSummary(linearRegressionOutput)

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 <- PhenList(dataset=read.csv(file),
  testGenotype="Sparc/Sparc")
result <- testDataset(test,
  depVariable="Lean.Mass")
linearRegressionOutput <- analysisResults(result)
parserOutputSummary(linearRegressionOutput)
```
Method "parserOutputSummaryLR"

Description

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

Usage

\[
\text{parserOutputSummaryLR(linearRegressionOutput)}
\]

Arguments

\[
\text{linearRegressionOutput}
\]

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

Value

Returns a named vector with logistic regression model output summary results

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenTestResult

Examples

\[
\begin{align*}
\text{file} & \leftarrow \text{system.file("extdata", "testLR.csv", package="PhenStat")} \\
\text{testLR} & \leftarrow \text{PhenList(dataset=read.csv(file),} \\
& \text{testGenotype="Mock")} \\
\text{testLR2} & \leftarrow \text{LRDataset(testLR,"V2")} \\
\text{resultLR} & \leftarrow \text{testDataset(testLR2,} \\
& \text{depVariable="V2",} \\
& \text{method="LR")} \\
\text{linearRegressionOutput} & \leftarrow \text{analysisResults(resultLR)} \\
\text{parserOutputSummaryLR(linearRegressionOutput)}
\end{align*}
\]
Method "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

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

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
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
  testGenotype="Sparc/Sparc")

test2 <- PhenList(dataset=read.csv(file),
  testGenotype="Arid4a/Arid4a",
  refGenotype="+/+",
  hemiGenotype="Arid4a/+",
  dataset.colname.weight="Weight.Value")

test3 <- PhenList(dataset=read.csv(file),
  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 <- PhenList(dataset=read.csv(file),
  testGenotype="Sparc/Sparc")
class(test)

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

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.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
model.effect.batch Result of the test for batch effect significance
model.effect.variance Result of the test for residual variance effect
model.effect.sex Result of the test for sex effect significance
model.effect.weight Result of the test for weight effect significance
model.effect.interaction Result of the test for genotype by sex interaction significance
model.output.interaction Interaction test result: p-value
numberSexes number of possible sex values in the dataset
pThreshold p-value threshold used to evaluate the effect significance, default value 0.05
PhenTestResult

model.formula.null
  formula for the model without genotype effect
model.formula.genotype
  formula for the model with genotype effect
model.output.genotype.nulltest.pVal
  Genotype test results: p-value
model.output.quality
  vector that contains diagnostic test output for mixed model quality of fit
model.output.summary
  named vector that contains summary of the model output
model.output.averageRefGenotype
  named vector that contains mean values calculated for reference genotype records
model.output.percentageChanges
  named vector that contains percentage changes values per sex

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 <- PhenList(dataset=read.csv(file),
  testGenotype="Sparc/Sparc")
# "testDataset" function calls "PhenTestResult" function internally
result <- testDataset(test,
  equation="withoutWeight",
  depVariable="Lean.Mass")
class(result)
```
PhenTestResult-class

