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

Natalja Kurbatova, Natasha Karp, Jeremy Mason

Last revised: 3rd October 2013

# Contents

# 1 Introduction

High-throughput phenotyping generates large volumes of varied data including both categorical and continuous data. Operational and cost constraints can lead to a work-flow that precludes the traditional analysis methods. Furthermore, for a high throughput environment, a robust automated statistical pipeline that alleviates manual intervention is required. 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 (Mixed Model framework), whilst for categorical data; a Fisher Exact Test is implemented (Fisher Exact Test framework). The Mixed Model (MM) framework is an iterative process to select the best model for the data which considers both the best modelling approach (mixed model or general linear regression) and which factors to include in the model. There is also user control functionality on whether to include body weight in the modelling process. The MM output includes model fit assessments (graphical and testing output). Both analysis frameworks output a statistical significance measure, an effect size measure, model diagnostics (when appropriate), and graphical visualisation of the genotype effect.

The statistical analysis generates measures of statistical significance, effect size estimates, model diagnostics where applicable and visualisation of data. 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.

This package has been tested and demonstrated with an application of XX lines of historic mouse phenotyping data.

The package consists of three stages as shown in Figure **??**:

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 a Fisher Exact framework implementations. The results are stored in *PhenTestResult* object. Potentially this layer can be extended adding new statical methods (shown as dotted box in

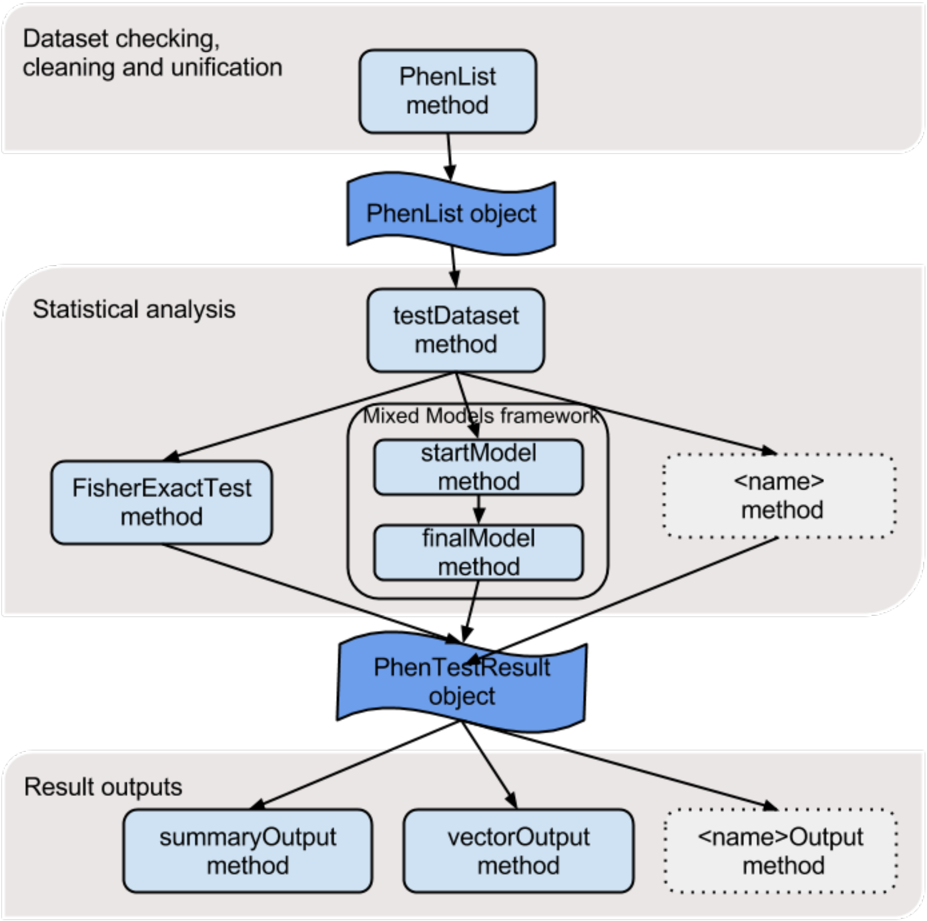


Figure 1: The PhenStat package's three stage structure: dataset processing, analysis and result output. Dotted boxes show the place-holders for new functions that could implement other methods for data analysis and/or output of results.

Figure **??**).

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. The output layer is also easily extendible (shown as dotted box in Figure **??**).

# 2 Data Processing with PhenList Function

*PhenList* function performs data processing and creates *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. We expect column names to represent variables.

```
> dataset <- read.csv("myPhenotypicDataset.csv")
> dataset <- read.table("myPhenotypicDataset.txt",sep="\t")
```

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

- terminology unification,
- filtering out undesirable records (only when function's argument *dataset.clean* is set to TRUE),
- and checking if the dataset can be used for the statistical analysis.

All tasks are accompanied by messages with errors, warnings and/or 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), information messages provide some details (for instance, Genotype levels). If the observed is 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 is setting out-messages to FALSE:

```
> dataset1 <- read.csv("./PhenStat/extdata/test.csv")

# default behaviour with messages
> test <- PhenList(dataset=dataset1,
  testGenotype="Sparc/Sparc")

Warning:
Dataset's column 'Assay.Date' has been renamed to 'Batch' and will be used for the batch effect modelling.

Information:
```

```
Dataset’s ’Genotype’ column has following values: ’+/+’, ’Sparc/Sparc’

Information:
Dataset’s ’Gender’ column has following value(s): ’Female’, ’Male’

# Out-messages are switched off
> test <- PhenList(dataset=dataset1,
  testGenotype="Sparc/Sparc",
  outputMessages=FALSE)

# There are no messages!
```

## 2.1 Terminology Unification

Under the term ”terminology unification” we mean the matching of the terminology for the data (variables) that are essential for the analysis. The PhenStat package uses the following nomenclature for the names of columns: ”Gender”, ”Genotype”, ”Batch” or ”Assay.Date”, ”Weight”. In addition expected gender values are ”Male” and ”Female”, missing value – NA. *PhenList* function firstly creates a copy of the dataset and then uses special arguments that help to map columns and values from user’s naming system into the package’s nomenclature. The original file with the dataset stays unchanged since all changes take place within *PhenList* object. ”Assay.Date” is renamed to ”Batch” automatically.

The following *PhenList* function’s arguments have to be specified when the other names of columns or gender values are used within the user’s dataset:

- *dataset.colname.batch* allows to define column name within dataset for the batch effect if this column name is other than ”Batch” or ”Assay.Date”,
- *dataset.colname.genotype* allows to define column name within dataset for the genotype info if this column name is other than ”Genotype”,
- *dataset.colname.gender* allows to define column name within dataset for the gender info if this column name is other than ”Gender” in the dataset,
- *dataset.colname.weight* allows to specify column name within dataset for the weight info if this column name is other than ”Weight” in the dataset,
- *dataset.values.missingValue* allows to specify value used as missing

value in the dataset if other than NA,

- *dataset.values.male* allows to define value used to label ”males” in the dataset if other than ”Male”,
- *dataset.values.female* allows to specify value used to label ”females” in the dataset if other than ”Female” value has been used.

In the example above dataset’s values for females and males are 1 and 2 accordingly. Those values are changed to ”Female” and ”Male”.

```
> dataset_test <- read.csv("./PhenStat/extdata/test3.csv")

> test <- PhenList(dataset=dataset_test,
  dataset.clean=TRUE,
  dataset.values.female=1,
  dataset.values.male=2,
  testGenotype="Mysm1/+")

Warning:
Dataset's column 'Assay.Date' has been renamed to 'Batch' and will be used for the batch effect modelling.

Information:
Dataset's 'Genotype' column has following values: '+/+', 'Mysm1/+'

Information:
Dataset's 'Gender' column has following value(s): 'Female', 'Male'
```

## 2.2 Filtering

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 (default value is *refGenotype* = ”+/+”) and test genotype (*testGenotype*). If *PhenList* function is in the cleaning mode then all records with genotype values others than reference or test genotype are filtered out. Another option for a user is to specify hemizygotes genotype value (*hemiGenotype*) then hemizygotes are treated as test genotype. This is necessary to manage sex linked genes, where the genotype will be described differently depending on the sex but biologically then should be equivalent. Consider the following example of the genotype values in the dataset:

With the dataset described in Table **??** when *hemiGenotype* argument of the PhenList function is defined as ”KO/Y” the actions of the function are: ”KO/Y” genotypes are relabelled to ”KO/KO” for males; females ”+/KO”

Table 1: Example of the dataset with sex linked genes

| Sex | Reference genotype | Test genotype | Heterozygous genotype |
|---|---|---|---|
| Female | +/+ | KO/KO | +/KO |
| Male | +/+ | KO/Y | |

heterozygous are filtered out.

```
> dataset1 <- read.csv("sex_linked_genes.csv")
> test <- PhenList(dataset=dataset1,
  testGenotype="KO/KO",
  refGenotype="+/+",
  hemiGenotype="KO/Y")

Warning:
Dataset's column 'Assay.Date' has been renamed to 'Batch' and will be used for the batch effect modelling.

Warning:
Hemizygotes 'KO/Y' have been relabeled to test genotype 'KO/KO'.
If you don't want this behaviour then don't define 'hemiGenotype' argument.

Information:
Dataset's 'Genotype' column has following values: '+/+', 'KO/KO'

Information:
Dataset's 'Gender' column has following value(s): 'Female', 'Male'
```

