PhenStat: statistical analysis of phenotypic data using Mixed Models and Fisher Exact Test

Natalja Kurbatova, Natasha Karp, Jeremy Mason

Modified: 20 September, 2013. Compiled: April 12, 2014

PhenStat is a package that provides statistical methods for the identification of abnormal phenotypes. The package contains dataset checks and cleaning in preparation for the analysis. For continuous data, an iterative fitting process is used to fit a regression model that is the most appropriate for the data, whilst for categorical data, a Fisher Exact Test is implemented.

Depending on the user needs, the output can either be interactive where the user can view the graphical output and analysis summary or for a database implementation the output consists of a vector of output and saved graphical files. PhenStat has been tested and demonstrated with an application of 420 lines of historic mouse phenotyping data.
The full PhenStat User's Guide with case studies and statistical analysis explanations is available as part of the online documentation, in "doc" section of the package and also through the github repository at `http://goo.gl/mKlX99`
Project github repository including dev version of the package: `http://goo.gl/2f9xqr`

Here we provide examples of functions usage. The package consists of three stages:

1. Dataset processing: includes checking, cleaning and terminology unification procedures and is completed by function *PhenList* which creates a *PhenList* object.

2. Statistical analysis: is managed by function *testDataset* and consists of Mixed Model or Fisher Exact framework implementations. The results are stored in *PhenTestResult* object.

3. Results Output: depending on user needs there are two functions for the test results output: *summaryOutput* and *vectorOutput* that present data from *PhenTestResult* object in a particular format.

1 Data Processing

PhenList function performs data processing and creates a *PhenList* object. As input, *PhenList* function requires dataset of phenotypic data that can be presented as data frame. For instance, it can be dataset stored in csv or txt file.

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test1.csv", package="PhenStat")
> dataset2 <- system.file("extdata", "test1.txt", package="PhenStat")
```

Data is organised with a row for a sample and each column provides information such as meta data (strain, genotype, etc.) and the variable of interest.

The main tasks performed by the PhenStat package's function *PhenList* are:

- terminology unification,

- filtering out undesirable records (when the argument *dataset.clean* is set to TRUE),

- and checking if the dataset can be used for the statistical analysis.

All tasks are accompanied by error messages, warnings and/or other information: error messages explain why function stopped, warning messages require user's attention (for instance, user is notified that column was renamed in the dataset), and information messages provide other details (for example, the values that are set in the Genotype column). If messages are not desirable *PhenList* function's argument *outputMessages* can be set to FALSE meaning there will be no essages.

Here is an example when the user sets out-messages to FALSE:

```
> # Default behaviour with messages
> library(PhenStat)
> dataset1 <- system.file("extdata", "test1.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(dataset1),
+         testGenotype="Sparc/Sparc")
> # Out-messages are switched off
> test <- PhenList(dataset=read.csv(dataset1),
+         testGenotype="Sparc/Sparc",
+         outputMessages=FALSE)
```

We define "terminology unification" as the terminology used to describe data (variables) that are essential for the analysis. The PhenStat package uses the following nomenclature for the names of columns: "Sex", "Genotype", "Batch" or "Assay.Date" and "Weight". In addition, expected sex values are "Male" and "Female" and missing value is *NA*.

In the example below dataset's values for females and males are 1 and 2 accordingly. Those values are changed to "Female" and "Male".

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test3.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(dataset1),
+         dataset.clean=TRUE,
+         dataset.values.female=1,
+         dataset.values.male=2,
+         testGenotype="Mysm1/+")
```

Filtering is required, as the statistical analysis requires there to be only two genotype groups for comparison (e.g. wild-type versus knockout). Thus the function *PhenList* requires users to define the reference genotype (mandatory argument *refGenotype* with default value "+/+") and test genotype (mandatory argument *testGenotype*). If the *PhenList* function argument *dataset.clean* is set to TRUE then all records with genotype values others than reference or test genotype are filtered out. The user may also specify hemizygotes genotype value (argument *hemiGenotype*) when hemizygotes are treated as the test genotype. This is necessary to manage sex linked genes, where the genotype will be described differently depending on the sex.

With *hemiGenotype* argument of the PhenList function defined as "KO/Y", the actions of the function are: "KO/Y" genotypes are relabelled to "KO/KO" for males; females "+/KO" heterozygous are filtered out.

If a user would like to switch off filtering, (s)he can set *PhenList* function's argument *dataset.clean* to FALSE (default value is TRUE). In the following example the same dataset is processed successfully passing the checks procedures when *dataset.clean* is set to TRUE and fails at checks otherwise.