Description

A list-based S4 class for storing model fitting results and other information like dependent variable, method used to build a model, etc. Helps 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:
   `analysedDataset(phenTestResult)`
2. "depVariable" contains dependent variable that was tested: `getVariable(phenTestResult)`
3. "refGenotype" contains reference genotype which is usually wildtype: `refGenotype(phenTestResult)`
4. "testGenotype" contains test genotype: `testGenotype(phenTestResult)`
5. "method" contains method name that was used for modelling/testing: `method(phenTestResult)`
6. "transformationRequired" contains TRUE/FALSE value indicating transformation of dependent variable:
   `transformationText(phenTestResult)` or `phenTestResult@transformationRequired`
7. "lambdaValue" contains lambda value for the Box-Cox transformation:
   `transformationText(phenTestResult)` or `phenTestResult$lambdaValue`
8. "scaleShift" contains the value of scale shift for Box-Cox transformation:
   `transformationText(phenTestResult)` or `phenTestResult@scaleShift`
9. "transformationCode" contains the code of transformation. Possible values:
   0 - transformation is not applicable (methods "FE", "RR", "LR") or switched off by user
   1 - transformation is not needed (1 is within the 95
   2 - log transformation
   3 - power transformation
   4 - transformation is not performed since optimal lambda value is not found (-5 > lambda > 5)
   `transformationText(phenTestResult)` or `phenTestResult@transformationCode`
10. "parameters" contains the parameters used during method application, e.g. pThreshold for MM and TF:
    `parameters(phenTestResult)`
11. Modelling/testing results are stored in the sections "analysisResults":
    `analysisResults(phenTestResult)`
    The contents of the analysisResults slot depend on framework that was used.
    The results of MM and TF frameworks:
    1. Equation used during linear modeling: `analysisResults(phenTestResult)$equation`
    2. Batch effect significance: `analysisResults(phenTestResult)$model.effect.batch`
    3. Variance equality: `analysisResults(phenTestResult)$model.effect.variance`
    4. Weight effect significance: `analysisResults(phenTestResult)$model.effect.weight`
    5. Sex effect significance: `analysisResults(phenTestResult)$model.output.interaction`
    6. Evidence of sex effect (p-value): `analysisResults(phenTestResult)$model.output.interaction`
PhenTestResult-class

7. Evidence of genotype effect (p-value): `analysisResults(phenTestResult)$model.output.genotype.nulltest.pVal`
8. Formula for the final genotype model: `analysisResults(phenTestResult)$model.formula.genotype`
9. Formula for the final null model: `analysisResults(phenTestResult)$model.formula.null`
10. Model fitting output: `analysisResults(phenTestResult)$model.output`
11. Model fitting summary: `summary(analysisResults(phenTestResult)$model.output)$tTable`

The results of FE and RR frameworks are stored as a list of htestPhenStat S4 objects: `analysisResults(phenTestResult)`
Each one htestPhenStat object contains:
1. Output of Fisher Exact Test: `pvalue(analysisResults(phenTestResult)[[1]])`.
2. Effect size: `ES(analysisResults(phenTestResult)[[1]])`.
3. Name of the table analysed (all, males, females): `analysedSubset(analysisResults(phenTestResult)[[1]])`.
4. Comparison, used for RR only (High vs Normal/Low, Low vs High/Normal): `comparison(analysisResults(phenTestResult)[[1]])`.
5. Matrix of counts: `matrixCount(analysisResults(phenTestResult)[[1]])`.

Slots

- `analysedDataset`: Object of class "data.frame" ~~ analysed dataset
- `transformationRequired`: Object of class "logical" ~~ flag indicating was or not transformation performed
- `lambdaValue`: Object of class "numeric" ~~ Box-Cox transform lambda value
- `scaleShift`: Object of class "numeric" ~~ Box-Cox transform scale shift
- `transformationCode`: Object of class "numeric" ~~ code explaining the transformation output
- `depVariable`: Object of class "character" ~~ dependent variable
- `refGenotype`: Object of class "character" ~~ reference genotype
- `testGenotype`: Object of class "character" ~~ test genotype
- `method`: Object of class "character" ~~ analysis method
- `parameters`: Object of class "matrix" ~~ parameters of method
- `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 <- PhenList(dataset=read.csv(file),
testGenotype="Sparc/Sparc")
result <- testDataset(test,
equation="withoutWeight",
depVariable="Bone.Area")
getVariable(result)
method(result)
# Batch effect is significant
analysisResults(result)$model.effect.batch
# Variance homogeneity
analysisResults(result)$model.effect.variance
# Weight effect is significant
analysisResults(result)$model.effect.weight
# Sex effect is significant
analysisResults(result)$model.effect.interaction
# Sex effect p-value - the result of the test
analysisResults(result)$model.output.interaction
# Genotype effect p-value
analysisResults(result)$model.output.genotype.nulltest.pVal
# Final model formula with genotype
analysisResults(result)$model.formula.genotype
# Final model formula without genotype
analysisResults(result)$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 <- PhenList(dataset=read.csv(file),
testGenotype="Aff3/Aff3")
result <- testDataset(test,
depVariable="Thoracic.Processes",
method="FE")
getVariable(result)
method(result)
for (i in seq_along(analysisResults(result))) {
val <- analysisResults(result)[[i]]
val
}

plotResidualPredicted  Method "plotResidualPredicted"
Method "pvalue"