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

```
> dataset <- read.csv("test_3genotypes.csv")
> test<-PhenList(dataset,
testGenotype="Mysm1/+")

Warning:
Dataset's column 'Assay.Date' has been renamed to 'Batch' and will be used for the batch effect modelling.

Warning:
Dataset has been cleaned by filtering out records with genotype value
other than test genotype 'Mysm1/+' or reference genotype '+/+'.

Information:
Dataset's 'Genotype' column has following values: '+/+', 'Mysm1/+'

Information:
```

```
Dataset's 'Gender' column has following value(s): 'Female', 'Male'

# Filtering is swtiched off
> test<-PhenList(dataset,
testGenotype="Mysm1/+",
dataset.clean=FALSE)

Warning:
Dataset's 'Batch' column is missed.
You can define 'dataset.colname.batch' argument to specify column
for the batch effect modelling. Otherwise you can only fit a glm.

Information:
Dataset's 'Genotype' column has following values: '+/+', 'HOM', 'Mysm1/+'

Information:
Dataset's 'Gender' column has following value(s): 'Female', 'Male'

********* Errors start *********

Check failed:
Dataset's 'Genotype' column has to have two values.
You can define 'testGenotype' and 'refGenotype' arguments to automatically
filter out records with genotype values other than specified.
Alternatively you can define 'hemiGenotype' and 'testGenotype' arguments to relabel hemizygotes to homozygotes.

********* Errors end ***********
```

Filtering take place also when there are records that do not have at least one more another record in the dataset with the same genotype and gender values.

Consider the following example of the genotype and gender values in the dataset:

Table 2: Example of the dataset with 3 gender values

| Sex | Reference genotype | Test genotype |
|---|---|---|
| Female | +/+ | Mysm1/+ |
| Male | +/+ | Mysm1/+ |
| unsexed | | Mysm1/+ (1 record only) |

When *dataset.clean* argument's is set to TRUE all "unsexed" records are filtered out since there are no records for genotype "+/+" and only one record for "Mysm1/+".

## 2.3 Dataset Checks

After terminology unification and filtering tasks, *PhenList* function checks the dataset availability for the statistical analysis:

- column names and gender values are there and described in the package's nomenclature,
- test and reference genotype records are in the dataset,
- there are at least two records for each genotype/gender values combination.

If one of the checks fails function stops and the PhenList object is not created. In the following example "Gender" column is missed in the dataset and checks fail.

```
> dataset <- read.csv("test_noGenderColumn.csv")
> test<-PhenList(dataset,testGenotype="Mysm1/+")
Warning:
Dataset's column 'Assay.Date' has been renamed to 'Batch'
and will be used for the batch effect modelling.

********* Errors start *********

Check failed:
Dataset's 'Gender' column is missed.

********* Errors end ***********
```

Next example is showing the dataset described in the previous section **??** : three gender values and not enough records for the "unsexed" gender and both genotype values.

```
> dataset <- read.csv("test_3genders.csv")
> test<-PhenList(dataset,
testGenotype="Mysm1/+")
...
Warning:
Since dataset has to have at least two data points for each genotype/gender combination
and there are not enough records for the combination(s): '+/+'/'unsexed' (0),
 'Mysm1/+'/'unsexed' (1), appropriate gender records have been filtered out from the dataset.

...

# Filtering is switched off
> test<-PhenList(dataset,
testGenotype="Mysm1/+",
dataset.clean=FALSE)
```

```
...

********* Errors start *********

Check failed:
Dataset's 'Gender' column has to have one or two values and currently the data has more than two.

Check failed:
Dataset's 'Gender' column has 'Female', 'Male', 'unsexed' values
instead of 'Female' and/or 'Male' values only.
Please delete records with gender(s) 'unsexed' from the dataset.

Check failed:
Dataset should have at least two data points for each genotype/gender combination.
At the moment there are no enough data points for the following combination(s):
'+/+'/'unsexed' (0), 'Mysm1/+'/'unsexed' (1).

********* Errors end ***********
```

We believe that a lot of checking failures are avoided when *dataset.clean* argument of the *PhenList* funciton is set to TRUE (default value). Few examples are given in this and in the previous section **??**.

## 2.4 PhenList Object

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

The example below is showing how to print out the whole cleaned dataset and how to view the statistics about it (output is shown in Table **??**).

```
> dataset1 <- read.csv("./PhenStat/extdata/test.csv")

> test <- PhenList(dataset=dataset1,
  testGenotype="Sparc/Sparc", outputMessages=FALSE)

> test$dataset
...
> test$dataset.stat
...
```

Table **??** shows the content of the *dataset.stat* section and describes the data focusing on the columns of the dataset. Each column will now be a variable with summary description. The description includes: whether variable is numerical or not, whether variable is continuous or not (variability is more than 5%), number of levels, number of data points, for the numerical variables: mean, standard deviation, minimal and maximal values.

Table 3: Simple statistics about dataset variables – *dataset.stat* content

| Variable | Num | Cont | Levels | # | Mean | StdDev | Min | Max |
|---|---|---|---|---|---|---|---|---|
| Age.In.Weeks | TRUE | FALSE | 10 | 468 | 14 | 0.21 | 13.1 | 14.6 |
| Batch | FALSE | FALSE | 49 | 468 | NA | NA | NA | NA |
| Birth.Date | FALSE | FALSE | 111 | 468 | NA | NA | NA | NA |
| Bone.Area | TRUE | TRUE | 248 | 463 | 9.6 | 0.84 | 7.46 | 11.73 |
| Bone.Mineral.Content | TRUE | TRUE | 405 | 463 | 0.48 | 0.06 | 0.31 | 0.64 |
| Bone.Mineral.Density | TRUE | TRUE | 120 | 463 | 0.05 | 0 | 0.04 | 0.06 |
| Cohort.Name | FALSE | FALSE | 59 | 468 | NA | NA | NA | NA |
| Colony.Name | FALSE | FALSE | 76 | 468 | NA | NA | NA | NA |
| Colony.Prefix | FALSE | FALSE | 76 | 468 | NA | NA | NA | NA |
| Core.Strain | FALSE | FALSE | 1 | 468 | NA | NA | NA | NA |
| Tissue.Mass | TRUE | TRUE | 427 | 463 | 35.22 | 5.3 | 20.44 | 49.86 |
| Fat.Mass | TRUE | TRUE | 385 | 463 | 14.92 | 3.35 | 4.52 | 23.21 |
| Fat.Percentage | TRUE | TRUE | 403 | 463 | 42.01 | 5.16 | 19.26 | 55.21 |
| Full.Strain | FALSE | FALSE | 9 | 468 | NA | NA | NA | NA |
| Gender | FALSE | FALSE | 2 | 468 | NA | NA | NA | NA |
| Gene.Name | FALSE | FALSE | 76 | 468 | NA | NA | NA | NA |
| Genotype | FALSE | FALSE | 2 | 468 | NA | NA | NA | NA |
| Lean.Mass | TRUE | TRUE | 369 | 463 | 20.31 | 2.81 | 14.84 | 28.8 |
| Mouse | FALSE | FALSE | 468 | 468 | NA | NA | NA | NA |
| Mouse.Name | FALSE | FALSE | 468 | 468 | NA | NA | NA | NA |
| Base.Length | TRUE | FALSE | 17 | 468 | 10.19 | 0.32 | 9.3 | 10.9 |
| Pipeline | FALSE | FALSE | 1 | 468 | NA | NA | NA | NA |
| Strain | FALSE | FALSE | 2 | 468 | NA | NA | NA | NA |
| Weight | TRUE | TRUE | 183 | 468 | 34.95 | 5.09 | 20.4 | 48.4 |

*PhenList* object has stored many characteristics of data: reference genotype, test genotype, hemizygotes genotype, original column names, etc. Note: reference and test genotypes are always defined. Other information can be missed if was not provided at the moment of *PhenList* object creation.

Example of some of characteristics is given below.

```
> dataset2 <- read.csv("./PhenStat/extdata/test2.csv")
> test2 <- PhenList(dataset=dataset2,
testGenotype="Arid4a/Arid4a",
dataset.colname.weight="Weight.Value")
```

```
> test2$testGenotype

[1] "Arid4a/Arid4a"

> test2$refGenotype

[1] "+/+"

> test2$dataset.colname.weight

[1] "Weight.Value"
```

# 3 Statistical Analysis

The PhenStat package provides two methods (frameworks) for the statistical analysis: Linear Mixed Models and Fisher Exact Test for categorical data. For both the MM and FE framework, the statistical significance is assessed, the biological significance through an effect size estimate and finally the genotype effect is classified e.g. "If phenotype is significant - both sexes equally".

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

## 3.1 Manager for Analysis Methods – *testDataset* function

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

Function's argument *depVariable* defines 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 should present in the dataset;

2. *depVariable* should be numeric for Mixed Model (MM) framework, otherwise performs Fisher Exact Test (FE);
3. *depVariable* column values are variable enough (variability ¿ 5%) for MM framework, otherwise recommends FE framework;
4. Each one genotype level should have more than one *depVariable* level (variability) for MM framework, otherwise recommends FE framework;
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.

To ensure flexibility and debugging potential framework can comprise of more than 1 stage. For instance, MM framework have two stages.

