PhenStat: statistical analysis of phenotypic data

Natalja Kurbatova, Natasha Karp, Jeremy Mason, Hamed Haselimashhadi

Modified: 15 September, 2017 Compiled: April 7, 2020

PhenStat is a package that provides statistical methods as well as detailed reports 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. In addition, Reference Range Plus method has been implemented for a quick, simple analysis of the continuous data. It can be used in cases when regression model doesn't fit or isn't appropriate.

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/YKo54J`

The new features in the current versions of *PhenStat* can always been retrieved by the *PhenStat:::WhatIsNew()* command.

```
> PhenStat:::WhatIsNew()

        What is new in Version 2.23.0 :

        1. Due to the complexity of the method, Soft windowing function is REMOVED,
                see https://github.com/cran/SmoothWin for the new implementation of the method

        2. VectorOutput now lets user defined variables in the output

        3. Several improvements on VectorOutput including summary statistics and so on

        4. A new parameter "upper" is added to the testDataset(...) and other related functions t

        4. Bug fixed and minor improvements
```

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. Data analysis:

   Report generating: PhenStat is capable of producing detailed PDF reports from the input data by calling *PhenStatReport* on a *PhenList* object.

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 to test the results output: *summary* and *vectorOutput* that present data from *PhenTestResult* object in a particular format.

Contents

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 messages.

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, na.strings = '-'),
+                  testGenotype = "Sparc/Sparc")
> # Out-messages are switched off
> test <- PhenList(
+   dataset = read.csv(dataset1, na.strings = '-'),
+   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,na.strings = '-'),
+                  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 relabeled to "KO/KO" for males; females "+/KO" heterozygous are filtered out.

Filtering also takes place when there are records that do not have at least two records in the dataset with the same genotype and sex values. This type of filtering is needed to successfully process a dataset with Mixed Model, Time Fixed Effect and Logistic Regression frameworks. However, in some cases it is beneficial to process dataset with all genotype/sex records by using Fisher Exact Test and Reference Range Plus frameworks. Unfiltered dataset is stored within *PhenList* object to allow such processing.

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.

```
> library(PhenStat)
> dataset1 <- system.file("extdata", "test3.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(dataset1,na.strings = '-'),
```

```
+                   dataset.clean=TRUE,
+                   dataset.values.female=1,
+                   dataset.values.male=2,
+                   testGenotype="Mysm1/+")
> PhenStat:::getDataset(test)
> test
```

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,na.strings = '-'),
+                    testGenotype="Arid4a/Arid4a",
+                    dataset.colname.weight="Weight.Value")
> PhenStat:::testGenotype(test2)
> PhenStat:::refGenotype(test2)
```

1.2 PhenStat report

PhenStat is capable of producing a detailed report from the input data including several statistical methods. The report is regularly updated with the new methods and can be accessible on the IMPC website `https://goo.gl/kp44Ci`. The main function *PhenStatReport()* requires a PhenList object as an input and produces a PDF output.