1.1 PhenList Object

The output of the *PhenList* function is the *PhenList* object that contains a cleaned dataset (*PhenList* object's section *dataset*), simple statistics about dataset columns and additional information.

The example below shows how to print out the whole cleaned dataset and how to view the statistics about it. *PhenList* object has stored many characteristics about the data: reference genotype, test genotype, hemizygotes genotype, original column names, etc.

An example is given below.

```
> library(PhenStat)
> dataset2 <- system.file("extdata", "test2.csv", package="PhenStat")
> test2 <- PhenList(dataset=read.csv(dataset2),
+         testGenotype="Arid4a/Arid4a",
+         dataset.colname.weight="Weight.Value")
> test2$testGenotype
> test2$refGenotype
> test2$dataset.colname.weight
```

2 Data Analysis

The PhenStat package provides two methods (frameworks) for statistical analysis: Linear Mixed Models for continuous data and Fisher Exact Test for categorical data.

PhenStat's function *testDataset* works as a manager for the different statistical analyses methods. It checks the dependent variable, runs the selected statistical analysis framework and returns modelling/testing results in the *PhenTestResult* object.

The *testDataset* function's argument *phenList* defines the dataset stored in *PhenList* object.

Function's argument *depVariable* defines the dependent variable.

Function's argument *method* defines which statistical analysis framework to use. The default value is "MM" which stands for mixed model framework. To perform Fisher Exact Test, the argument *method* is set to "FE".

The *testDataset* function performs basic checks which ensure the statistical analysis would be appropriate and successful:

1. *depVariable* column is present in the dataset;
2. *depVariable* is numeric for Mixed Model (MM) framework, otherwise Fisher Exact Test (FE) is performed;
3. *depVariable* column values are variable enough (variability $> 0.5\%$) for MM framework, otherwise FE framework is recommended;
4. Each one genotype level has more than one *depVariable* level for MM framework, otherwise FE framework is recommended;
5. Number of *depVariable* levels is 10 or less for the FE framework.

If issues are identified, clear guidance is returned to the user. After the checking procedures, *testDataset* function runs the selected framework to analyse dependent variable.

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test1.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(dataset1),
+         testGenotype="Sparc/Sparc",
+         outputMessages=FALSE)
> # Default behaviour
> result <- testDataset(test,
+         depVariable="Bone.Area",
+         equation="withoutWeight")
> # Perform each step of the MM framework separatly
> result <- testDataset(test,
```

```
+         depVariable="Bone.Area",
+         equation="withoutWeight",callAll=FALSE)
> # Estimated model effects
> result$model.effect.batch
> result$model.effect.variance
> result$model.effect.weight
> result$model.effect.sex
> result$model.effect.interaction
> result$numberSexes
> # Change the effect values: interaction effect will stay in the model
> result <- testDataset(test,
+         depVariable="Bone.Area",
+         equation="withoutWeight",
+         keepList=c(TRUE,TRUE,FALSE,TRUE,TRUE),
+         callAll=FALSE)
> result <- finalModel(result)
> summaryOutput(result)
```

There are two functions we've implemented for the diagnostics and classification of MM framework results: *testFinalModel* and *classificationTag*.

```
> testFinalModel(result)
> classificationTag(result)
```

Example of Fisher Exact Test framework:

```
> library(PhenStat)
> dataset_cat <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test_cat <- PhenList(read.csv(dataset_cat),testGenotype="Aff3/Aff3")
> result_cat <- testDataset(test_cat,
+         depVariable="Thoracic.Processes",
+         method="FE")
> result_cat$depVariable
> result_cat$method
> result_cat$numberSexes
> # Chi squared table for all data
> result_cat$model.output$count_matrix_all
> # Chi squared table for males only records
> result_cat$model.output$count_matrix_male
> # Percentage matrix for all data
> result_cat$model.output$percentage_matrix_all
> # Percentage matrix for females only records
> result_cat$model.output$percentage_matrix_female
> # Matrix statistics for all data
> result_cat$model.output$stat_all
> # Matrix statistics for males only records
> result_cat$model.output$stat_male
> # Effect size for all data
> result_cat$model.output$ES
> # Effect size for females only records
> result_cat$model.output$ES_female
> # Fisher Exact Test results for all data
> result_cat$model.output$all
> # p-value for all data
> result_cat$model.output$all$p.value
```

3 Output of Results

The PhenStat package stores the results of statistical analyses in the *PhenTestResult* object. For numeric summary of the analysis, there are two functions to present *PhenTestResult* object data to the user: *summaryOutput* that provides a printed summary output and *vectorOutput* that creates a vector form output. These output forms were generated for differing users needs.

The *summaryOutput* function supports interactive analysis of the data and prints results on the screen.

The following is an example of summary output of MM framework:

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test1.csv", package="PhenStat")
> # MM framework
> test <- PhenList(dataset=read.csv(dataset1),
+         testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
+         depVariable="Lean.Mass",
+         outputMessages=FALSE)
> summaryOutput(result)
```