*testDataset* function's argument *callAll* allows to run all stages of the framework one after another when set to TRUE (default behaviour). However, when *callAll* flag is set to FALSE it indicates *testDataset* function to run only the first stage of the selected framework. For instance, *testDataset* function runs *startModel* and after that *finalModel* functions of the MM framework if the argument *callAll* is set to TRUE. If framework contains only one stage like in Fisher Exact Test case then *testDataset* function runs that one stage regardless the *callAll* argument's value. See example of *callAll* argument in the section **??**. The example how to call MM and FE framework is given below.

```
> dataset1 <- read.csv("./PhenStat/extdata/test.csv")

> test <- PhenList(dataset=dataset1,
  testGenotype="Sparc/Sparc", outputMessages=FALSE)

> result_MM_Lean.Mass <- testDataset(test,depVariable="Lean.Mass", method="MM")
...
> result_FE_Length <- testDataset(test,depVariable="Nose.To.Tail.Base.Length", method="FE")
..
```

The details about MM and FE framework are in the next two subsections.

## 3.2 Mixed Model Framework

First, we will describe the mixed models top-down methodology which starts with a fully loaded model and ends with final reduced model and genotype effect evaluation procedures.

### 3.2.1 Theory

There are two possible start models, depending on whether weight is included as a factor (see **??**. for the model including weight and **??**. for the model without weight).

$$depVariable \backsim Genotype + Gender + Genotype * Gender \qquad \text{(Eq1)}$$

$$depVariable \backsim Genotype + Gender + Genotype * Gender + Weight \qquad \text{(Eq2)}$$

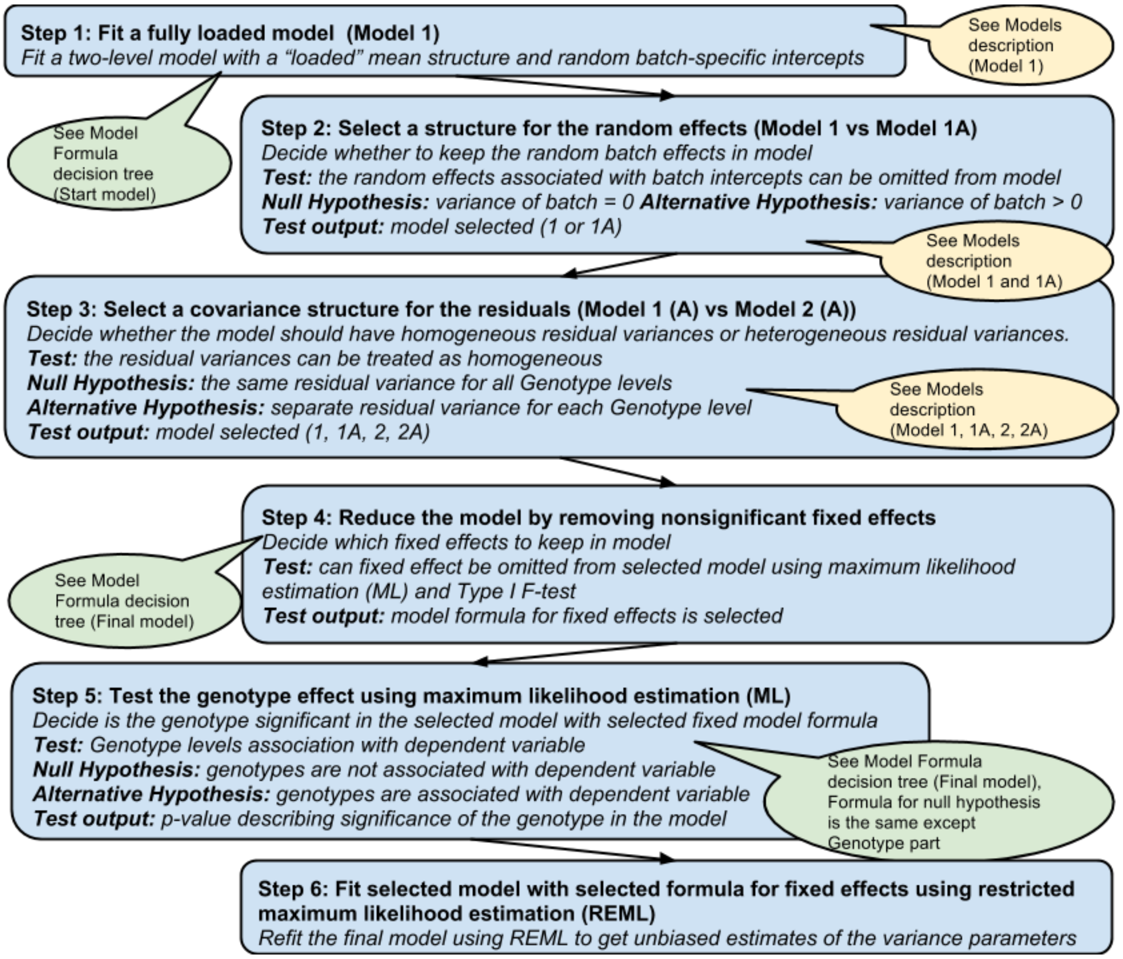
We reference to the **??** and **??** as to the models with "loaded" mean structure and random batch-specific intercepts or fully loaded model (see Figure **??**).

The final model construct is influenced by a number of criteria. All these criteria such as fixed effects, batch effect and the structure of residual variances can be either evaluated from the dataset or defined by user (see Figure **??**). The following criteria (effects) are considered:

- Batch effect (batch variation). Considered only when batch column is present in the dataset.
- Residual variances homogeneity where homogeneous residual variances means the variance for all genotype levels is considered equivalent.
- Body weight effect. Considered only when Eq2 is used.
- Gender effect. Considered only when there are more than one gender in the dataset.
- Genotype by gender interaction effect. Considered only when there are more than one gender in the dataset.

The selection of model rest on a batch effect (random effects) — is it in the dataset and if so is it significant or not — and a covariance structure for the residuals that can be homogeneous or heterogeneous (see Figure **??** Step 1-3). The selected model is modified by reducing non-significant effects (see Figure **??** Step 4 and Figure **??**).

When the final model is selected and reduced genotype effect is assessed by comparing a genotype and null model fitted with maximum likelihood evaluation method (ML). Finally, the final genotype model is refitted using restricted maximum likelihood evaluation method (REML) to get unbiased estimates of the variance parameters (see Figure **??** Step 5,6 and Figure **??**).



**Step 1: Fit a fully loaded model (Model 1)**
*Fit a two-level model with a "loaded" mean structure and random batch-specific intercepts*

See Models description (Model 1)

See Model Formula decision tree (Start model)

**Step 2: Select a structure for the random effects (Model 1 vs Model 1A)**
*Decide whether to keep the random batch effects in model*
***Test:** the random effects associated with batch intercepts can be omitted from model*
***Null Hypothesis:** variance of batch = 0 **Alternative Hypothesis:** variance of batch > 0*
***Test output:** model selected (1 or 1A)*

See Models description (Model 1 and 1A)

**Step 3: Select a covariance structure for the residuals (Model 1 (A) vs Model 2 (A))**
*Decide whether the model should have homogeneous residual variances or heterogeneous residual variances.*
***Test:** the residual variances can be treated as homogeneous*
***Null Hypothesis:** the same residual variance for all Genotype levels*
***Alternative Hypothesis:** separate residual variance for each Genotype level*
***Test output:** model selected (1, 1A, 2, 2A)*

See Models description (Model 1, 1A, 2, 2A)

**Step 4: Reduce the model by removing nonsignificant fixed effects**
*Decide which fixed effects to keep in model*
***Test:** can fixed effect be omitted from selected model using maximum likelihood estimation (ML) and Type I F-test*
***Test output:** model formula for fixed effects is selected*

See Model Formula decision tree (Final model)

**Step 5: Test the genotype effect using maximum likelihood estimation (ML)**
*Decide is the genotype significant in the selected model with selected fixed model formula*
***Test:** Genotype levels association with dependent variable*
***Null Hypothesis:** genotypes are not associated with dependent variable*
***Alternative Hypothesis:** genotypes are associated with dependent variable*
***Test output:** p-value describing significance of the genotype in the model*

See Model Formula decision tree (Final model), Formula for null hypothesis is the same except Genotype part

**Step 6: Fit selected model with selected formula for fixed effects using restricted maximum likelihood estimation (REML)**
*Refit the final model using REML to get unbiased estimates of the variance parameters*

Figure 2: MM framework steps: model selection process and model reducing by using significance of fixed effects.