Description

Method pvalue returns p-value that was calculate 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

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

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

PhenList and PhenTestResult

Examples

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

Method "qqplotRandomEffects"

**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

**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 <- PhenList(dataset = read.csv(file),
testGenotype="Sparc/Sparc")
result <- testDataset(test,
depVariable="Lean.Mass")
qqplotRandomEffects(result)
```
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

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

**References**


**See Also**
- `PhenList` and `PhenTestResult`

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

**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 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 <- PhenList(dataset=read.csv(file),
                 testGenotype="Sparc/Sparc")
recommendMethod(test, "Lean.Mass")
```
refGenotype

**Method "refGenotype"**

**Description**

Method refGenotype returns reference genotype

**Methods**

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


See Also

*PhenList* and *testDataset*

Examples

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

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

```
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

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason
References


See Also

PhenList

Examples

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
                  testGenotype="Sparc/Sparc")
scatterplotGenotypeWeight(test,
                         depVariable="Bone.Mineral.Content",
                         graphingName="BMC")

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenList(dataset=read.csv(file),
                      testGenotype="Mysml/+")
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)
```

**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

**Author(s)**

Natalja Kurbatova, Natasha Karp, Jeremy Mason
scatterplotSexGenotypeBatch

References


See Also

PhenList

Examples

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

file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenList(dataset=read.csv(file),
                      testGenotype="Mysm1/+")
result_1sex <- testDataset(test_1sex,
                     depVariable="Lean.Mass")
scatterplotGenotypeWeightResult(result_1sex,
                               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
Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

*PhenList*

Examples

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

```r
file <- system.file("extdata", "test4.csv", package="PhenStat")
test_1sex <- PhenList(dataset=read.csv(file),
  testGenotype="Mysm1/+")
# box plot for females only dataset
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

```r
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

Author(s)

Natalja Kurbatova, Natasha Karp, Jeremy Mason

References


See Also

- `PhenList`

Examples

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

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

**setBatch**

Method "setBatch"

Description

Method `setBatch` sets batch column in the dataset

Methods

signature(obj = "PhenList")
setGenotype

**Method** "setGenotype"

**Description**
Method `setGenotype` sets genotype column in the dataset

**Methods**

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

---

setMissingValue

**Method** "setMissingValue"

**Description**
Method `setMissingValue` sets missing value in the dataset

**Methods**

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

---

setSex

**Method** "setSex"

**Description**
Method `setSex` sets sex column in the dataset

**Methods**

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

---

setWeight

**Method** "setWeight"

**Description**
Method `setWeight` sets weight column in the dataset

**Methods**

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

```r
file <- system.file("extdata", "testLR.csv", package="PhenStat")
testLR <- PhenList(dataset=read.csv(file),
   testGenotype="Mock")
testLR2 <- 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 <- testDataset(testLR2,
   depVariable="V2",
   callAll=FALSE,
   method="LR")

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

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

### startModel

**Method "startModel"**

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)
```
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
code snippet
```

```r
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
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 <- testDataset(test,
depVariable="Lean.Mass",
callAll=FALSE)
# print out formula that has been created
analysisResults(result)$model.formula.genotype
# print out batch effect's significance
analysisResults(result)$model.effect.batch
# change the model
result <- testDataset(test,
depVariable="Lean.Mass",
equation="withWeight",
callAll=FALSE)
# print out new formula
analysisResults(result)$model.formula.genotype
```
Method "startTFModel"

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, 
equation="withWeight", outputMessages=TRUE, 
pThreshold=0.05, keepList=NULL)
```

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 <- PhenList(dataset=read.csv(file),
  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 <- 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 <- testDataset(test_TF, depVariable="Cholesterol", callAll=FALSE, method="TF")

# print out formula that has been created
analysisResults(result)$model.formula.genotype
# print out batch effect's significance
analysisResults(result)$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.