For the "FE" framework results *summaryOutput* function's output includes count matrices, statistics and effect size measures.

```
> library(PhenStat)
> dataset_cat <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test2 <- PhenList(dataset=read.csv(dataset_cat),
+         testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result2 <- testDataset(test2,
+         depVariable="Thoracic.Processes",
+         method="FE",outputMessages=FALSE)
> summaryOutput(result2)
```

vectorOutput function was developed for large scale application where automatic implementation would be required.

```
> library(PhenStat)
> dataset_cat <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test_cat <- PhenList(dataset=read.csv(dataset_cat),
+         testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result_cat <- testDataset(test_cat,
+         depVariable="Thoracic.Processes",
+         method="FE",outputMessages=FALSE)
> vectorOutput(result_cat)
```

There is an additional function to support the FE framework: *vectorOutputMatrices*. This function returns values from count matrices in the vector format.

```
> library(PhenStat)
> dataset_cat <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test_cat <- PhenList(dataset=read.csv(dataset_cat),
+         testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result_cat <- testDataset(test_cat,
+         depVariable="Thoracic.Processes",
+         method="FE",outputMessages=FALSE)
> vectorOutputMatrices(result_cat)
```

4 Graphics

For graphical output of the analysis, multiple graphical functions have been generated and these can be called by a user individually or alternatively, *generateGraphs* generates all relevant graphs for an analysis and stores the graphs in the defined directory.

There is only one graphical output for FE framework: categorical bar plots. This graph allows a visual representation of the count data, comparing observed proportions between reference and test genotypes.

```
> library(PhenStat)
> dataset_cat <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test_cat <- PhenList(dataset=read.csv(dataset_cat),
+         testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result_cat <- testDataset(test_cat,
+         depVariable="Thoracic.Processes",
+         method="FE",outputMessages=FALSE)
> categoricalBarplot(result_cat)
```

There are many graphic functions for the MM framework results. They can be divided into two types: dataset based graphs and results based graphs. There are three functions in the dataset based graphs category:

- *boxplotSexGenotype* creates a box plot split by sex and genotype.
- *boxplotSexGenotypeBatch* creates a box plot split by sex, genotype and batch if batch data present in the dataset. Please note the batches are not ordered with time but allow assessment of how the treatment groups lie relative to the normal control variation.
- *scatterplotGenotypeWeight* 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.

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test1.csv", package="PhenStat")
> # MM framework
> test <- PhenList(dataset=read.csv(dataset1),
+         testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
+         depVariable="Lean.Mass",
+         outputMessages=FALSE)
> boxplotSexGenotype(test,
+         depVariable="Lean.Mass",
+         graphingName="Lean Mass")
> boxplotSexGenotypeBatch(test,
+         depVariable="Lean.Mass",
+         graphingName="Lean Mass")
> scatterplotGenotypeWeight(test,
+         depVariable="Bone.Mineral.Content",
+         graphingName="BMC")
```

There are five functions in the results based graphs category:

- *qqplotGenotype* creates a Q-Q plot of residuals for each genotype.
- *qqplotRandomEffects* creates a Q-Q plot of blups (best linear unbiased predictions).
- *qqplotRotatedResiduals* creates a Q-Q plot of "rotated" residuals.

- *plotResidualPredicted* creates predicted versus residual values plots split by genotype.
- *boxplotResidualBatch* creates a box plot with residue versus batch split by genotype.

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test1.csv", package="PhenStat")
> # MM framework
> test <- PhenList(dataset=read.csv(dataset1),
+         testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
+         depVariable="Lean.Mass",
+         outputMessages=FALSE)
> qqplotGenotype(result)
> qqplotRandomEffects(result)
> qqplotRotatedResiduals(result)
> plotResidualPredicted(result)
> boxplotResidualBatch(result)
```