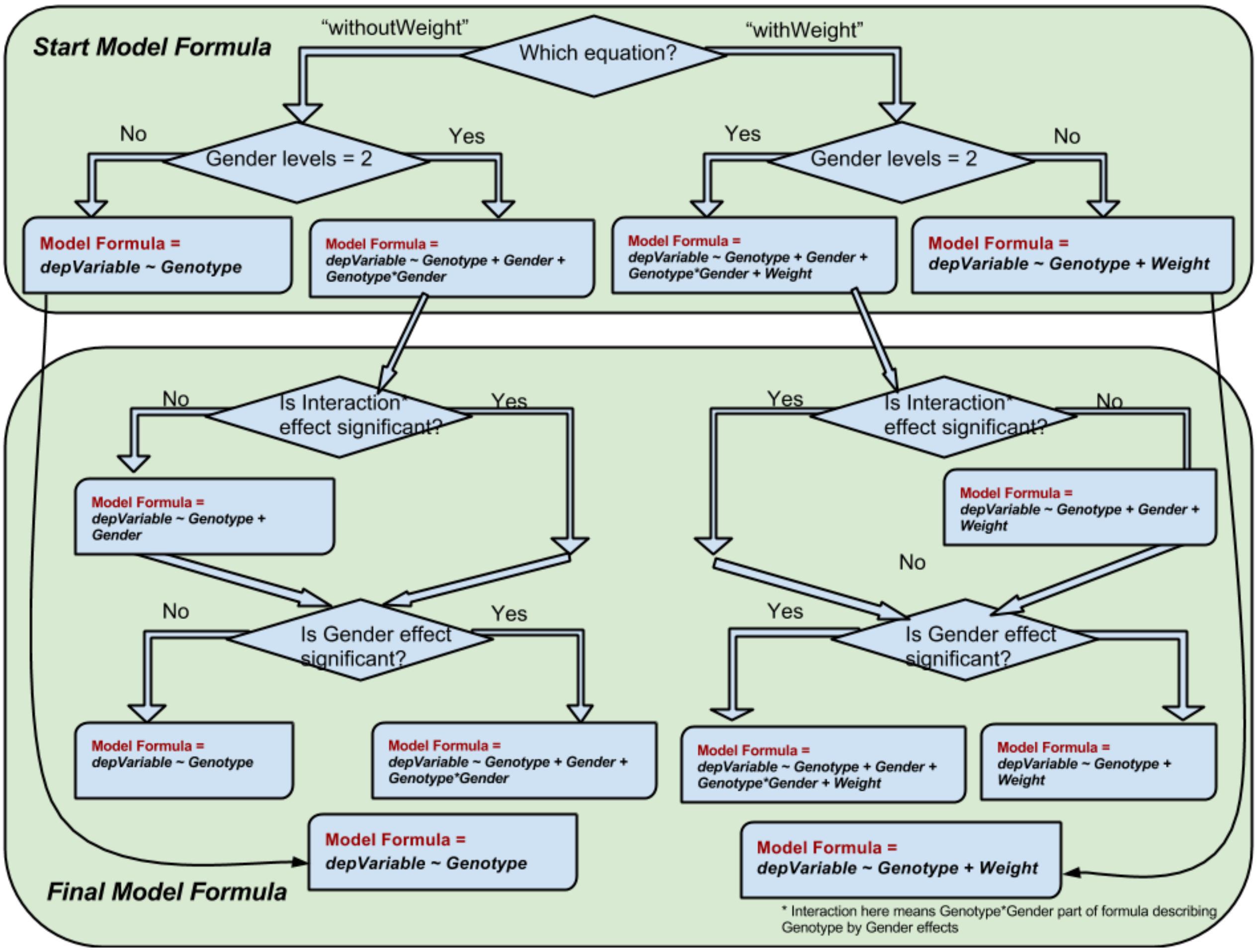
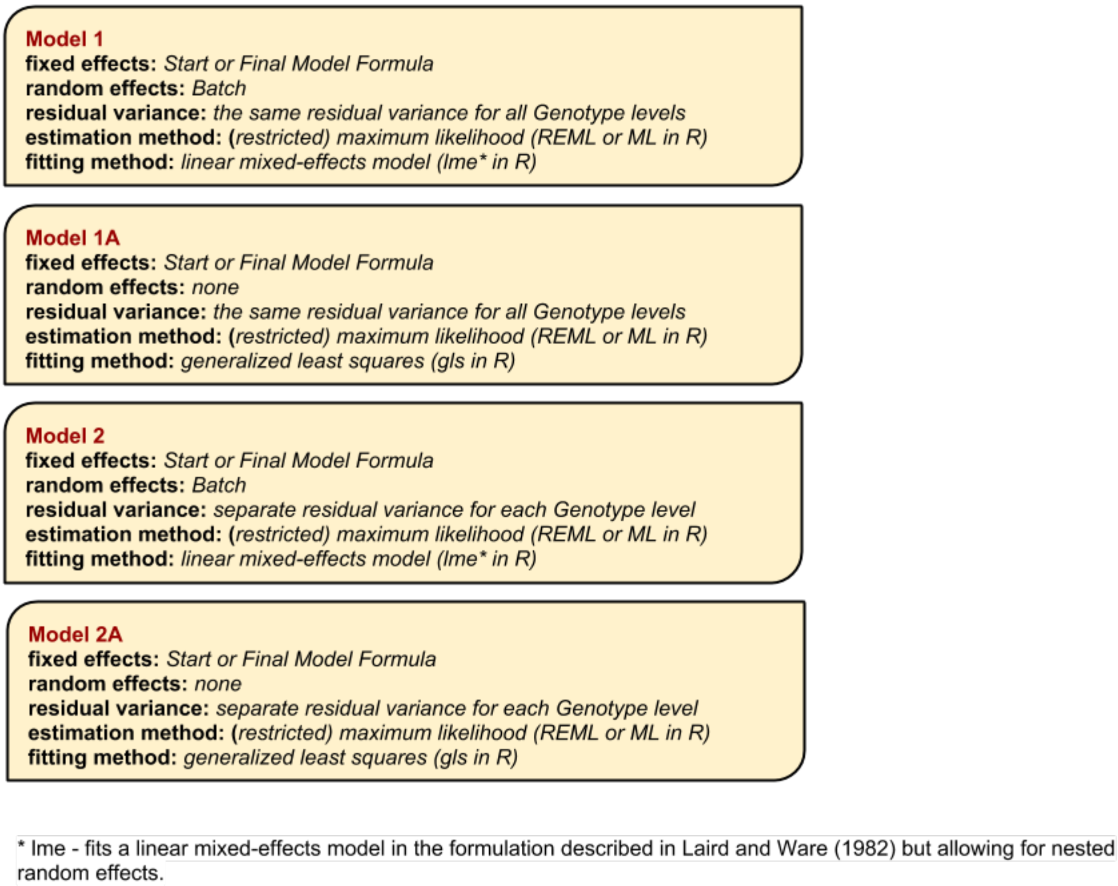

Start Model Formula
"withoutWeight"
Which equation?
"withWeight"
No
Gender levels = 2
Yes
Yes
Gender levels = 2
No
Model Formula = depVariable ~ Genotype
Model Formula = depVariable ~ Genotype + Gender + Genotype*Gender
Model Formula = depVariable ~ Genotype + Gender + Genotype*Gender + Weight
Model Formula = depVariable ~ Genotype + Weight
No
Is Interaction* effect significant?
Yes
Yes
Is Interaction* effect significant?
No
Model Formula = depVariable ~ Genotype + Gender
Model Formula = depVariable ~ Genotype + Gender + Weight
No
No
Is Gender effect significant?
Yes
Yes
Is Gender effect significant?
Model Formula = depVariable ~ Genotype
Model Formula = depVariable ~ Genotype + Gender + Genotype*Gender
Model Formula = depVariable ~ Genotype + Gender + Genotype*Gender + Weight
Model Formula = depVariable ~ Genotype + Weight
Model Formula = depVariable ~ Genotype
Model Formula = depVariable ~ Genotype + Weight
Final Model Formula
* Interaction here means Genotype*Gender part of formula describing Genotype by Gender effects

Figure 3: MM framework: start model formula and final model formula creation based on the dataset and significances of the effects (can be estimated or defined by user).



**Model 1**
**fixed effects:** *Start or Final Model Formula*
**random effects:** *Batch*
**residual variance:** *the same residual variance for all Genotype levels*
**estimation method: (***restricted) maximum likelihood (REML or ML in R)*
**fitting method:** *linear mixed-effects model (lme* in R)*

**Model 1A**
**fixed effects:** *Start or Final Model Formula*
**random effects:** *none*
**residual variance:** *the same residual variance for all Genotype levels*
**estimation method: (***restricted) maximum likelihood (REML or ML in R)*
**fitting method:** *generalized least squares (gls in R)*

**Model 2**
**fixed effects:** *Start or Final Model Formula*
**random effects:** *Batch*
**residual variance:** *separate residual variance for each Genotype level*
**estimation method: (***restricted) maximum likelihood (REML or ML in R)*
**fitting method:** *linear mixed-effects model (lme* in R)*

**Model 2A**
**fixed effects:** *Start or Final Model Formula*
**random effects:** *none*
**residual variance:** *separate residual variance for each Genotype level*
**estimation method: (***restricted) maximum likelihood (REML or ML in R)*
**fitting method:** *generalized least squares (gls in R)*

* lme - fits a linear mixed-effects model in the formulation described in Laird and Ware (1982) but allowing for nested random effects.

Figure 4: MM framework: different models that are considered.

### 3.2.2 Implementation

There are two functions in the PhenStat package that implements the mixed model framework:

- *startModel* function evaluates model's criteria and stores the result in the *PhenTestResult* object;
- *finalModel* function builds the final model using the model's criteria from *PhenTestResult* object and fits the model using restricted maximum likelihood method (REML).

By default, both functions will be called from *testDataset* manager one after another that is why *startModel* function's arguments and specific for MM method *testDataset* function's arguments concur. In the text above we mention *startModel* function's arguments only.