**Methods**

signature(obj = "htestPhenStat")
summaryOutput

Method "summaryOutput"

Description

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

Usage

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

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

# Fishe Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
test2 <- PhenList(dataset=read.csv(file),
```r
result2 <- testDataset(test2, 
  depVariable="Thoracic.Processes", 
  method="FE")
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 vari-
able.

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

```r
testDataset(phenList=NULL, depVariable=NULL,
  equation="withWeight", outputMessages=TRUE,
  pThreshold=0.05, method="MM", callAll=TRUE, keepList=NULL, dataPointsThreshold=4,
  RR_naturalVariation=95,RR_controlPointsThreshold=60,transformValues=FALSE,useUnfiltered=
```

### Arguments

- **phenList**: instance of the `PhenList` class; mandatory argument
- **depVariable**: a character string defining the dependent variable of interest; mandatory argu-
  ment
- **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" or "RR") defining the method to use for model
  building; default value "MM" for mixed model
- **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
testDataset

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

Value
Returns results stored in instance of the PhenTestResult class

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

References

See Also
PhenList

Examples

# Mixed Model framework
file <- system.file("extdata", "test1.csv", package="PhenStat")
test <- PhenList(dataset=read.csv(file),
testGenotype="Sparc/Sparc")
result <- testDataset(test,
depVariable="Lean.Mass")
# print out formula that has been created
analysisResults(result)$model.formula.genotype
summaryOutput(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 <- testDataset(test,
depVariable="Lean.Mass",
keepList=user_defined_effects)
# print out formula that has been created
analysisResults(result3)$model.formula.genotype
summaryOutput(result3)
testFinalLRModel

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

testFinalLRModel  Method "testFinalLRModel"

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

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

```r
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**

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

**Description**

Method `testGenotype` returns test genotype

**Methods**

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

---

**TFDataset**

**Method "TFDataset"**

**Description**

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

**Usage**

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

**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
- `forDecisionTree` flag: "FALSE" value for normal procedure; "TRUE" value to indicate that is called from decisionTree function; default value FALSE

**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 <- PhenList(dataset=read.csv(file),
                 testGenotype="Sparc/Sparc")
test2 <- TFDataset(test,"Lean.Mass")
```

---

transformation

**Method "transformation"**

**Description**

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

**Methods**

signature(obj = "PhenTestResult")

---

transformationJSON

**Method "transformationJSON"**

**Description**

Method transformationJSON returns transformation values as JSON string

**Methods**

signature(obj = "PhenTestResult")

---

transformationText

**Method "transformationText"**

**Description**

Method transformationText 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**

```r
vectorOutput(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: model output summary and other values

**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 <- PhenList(dataset=read.csv(file),
testGenotype="Sparc/Sparc")
result <- testDataset(test,
depVariable="Lean.Mass")
vectorOutput(result)

# Fishe Exact Test framework
file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
```
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 <- PhenList(dataset=read.csv(file),
testGenotype="Aff3/Aff3")
result <- testDataset(test,
  depVariable="Thoracic.Processes",
  method="FE")
vectorOutputMatrices(result)
```
**weightIn Method "weightIn"**

**Description**

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

**Methods**

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

*Topic other possible keyword(s) weightIn, 65

*Topic classes
PhenList-class, 31
PhenTestResult-class, 36

*Topic methods
analysedDataset, 3
analysedDatasetPhenList, 3
analysedSubset, 3
analysisResults, 4
batchIn, 4
comparison, 14
dataset, 14
getColumn, 20
getColumnBatchAdjusted, 20
getColumnView, 20
getColumnWeightBatchAdjusted, 21
getCountMatrices, 21
generateGraphs, 20