```
> file <- system.file("extdata", "test1.csv", package = "PhenStat")
> test = PhenStat:::PhenList(dataset = read.csv(file, na.strings = '-'),
+                            testGenotype = "Sparc/Sparc")
> PhenStatReport(test,
+                depVariable = 'Bone.Area',
+                open = TRUE)
```

The function automatically searches for the new version on the IMPC website and downloads the newest one. If an internet connection is not provided, the default version of the report would be used.

2 Data Analysis

The package contains four statistical frameworks for the phenodeviants identification:

1. Mixed Models framework assumes that base line values of dependent variable are normally distributed but batch (assay date) adds noise and models variables accordingly in order to separate the batch and the genotype. Assume batch is normally distributed with defined variance. This framework can be used in case when you have controls measured over multiple batches and you ideally have knockout mice measured in multiple batches. The knockouts do not have to be concurrent with controls.

   Model weight: PhenStat lets assigning arbitrary weights to the observations in the Mixed model. This requires a vector of weights, $\sum w_i = 1.$ to be assigned to the parameter, *modelWeight*, in the main function *tesDataset*.

2. Time Fixed Effect framework estimates each batch effect to separate it from genotype. This framework can be used in case when there are up to 5 batches of the test genotype and concurrent controls approach had been used.

3. Reference Range Plus framework identifies the normal variation form the wild-type animals, classifies dependent variables from the genotype of interest as low, normal or high and compare proportions. This framework requires sufficient number of controls (more than 60 records) in order to correctly identify normal variation and can be used when other methods are not applicable or as a first simple data assessment method.

4. Fisher Exact Test is a standard framework for categorical data which compares data proportions and calculates the percentage change in classification.

All analysis frameworks output a statistical significance measure, an effect size measure, model diagnostics (when appropriate), and graphical visualisation of the genotype effect.

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.

The *testDataset* function's argument *depVariable* defines the dependent variable.

The *testDataset* function's argument *method* defines which statistical analysis framework to use. The default value is "MM" which stands for mixed model framework. To perform Time as Fixed Effect method the argument *method* is set to "TF". To perform Fisher Exact Test, the argument *method* is set to "FE". For the Reference Range Plus framework *method* is set to "RR".

There are two arguments specific for the "MM" and "TF" frameworks:

- *dataPointsThreshold* defines the required number of data points in a group (subsets per genotype and sex combinations) for a successful analysis. The default value is 4. The minimal value is 2.

- *transformValues* defines to perform or not data transformation if needed. The default value is FALSE.

There is an argument *useUnfiltered* specific for "RR" and "FE" frameworks which defines to use or not unfiltered dataset (dataset with all records regardless the number of records per genotype/sex combinations). The default value is FALSE.

There are two more arguments specific for the "RR" framework:

- *RR_naturalVariation* for the variation ranges in the RR framework with default value set to 95 and minimal value set to 60;

- *RR_controlPointsThreshold* for the number of control data points in the RR framework with default value 60 and minimal value set to 40.

The *testDataset* function performs basic checks which ensure the statistical analysis would be appropriate and successful: *depVariable* column is present in the dataset; thresholds value are set and do not exceed minimal values.

After the basic checks the *testDataset* function performs framework specific checks:

- Mixed Model (MM) and Time as Fixed Effect (TF) framework checks:

    1. *depVariable* column values are numeric.
    2. Variability check 1 (whole column): *depVariable* column values are variable enough (the ratio of different values to all values in the column $\geq 0.5\%$);

3. Variability check 2 (variability within a group): there are enough data points in subsets per genotype/sex combinations. The number of values from *dep Variable* column should exceed *dataPoints Threshold* in all subsets.
4. Variability check 3 (variability for "Weight" column) applied only when *equation* argument value is set to "withWeight": there are enough weight records in subsets per genotype/sex combinations. The number of values from "Weight" column should exceed *dataPoints Threshold* in all subsets, otherwise *equation* "withoutWeight" is used;

- Additional Time as Fixed Effect (TF) framework's checks:
  1. Number of batches: there are from 2 to 5 batches (assay dates) in the dataset.
  2. Control points: there are concurrent controls data in the dataset, meaning the presence of data points for at least one sex in all genotype/batch level combinations.
- Reference Range Plus (RR) framework's checks:
  1. *dep Variable* column values are numeric.
  2. There are data: the number of levels in *dep Variable* column after filtering out of null values exceeds zero.
  3. Control points: there are enough data points in subsets per reference genotype/sex combinations. The number of values from *dep Variable* column should exceed *RR_control Points Threshold* in all subsets.
- Fisher Exact Test (FE) framework's checks:
  1. There are data: the number of levels in *dep Variable* column after filtering out of null values exceeds zero.
  2. Number of levels: number of *dep Variable* levels is less than 10.

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,na.strings = '-'),
+                  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
> linearRegressionResults <- PhenStat:::analysisResults(result)
> linearRegressionResults$model.effect.batch
> linearRegressionResults$model.effect.variance
> linearRegressionResults$model.effect.weight
> linearRegressionResults$model.effect.sex
> linearRegressionResults$model.effect.interaction
> # 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 <- PhenStat:::finalModel(result)
> summary(result)
```

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

```
> PhenStat:::testFinalModel(result)
> PhenStat:::classificationTag(result)
```

Example of Time Fixed Effect framework:

```
> file <- system.file("extdata", "test7_TFE.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(file,na.strings = '-'),
+                  testGenotype="het",
+                  refGenotype = "WT",
+                  dataset.colname.sex="sex",
+                  dataset.colname.genotype="Genotype",
+                  dataset.values.female="f",
+                  dataset.values.male= "m",
+                  dataset.colname.weight="body.weight",
+                  dataset.colname.batch="Date_of_procedure_start")
> # TFDataset function creates cleaned dataset - concurrent controls dataset
> test_TF <- PhenStat:::TFDataset(test,depVariable="Cholesterol")
> # TF method is called
> result  <- testDataset(test_TF,
+                        depVariable="Cholesterol",
+                        method="TF")
> summary(result)
```

Example of Reference Range Plus framework:

```
> library(PhenStat)
> file <- system.file("extdata", "test1.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(file,na.strings = '-'),
+                  testGenotype="Sparc/Sparc")
> # RR method is called
> result <- testDataset(test,
+                       depVariable="Lean.Mass",
+                       method="RR")
> summary(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,na.strings = '-'),testGenotype="Aff3/Aff3")
> result_cat <- testDataset(test_cat,
+                           depVariable="Thoracic.Processes",
+                           method="FE")
> PhenStat:::getVariable(result_cat)
> PhenStat:::method(result_cat)
> summary(result_cat)
```

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: *summary* that provides a printed summary output and *vectorOutput* that creates a vector form output. These output forms were generated for differing users needs.