The equation type is defined by *startModel* function's argument *equation* that can take value "withWeight" which is default one and "withoutWeight". The argument defines the presence or absence of body weight effect in the model (see **??** and **??**). In case when there are no body weight records in the dataset *startModel* sets *equation* argument to "withoutWeight" automatically.

*startModel* function creates start fully loaded model and modifies it after testing of different hypothesis. As was described in the previous theory section the model view is influenced by the number of criteria. Each one criteria or effect (body weight effect, residual variances homogeneity, gender effect, genotype by gender interaction effect, batch effect) is evaluated and TRUE/FALSE values are assigned to the appropriate sections of *PhenTestResult* object based on evaluation results. TRUE value means that effect is significant and will be modelled. FALSE value means deletion of the effect from the model.

The package allows to assign user defined values to the effects of the model. If user would like to assign TRUE/FALSE values to the effects of the model that differ from calculated ones then (s)he has to define *keepList* argument of *startModel* functions which is a list of TRUE/FALSE values for each one criterion in the following order: is batch effect significant, are residual variances homogeneous, is body weight effect significant, is gender effect significant, is gender by genotype interaction effect significant. For instance, keepList=c(TRUE, TRUE, TRUE, TRUE, TRUE) defines the fully loaded model will all possible fixed effects with homogeneous residual variances; in turn keepList=c(FALSE, FALSE, TRUE, TRUE, TRUE) defines the fully loaded model without random effects and with heterogeneous residual variances.

*startModel* function 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 prints out both calculated and user defined effects (only when *outputMessages* argument is set to TRUE) for the user's convenience. Note: user defined effects have a priority over calculated (evaluated) effects.

The result of the *startModel* function is MM start model with reduced non-significant effects stored in the *PhenTestResult* object together with the evaluated or user defined effects.

The next step of MM framework: evaluation of genotype effect and fitting of selected model using REML is implemented in package's function *finalModel*. The results are added into the *PhenTestResult* object. *PhenTestResult* object at the end of the MM framework contains model formula, significances of the effects, genotype evaluation results and model fitting results including effect sizes.

By default both functions (*startModel* and *finalModel*) will be called from *testDataset* manager one after another. We've made this logical separation of functionality in order to add more flexibility for the statisticians. Basically, it means that a user can check the evaluation of fixed effects and the selected model before final model fitting. This kind of "debugging" functionality allows to change some of the arguments of functions and start the model building process from scratch if needed.

We believe that described above possibility to change mixed models framework behaviour will help users to go deeper into details of the modelling process, do debugging and compare the results from different models.

```
# Default behaviour
> result <- testDataset(test,depVariable="Bone.Area", equation="withoutWeight")
Information:
Dependent variable: 'Bone.Area'.

Information:
Method: Mixed Model framework.

Information:
Calculated values for model effects are: keepBatch=TRUE, keepVariance=TRUE,
keepWeight=FALSE, keepGender=TRUE, keepInteraction=FALSE.

Information:
```

```
Equation: 'withoutWeight'.

Information:
Perform all MM framework stages: startModel and finalModel

# Perform each step of the MM framework separatly
> result <- testDataset(test,depVariable="Bone.Area", equation="withoutWeight",callAll=FALSE)

Information:
Dependent variable: 'Bone.Area'.

Information:
Method: Mixed Model framework.

Information:
Calculated values for model effects are: keepBatch=TRUE, keepVariance=TRUE,
keepWeight=FALSE, keepGender=TRUE, keepInteraction=FALSE.

Information:
Equation: 'withoutWeight'.

# Estimated model effects
> result$model.effect.batch
[1] TRUE
> result$model.effect.variance
[1] TRUE
> result$model.effect.weight
[1] FALSE
> result$model.effect.gender
[1] TRUE
> result$model.effect.interaction
[1] FALSE

> result$numberGenders
[1] 2

# 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)

Information:
Dependent variable: 'Bone.Area'.

Information:
Method: Mixed Model framework.

Information:
User's values for model effects are: keepBatch=TRUE, keepVariance=TRUE,
keepWeight=FALSE, keepGender=TRUE, keepInteraction=TRUE.
```

```
Information:
Calculated values for model effects are: keepBatch=TRUE, keepVariance=TRUE,
keepWeight=FALSE, keepGender=TRUE, keepInteraction=FALSE.

Warning:
Calculated values differ from user defined values for model effects.

Information:
Equation: 'withoutWeight'.

> result <- finalModel(result)

> summaryOutput(result)
...
```

### 3.2.3 Diagnostics

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

The first one performs diagnostic test for MM quality of fit: normality test (Cramer-von Mises test) for the two genotype levels residuals, BLUPs (best linear unbiased prediction) normality test, rotated residual test (last two if applicable). There is only one argument of the function which is *PhenTestResult* object. There are no arguments checks assuming that function is called internally from the *finalModel* function. Otherwise should be used with precaution.

*testFinalModel* returns list of the following values:

- Reference genotype value.
- Normality test result (p-value) for the reference genotype's residuals.
- Test genotype value.
- Normality test result (p-value) for the test genotype's residuals.
- BLUPs normality test result (p-value); applicable only when there is batch random effects in the model.
- "Rotated" residuals normality test result (p-value); applicable only when there is batch random effects in the model.

BLUP in statistics is best linear unbiased prediction and is used in linear mixed models for the estimation of random effects. See tutorial BLUPs for more details.

"Rotated" residuals are constructed by multiplying the estimated marginal residual vector by the Cholesky decomposition of the inverse of the estimated marginal variance matrix. The resulting "rotated" residuals are used to construct an empirical cumulative distribution function and pointwise standard errors. See Cholesky Residuals for Assessing Normal Errors in a Linear Model with Correlated Outcomes: Technical Report for more details about "rotated" residuals.

### 3.2.4 Classification Tag

*classificationTag* function returns a classification tag to assign a sexual dimorphism assessment of the phenotypic change from the results of MM framework.

```
> testFinalModel(result)
[1] "+/+"                "0.0560133469740866" "Sparc/Sparc"
[4] "0.816672883686998"  "0.345325318416593"  "0.0480124939288989"
> classificationTag(result)
[1] "With phenotype threshold value 0.01 - both sexes equally"
```

## 3.3 Fisher Exact Test Framework

The Fisher Exact Test is implemented with basic R functions from the stats package after the construction of count matrices (also called chi squared tables) from the dataset.

Together with count matrices we calculate also percentage matrices and statistics for the chi squared tables (using "vcd" R package "Visualizing Categorical Data"). As a measure of change we calculate the maximum effect sizes.

From the chi squared table statistical significance is assessed using a Fisher Exact Test whilst the biological significance is estimated by an effect size.

This is calculated separately for 3 subsets (if there are multiple gender values in the dataset):

- combined dataset (regardless the gender values),
- males only subset,

- females only subset.

A Fisher Exact Test was chosen as most abnormal phenotype traits are rare event thus the signal is low. Batch is not considered significant because day to day variation does not effect abnormality call for these types of variables.

All results are stored in *PhenTestResult* object:

```
> dataset_cat <- read.csv("./PhenStat/extdata/test_categorical.csv")
> test_cat <- PhenList(dataset_cat,testGenotype="Aff3/Aff3")

Warning:
Dataset’s column ’Assay.Date’ has been renamed to ’Batch’ and will be used for the batch effect modeling.

Warning:
Dataset has been cleaned by filtering out records with genotype value
other than test genotype ’Aff3/Aff3’ or reference genotype ’+/+’.

Warning:
Dataset’s ’Weight’ column is missed.
You can define ’dataset.colname.weight’ argument to specify column
for the weight effect modeling. Otherwise you can only use mixed model equation ’withoutWeight’.

Information:
Dataset’s ’Genotype’ column has following values: ’+/+’, ’Aff3/Aff3’

Information:
Dataset’s ’Gender’ column has following value(s): ’Female’, ’Male’

> result_cat <- testDataset(test_cat,
 depVariable="Thoracic.Processes",
 method="FE")

Information:
Dependent variable: ’Thoracic.Processes’.

Information:
Method: Fisher Exact Test framework.

> result_cat$depVariable
[1] "Thoracic.Processes"
> result_cat$method
[1] "FE"
> result_cat$numberGenders
[1] 2

# Chi squared table for all data
> result_cat$model.output$count_matrix_all
```

```
         +/+ Aff3/Aff3
Abnormal 144        12
Normal   755         1

# Chi squared table for males only records
> result_cat$model.output$count_matrix_male

         +/+ Aff3/Aff3
Abnormal  61         5
Normal   392         1

# Percentage matrix for all data
> result_cat$model.output$percentage_matrix_all

         +/+ Aff3/Aff3 ES change
Abnormal  16        92        76
Normal    84         8        76

# Percentage matrix for females only records
> result_cat$model.output$percentage_matrix_female

         +/+ Aff3/Aff3 ES change
Abnormal  19       100        81
Normal    81         0        81

# Matrix statistics for all data
> result_cat$model.output$stat_all

                    X^2 df   P(> X^2)
Likelihood Ratio 36.466  1 1.5536e-09
Pearson          52.600  1 4.0890e-13

Phi-Coefficient   : 0.24
Contingency Coeff.: 0.234
Cramer's V        : 0.24

# Matrix statistics for males only records
> result_cat$model.output$stat_male

                    X^2 df  P(> X^2)
Likelihood Ratio 14.610  1 1.322e-04
Pearson          23.479  1 1.263e-06

Phi-Coefficient   : 0.226
Contingency Coeff.: 0.221
Cramer's V        : 0.226

# Effect size for all data
```

```
> result_cat$model.output$ES

[1] 76

# Effect size for females only records
> result_cat$model.output$ES_female

[1] 81

# Fisher Exact Test results for all data
> result_cat$model.output$all

Fisher's Exact Test for Count Data

data:  count_matrix_all
p-value = 4.844e-09
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.0003770171 0.1096287774
sample estimates:
odds ratio
 0.0159923

# p-value for all data
> result_cat$model.output$all$p.value

[1] 4.844291e-09
```

The same data as shown in examples can be obtained by using output functions of the package: *summaryOutput*, *vectorOutput* and *vectorOutputMatrices*. See section **??** for more details.