*Method weightIn

finalLRModel, 14
finalModel, 15
finalTFModel, 17
FisherExactTest, 18

generateGraphs, 19
getGenotype, 20

getColumn, PhenList, character-method
   (PhenList-class), 31
getColumnBatchAdjusted, 20
getColumnBatchAdjusted, PhenList, character-method
   (PhenList-class), 31
getColumnView, 20
getColumnBatchAdjusted, PhenList, character-method
   (PhenList-class), 31
getCountMatrices, 21
getCountMatrices, PhenTestResult-method
   (PhenTestResult-class), 36
getDataset, 21
getGenotypeEffect, 21
getGenotypeEffect, PhenTestResult-method
   (PhenTestResult-class), 36
getMatrix, 22
getPercentageMatrix, 22
getStat, 22
getStat, PhenList-method
   (PhenList-class), 31
getAddress, 22
getVariable, PhenTestResult-method
   (PhenTestResult-class), 36
getAddress, 23
getVariables, PhenList-method
   (PhenList-class), 31
hemiGenotype, 23
hemiGenotype, PhenList-method
   (PhenList-class), 31
JSONOutput, 23
LRDataset, 24
matrixCount, 25
method, 25
method, PhenTestResult-method
   (PhenTestResult-class), 36
methodText, 25
methodText, PhenTestResult-method
   (PhenTestResult-class), 36
multipleBatches, 26
multipleBatches, PhenList-method
   (PhenList-class), 31
noSexes, 26
noSexes, PhenList-method (noSexes), 26
noSexes, PhenTestResult-method
   (noSexes), 26
parameters, 26
parameters, PhenTestResult-method
   (PhenTestResult-class), 36
parserOutputSummary, 27
parserOutputSummaryLR, 28
parserOutputTFSummary, 29
PhenList, 5–12, 18, 24, 30, 30, 31–33, 39–49,
   51, 53–55, 57, 58, 61, 62
PhenTestResult-class, 31
PhenTestResult, 4, 5, 8, 10, 12–20, 23, 24,
   27–29, 34, 34, 35, 39–42, 44, 46, 49,
   51, 53, 54, 56, 58–60, 63, 64
PhenTestResult-class, 36
plotResidualPredicted, 36
pvalue, 39
qqplotGenotype, 40
qqplotRandomEffects, 41
qqplotRotatedResiduals, 42
recommendMethod, 43
refGenotype, 44
refGenotype, PhenList-method
   (PhenList-class), 31
refGenotype, PhenTestResult-method
   (PhenTestResult-class), 36
RRTest, 44
scatterplotGenotypeWeight, 45
scatterplotGenotypeWeightResult, 46
scatterplotSexGenotypeBatch, 47
scatterplotSexGenotypeBatchResult, 48
setBatch, 49
setBatch, PhenList, character-method
   (PhenList-class), 31
setGenotype, 50
setGenotype, PhenList, character-method
   (PhenList-class), 31
setMissingValue, 50
setMissingValue, PhenList, character-method
   (PhenList-class), 31
setSex, 50
setSex, PhenList, character-method
   (PhenList-class), 31
setWeight, 50
setWeight, PhenList, character-method
   (PhenList-class), 31
show, PhenTestResult-method
   (PhenTestResult-class), 36
startLRModel, 14, 51
startModel, 15, 52
startTFModel, 17, 54
subsetText, 55
summaryOutput, 56
testDataset, 4, 13–17, 19, 23, 36, 39–42, 45, 56, 57, 63, 64

testFinalLRModel, 59
testFinalModel, 60
testGenotype, 61
testGenotype,PhenList-method
   (PhenList-class), 31
testGenotype,PhenTestResult-method
   (PhenTestResult-class), 36
TFDataset, 61
transformation, 62
transformationJSON, 62
transformationText, 62
transformationText,PhenTestResult-method
   (PhenTestResult-class), 36

vectorOutput, 63
vectorOutputMatrices, 64

weightIn, 65
weightIn,PhenList-method (weightIn), 65
weightIn,PhenTestResult-method (weightIn), 65
weightIn-methods (weightIn), 65