The *summary* 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,na.strings = '-'),
+                  testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
+                       depVariable="Lean.Mass",
+                       outputMessages=FALSE)
> summary(result)
```

For the "FE" framework results *summary* 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,na.strings = '-'),
+                   testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result2 <- testDataset(test2,
+                        depVariable="Thoracic.Processes",
+                        method="FE",outputMessages=FALSE)
> summary(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,na.strings = '-'),
+                      testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result_cat <- testDataset(test_cat,
+                           depVariable="Thoracic.Processes",
+                           method="FE",outputMessages=FALSE)
> PhenStat:::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,na.strings = '-'),
+                      testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result_cat <- testDataset(test_cat,
+                           depVariable="Thoracic.Processes",
+                           method="FE",outputMessages=FALSE)
>
> #vectorOutputMatrices(result_cat)
```

4 Graphics

Graphics in the PhenStat are as easy as calling the **plot()** function on a PhenList or the testDataset (or called PhenTestResult) object.

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

All plots in the *plot()* function can be produced individually either by directly calling them or passing the names as a parameter to the plot() function. To store the graphics, one can use the *generateGraphs* function. We explain each individual graphic in the following.

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,na.strings = '-'),
+                      testGenotype="Aff3/Aff3",outputMessages=FALSE)
> result_cat <- testDataset(test_cat,
+                           depVariable="Thoracic.Processes",
+                           method="FE",outputMessages=FALSE)
> plot(result_cat)
```

There are many graphic functions for the regression frameworks' results. Though some are specific to MM. Those graphic functions can be divided into two types: dataset based graphs and results based graphs. There are three functions in the dataset based graphs category:

- *boxplotSexGenotype* or similarly *plot(., type='boxplotSexGenotype')* create a box plot split by sex and genotype.
- *scatterplotSexGenotypeBatch* or similarly *plot(., type='scatterplotSexGenotypeBatch')* create a scatter 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* or similarly *plot(., type='scatterplotGenotypeWeight')* create 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,na.strings = '-'),
+                  testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
+                       depVariable="Lean.Mass",
+                       outputMessages=FALSE)
> PhenStat:::boxplotSexGenotype(test,
```

```
+                                depVariable="Lean.Mass",
+                                graphingName="Lean Mass")
> PhenStat:::scatterplotSexGenotypeBatch(test,
+                                        depVariable="Lean.Mass",
+                                        graphingName="Lean Mass")
> PhenStat:::scatterplotGenotypeWeight(test,
+                                      depVariable="Bone.Mineral.Content",
+                                      graphingName="BMC")
>
> # All in one
> #plot(
> #  test,
> #  depVariable = 'Lean.Mass',
> #  type = c(
> #  'boxplotSexGenotype',
> #  'scatterplotSexGenotypeBatch',
> #  'scatterplotGenotypeWeight'
> #  )
> #  )
```

There are five functions in the results based graphs category:

- *qqplotGenotype* or similarly *plot(., type='qqplotGenotype')* create a Q-Q plot of residuals for each genotype.
- *qqplotRandomEffects* or similarly *plot(., type='qqplotGenotype')* create a Q-Q plot of blups (best linear unbiased predictions). MM specific.
- *qqplotRotatedResiduals* or similarly *plot(., type='qqplotRotatedResiduals')* create a Q-Q plot of "rotated" residuals. MM specific.
- *plotResidualPredicted* or similarly *plot(., type='plotResidualPredicted')* create predicted versus residual values plots split by genotype.
- *boxplotResidualBatch* or similarly *plot(., type='boxplotResidualBatch')* create 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,na.strings = '-'),
+                  testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
+                       depVariable="Lean.Mass",
+                       outputMessages=FALSE)
> # All plots together
> # plot(result)
>
> PhenStat:::qqplotGenotype(result)
> PhenStat:::qqplotRandomEffects(result)
> PhenStat:::qqplotRotatedResiduals(result)
> PhenStat:::plotResidualPredicted(result)
> PhenStat:::boxplotResidualBatch(result)
```