# 4 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.

## 4.1 Summary Output

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:

```
 # Mixed Model framework
> test <- PhenList(dataset=read.csv("./PhenStat/extdata/test.csv"),
            testGenotype="Sparc/Sparc",outputMessages=FALSE)
> result <- testDataset(test,
            depVariable="Lean.Mass",outputMessages=FALSE)
> summaryOutput(result)

Test for dependent variable: Lean.Mass
Method: Mixed Model framework

Was batch significant? TRUE
Was variance equal? FALSE
Was there evidence of sexual dimorphism? no (p-value 0.102)
Final fitted model: Lean.Mass ~ Genotype + Gender + Weight
Model output:
Genotype effect: 0.371508943
Classification tag: With phenotype threshold value 0.01 - no significant change
                         Value  Std.Error  DF    t-value      p-value
(Intercept)          7.6111388 0.58862654 411 12.9303357 2.512303e-32
GenotypeSparc/Sparc -0.2914357 0.33047985 411 -0.8818562 3.783700e-01
GenderMale           1.6407343 0.18080930 411  9.0743913 4.791912e-18
Weight               0.3430502 0.01808121 411 18.9727422 4.147891e-58
```

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

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

Test for dependent variable: Thoracic.Processes
Method: Fisher Exact Test framework

Model output:
All data p-val: 4.84429148175386e-09
All data effect size: 76%
Males only p-val: 0.000286667802768362
Males only effect size: 70%
Females only p-val: 1.00779809539594e-05
Females only effect size: 81%

Matrix 'all':
         +/+ Aff3/Aff3
Abnormal 144        12
Normal   755         1

Percentage matrix 'all' statistics:
         +/+ Aff3/Aff3 ES change
Abnormal  16        92        76
Normal    84         8        76

Matrix 'all' statistics:
                    X^2 df   P(> X^2)
Likelihood Ratio 36.466  1 1.5536e-09
Pearson          52.600  1 4.0890e-13

Phi-Coefficient   : 0.24
Contingency Coeff.: 0.234
Cramer's V        : 0.24

Matrix 'males only':
...
```

## 4.2 Vector Format

*vectorOutput* function was developed for large scale application where automatic implementation would be required. As such, each value within the output vector is strictly defined and depends only on the statistical analysis method that has been used. The main idea here is that vector format is specified and is the same regardless the analysis framework.

```
> vectorOutput(result)
                                      Method 
                                    "MM - Eq2" 
                            Dependent variable 
                                  "Lean.Mass" 
                                Batch included 
                                       "TRUE" 
                Residual variances homogeneity 
                                      "FALSE" 
                         Genotype contribution 
                          "0.371508943144266" 
                             Genotype estimate 
                          "-0.29143571549456" 
                       Genotype standard error 
                          "0.330479850268177" 
```

```
        Genotype p-Val
   "0.378369997588029"
       Gender estimate
    "1.64073430331594"
 Gender standard error
   "0.180809296427475"
          Gender p-val
"4.79191190571249e-18"
       Weight estimate
   "0.343050209791982"
 Weight standard error
  "0.0180812139273457"
          Weight p-val
 "4.1478905048872e-58"
```

...

Vectors data contains the following values in the defined order:

1. ”Method”,
2. ”Dependent variable”,
3. ”Batch included”,
4. ”Residual variances homogeneity”,
5. ”Genotype contribution”,
6. ”Genotype estimate”,
7. ”Genotype standard error”,
8. ”Genotype p-Val”,
9. ”Gender estimate”,
10. ”Gender standard error”,
11. ”Gender p-val”,
12. ”Weight estimate”,
13. ”Weight standard error”,
14. ”Weight p-val”,
15. ”Gp1 genotype”,
16. ”Gp1 Residuals normality test”,
17. ”Gp2 genotype”,

18. ”Gp2 Residuals normality test”,
19. ”Blups test”,
20. ”Rotated residuals normality test”,
21. ”Intercept estimate”,
22. ”Intercept standard error”,
23. ”Interaction included”,
24. ”Interaction p-val”,
25. ”Gender FvKO estimate”,
26. ”Gender FvKO standard error”,
27. ”Gender FvKO p-val”,
28. ”Gender MvKO estimate”,
29. ”Gender MvKO standard error”,
30. ”Gender MvKO p-val”,
31. ”Classification tag”.

As was mentioned above *vectorOutput* format is the same for both frameworks. However, in case of ”FE” a lot of values are not defined. For example:

```
> vectorOutput(result_cat)

                                                 Method
                                    "Fisher Exact Test"
                                     Dependent variable
                                   "Thoracic.Processes"
                                         Batch included
                                                     NA
                         Residual variances homogeneity
                                                     NA
                                  Genotype contribution
                                                     NA
                                      Genotype estimate
                                                   "76"
                                Genotype standard error
                                                     NA
                                         Genotype p-Val
                                 "4.84429148175386e-09"
                                        Gender estimate
```

```
                                                       NA
                                    Gender standard error
                                                       NA
                                             Gender p-val
                                                       NA
                                          Weight estimate
                                                       NA
                                    Weight standard error
                                                       NA
                                             Weight p-val
                                                       NA
                                             Gp1 genotype
                                                    "+/+"
                            Gp1 Residuals normality test
                                                       NA
                                             Gp2 genotype
                                              "Aff3/Aff3"
                            Gp2 Residuals normality test
                                                       NA
                                               Blups test
                                                       NA
                        Rotated residuals normality test
                                                       NA
                                       Intercept estimate
                                                       NA
                                 Intercept standard error
                                                       NA
                                     Interaction included
                                                       NA
                                        Interaction p-val
                                                       NA
                                     Gender FvKO estimate
                                                     "81"
                               Gender FvKO standard error
                                                       NA
                                        Gender FvKO p-val
                                    "1.00779809539594e-05"
                                     Gender MvKO estimate
                                                     "70"
                               Gender MvKO standard error
                                                       NA
                                        Gender MvKO p-val
                                    "0.000286667802768362"
                                       Classification tag
"Significant in males, females and in combined dataset"
```

## 4.3 Count Matrices in Vector Format

There is an additional function to support FE framework: *vectorOutputMatrices* that returns values from count matrices in the vector format. We've limited the number of values for dependent variable up to 10. In the vector first three positions represent: dependent variable, genotype level 1 (reference genotype) and genotype level 2 (test genotype). Next 10 positions are used for the dependent variable levels. When there are less than 10 levels "NA" value is used. Next 20 positions represent combined count matrix values row after row. Males only count matrix values and females only count matrix values are coming after. Again "NA" is used when value is not present.

For the chi squared tables from example described in "Fisher Exact Test framework" subsection (see **??**) results of *vectorOutputMatrices* function look like this:

```
> vectorOutputMatrices(result_cat)
              Dependent variable                Gp1 Genotype (g1)
            "Thoracic.Processes"                            "+/+"
               Gp2 Genotype (g2)   Dependent variable level1 (l1)
                     "Aff3/Aff3"                       "Abnormal"
  Dependent variable level2 (l2)   Dependent variable level3 (l3)
                        "Normal"                               NA
  Dependent variable level4 (l4)   Dependent variable level5 (l5)
                              NA                               NA
  Dependent variable level6 (l6)   Dependent variable level7 (l7)
                              NA                               NA
 Dependent variable level18 (l8)        Dependent variable level9
                              NA                               NA
Dependent variable level10 (l10)                      Value g1_l1
                              NA                            "144"
                     Value g2_l1                      Value g1_l2
                            "12"                            "755"
                     Value g2_l2                      Value g1_l3
                             "1"                               NA
                     Value g2_l3                      Value g1_l4
                              NA                               NA
                     Value g2_l4                      Value g1_l5
                              NA                               NA
                     Value g2_l5                      Value g1_l6
                              NA                               NA
                     Value g2_l6                      Value g1_l7
                              NA                               NA
                     Value g2_l7                      Value g1_l8
                              NA                               NA
                     Value g2_l8                      Value g1_l9
                              NA                               NA
                     Value g2_l9                     Value g1_l10
```

```
                 NA
       Value g2_l10
                 NA
   Male Value g2_l1
                "5"
   Male Value g2_l2
                "1"
   Male Value g2_l3
                 NA
   Male Value g2_l4
                 NA
   Male Value g2_l5
                 NA
   Male Value g2_l6
                 NA
   Male Value g2_l7
                 NA
   Male Value g2_l8
                 NA
   Male Value g2_l9
                 NA
  Male Value g2_l10
                 NA
 Female Value g2_l1
                "7"
 Female Value g2_l2
                "0"
 Female Value g2_l3
                 NA
 Female Value g2_l4
                 NA
 Female Value g2_l5
                 NA
 Female Value g2_l6
                 NA
 Female Value g2_l7
                 NA
 Female Value g2_l8
                 NA
 Female Value g2_l9
                 NA
Female Value g2_l10
                 NA
```

```
                 NA
   Male Value g1_l1
               "61"
   Male Value g1_l2
              "392"
   Male Value g1_l3
                 NA
   Male Value g1_l4
                 NA
   Male Value g1_l5
                 NA
   Male Value g1_l6
                 NA
   Male Value g1_l7
                 NA
   Male Value g1_l8
                 NA
   Male Value g1_l9
                 NA
  Male Value g1_l10
                 NA
 Female Value g1_l1
               "83"
 Female Value g1_l2
              "363"
 Female Value g1_l3
                 NA
 Female Value g1_l4
                 NA
 Female Value g1_l5
                 NA
 Female Value g1_l6
                 NA
 Female Value g1_l7
                 NA
 Female Value g1_l8
                 NA
 Female Value g1_l9
                 NA
Female Value g1_l10
                 NA
```

# 5 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.

In order to create all possible graphics for the particular results we've created a function called *generateGraphs*. This function calls graphic generation functions specific for the framework and stores the results in the directory sepcified by the user.

```
> generateGraphs(phenTestResult=result,dir="./graphs",graphingName="Lean Mass",type="windows")
```

```
> generateGraphs(phenTestResult=result_cat,dir="./graphs_categorical",type="windows")
```

## 5.1 Graphics for Categorical Data

There is only one graphical output for FE framework: categorical bar plots. This graph allows a visual representation o

```
> categoricalBarplot(result_cat)
```

The example of bar plot is shown in Figure **??**.

## 5.2 Graphics for Continous Data

There is a bunch of graphic functions for the MM framework results. They can be divided into two main categories: dataset based graphs, results based graphs. There are three functions in the dataset based graphs category:

- *boxplotGenderGenotype* creates a box plot split by gender and genotype.
- *boxplotGenderGenotypeBatch* creates a box plot split by gender, 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.

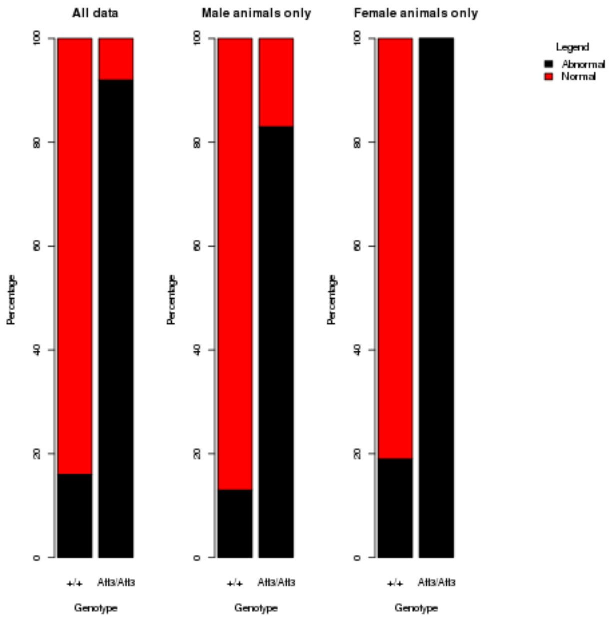


Figure 5: The PhenStat package's graphical output: categorical bar plot.

```
> boxplotGenderGenotype(test,depVariable="Lean.Mass",graphingName="Lean Mass")
> boxplotGenderGenotypeBatch(test,depVariable="Bone.Area",graphingName="Bone Area")
> scatterplotGenotypeWeight(test,depVariable="Bone.Mineral.Content",graphingName="BMC")
```

The example of bar plot split by gender and genotype is shown in Figure ??.

The example of bar plot split by gender, genotype and batch is shown in Figure ??.

The example of scatter plot of body weight versus dependent variable is shown in Figure ??.

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.

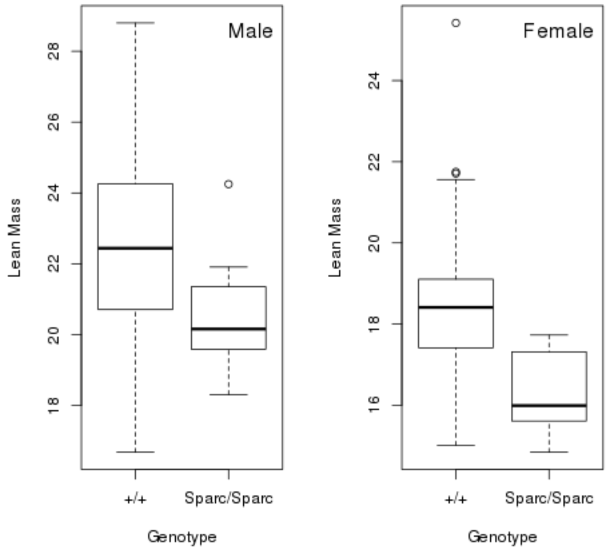


Figure 6: The PhenStat package's graphical output: box plot split by gender and genotype.

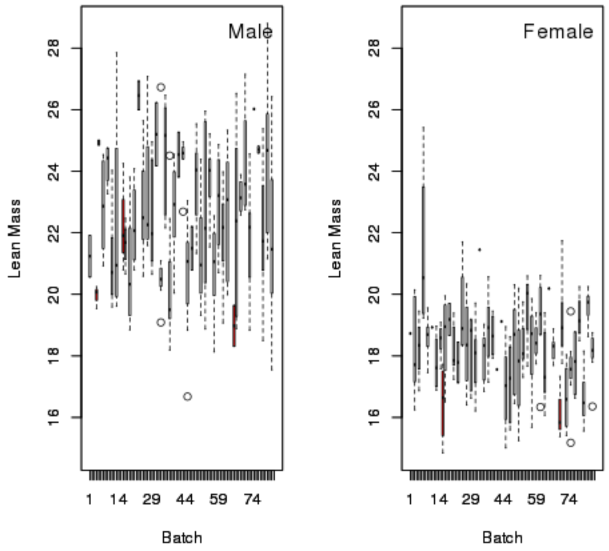


Figure 7: The PhenStat package's graphical output: box plot split by gender, genotype and batch.

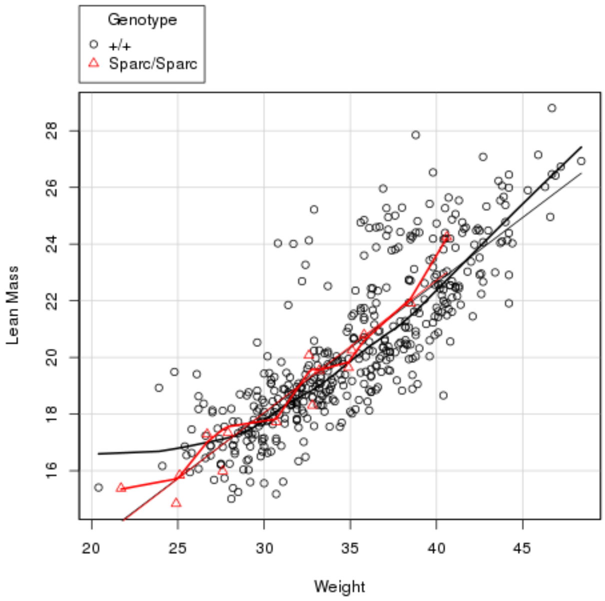


Figure 8: The PhenStat package's graphical output: scatter plot of body weight versus dependent variable.

- *boxplotResidualBatch* creates a box plot with residue versus batch split by genotype.

```
> qqplotGenotype(result)
> qqplotRandomEffects(result)
> qqplotRotatedResiduals(result)
> plotResidualPredicted(result)
> boxplotResidualBatch(result)
```

The example of Q-Q plot of residuals for each genotype is shown in Figure **??**.

The example of Q-Q plot of blups (best linear unbiased predictions) is shown in Figure **??**.

BLUP in statistics is best linear unbiased prediction and is used in linear mixed models for the estimation of random effects. See tutorial BLUPs for more details.

The example of Q-Q plot of rotated residuals is shown in Figure **??**.

"Rotated" residuals are constructed by multiplying the estimated marginal residual vector by the Cholesky decomposition of the inverse of the estimated marginal variance matrix. The resulting "rotated" residuals are used to con-

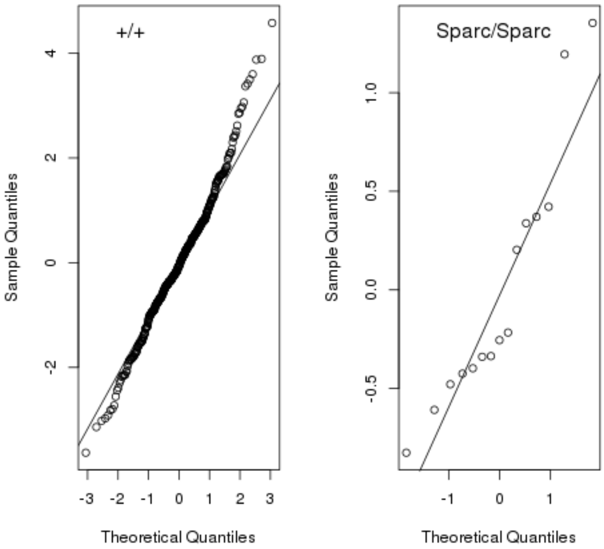


Figure 9: The PhenStat package's graphical output: Q-Q plot of residuals for each genotype.

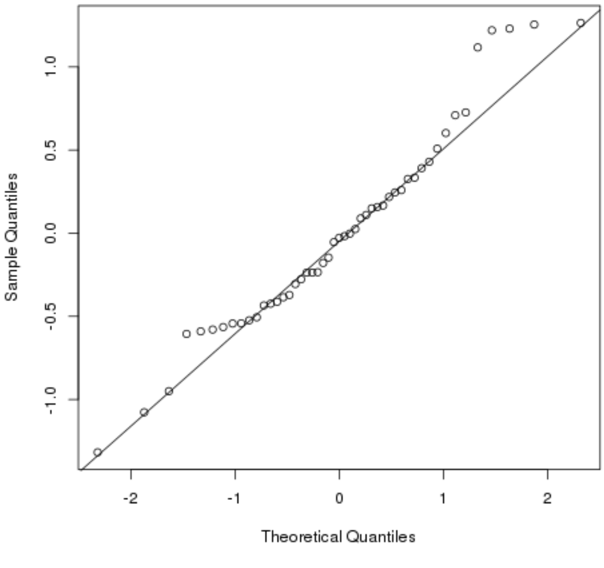


Figure 10: The PhenStat package's graphical output: Q-Q plot of blups (best linear unbiased predictions).

struct an empirical cumulative distribution function and pointwise standard errors. See Cholesky Residuals for Assessing Normal Errors in a Linear Model with Correlated Outcomes: Technical Report for more details about "rotated" residuals.

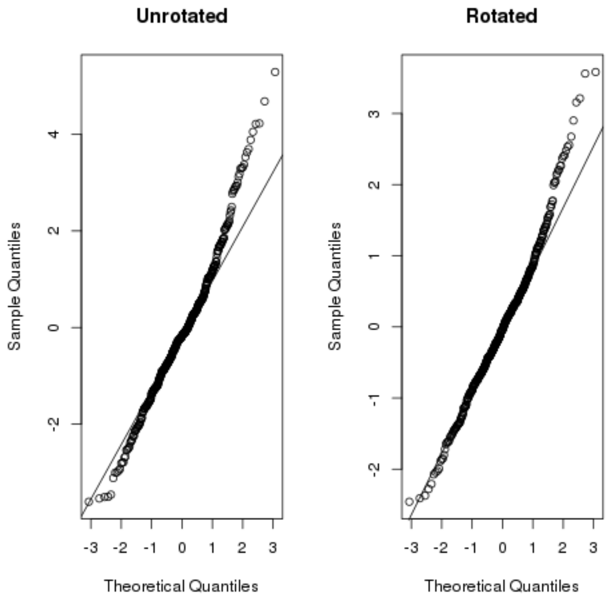


Figure 11: The PhenStat package's graphical output: Q-Q plot of "rotated" residuals.

The example of residual-by-predicted plot is shown in Figure ??. Residuals, differences between fitted and real values, are plotted against the prediced (fitted) values of dependent varibale. A residual-by-predicted plot can be used to diagnose nonlinearity or nonconstant error variance. It is also can be used to find outliers.

Here are the characteristics of a residual-by-predicted plot when model fitness is close to the ideal and what they suggest about the appropriateness of the model:

- The residuals are arranged randomly around the 0 line. This suggests that the assumption that the relationship is linear is reasonable.
- The residuals roughly form a "horizontal band" around the 0 line. This suggests that the variances of the error terms are equal.
- No one residual outstands from the basic random pattern of residuals. This suggests that there are no outliers. See Regression Methods for

more details.

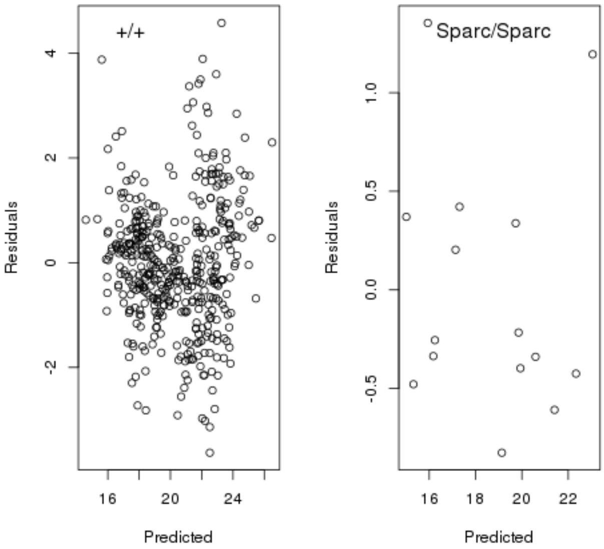


Figure 12: The PhenStat package's graphical output: residual-by-predicted plot split by genotype.

The example of box plot with residue versus batch split by genotype is shown in Figure ??. This allows assessment that the residual behaviour for all batches is within natural deviation but do not differ a lot and the model is fitting the data well.

# 6 Case Studies

## 6.1 PhenStat Integration with Database

## 6.2 PhenStat Example Using Cluster

If someone would like to analyse all variables in the dataset and has a cluster available for such kind of job then here is an example of PhenStat package usage.

First, the function that runs on each cluser's node and stores the results in particular directory is created. This function is based on the section *dataset.stat* of the *PhenList* object.

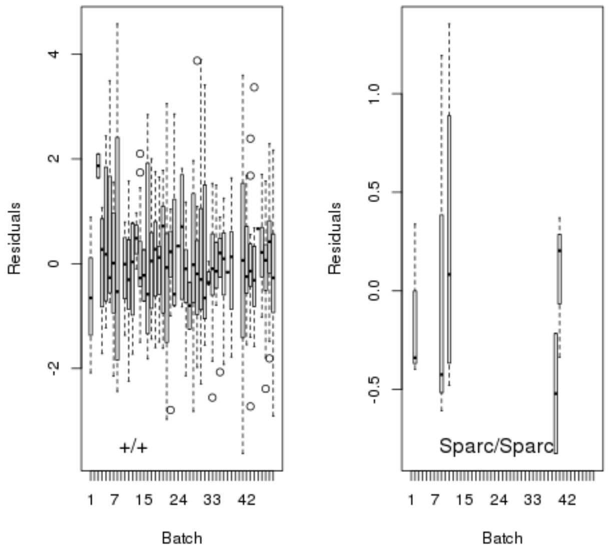


Figure 13: The PhenStat package's graphical output: box plot with residue versus batch split by genotype.

```
PhenStatCluster<-function(phenList,i){
    # reads variable names from dataset.stat table
    variable <- as.character(phenList$dataset.stat$Variables[i])

    # checks if variable is continuous again by using dataset.stat table
    isContinuous <- phenList$dataset.stat$Continuous[i]

    # skip the analysis for Batch and Genotype variables
    if (!(variable %in% c("Batch","Genotype"))){
      if (isContinuous && !(variable %in% c("Weight")))
  # performs MM framework for continuous data
  result <- testDataset(phenList, variable, method="MM",outputMessages=FALSE)
      else
  if (!isContinuous){
      # performs FET framework for categorical data
      result <- testDataset(phenList, variable, method="FE",outputMessages=FALSE)
   }
  else
      # performs MM framework for weight variable
      result <- testDataset(phenList, variable, method="MM",equation="withoutWeight",outputMessages=FALSE)

      write(vectorOutput(result),paste("./",variable,".txt",sep="")) # stores the results
    }
}
```

We are planning to analyse each one variable of the dataset by using a cluster. Each one cluster node has to have sourced function *PhenStatCluster* and loaded *PhenStat* library. *PhenList* object with datatset to analyse should also be available for every cluster node.

```
# cluster preparation
# set current folder
setwd("/some/folder/where/you/perform/processing")

# create logs folder in it
dir.create(paste(getwd(), "/logs", sep=""))

# define tasks
tasks <- c(1:length(test$dataset.stat$Variables))

# load snow
# snow creates and manages clusters

library(snow)
# create cluster
cluster = makeCluster(length(tasks), type="...") # type values: MPI, RCLOUD, etc.

# Setup cluster nodes
# set current folder on each node
clusterEvalQ(cluster, setwd("/some/folder/where/you/do/processing"))

# create logs and forward output to the log files
clusterEvalQ(cluster, try({ fn = paste(getwd(), "/logs/", Sys.info()[4], "-", Sys.getpid(), ".log", sep="");
o <- file(fn, open = "w"); sink(o); sink(o, type = "message"); }))

# test output is routed to the logs
clusterEvalQ(cluster, message("message - OK"))
clusterEvalQ(cluster, cat("cat - OK"))

# load package and source function for each node
clusterEvalQ(cluster, library(PhenStat))
clusterEvalQ(cluster, source("/path to the source/PhenStatCluster.R"))

# export PhenList object to make it available for every node
clusterExport(cluster, "test")

# finally apply function for each one variable within the dataset
clusterApplyLB(cluster, tasks, function(x){ message("---------- processing ",
test$dataset.stat$Variables[x], " ----------"); try(PhenStatCluster(test,x)); })

# clean up
stopCluster(cluster)
rm(cluster)
clusterCleanup()
```

The output is avaialble in the specified directory: "/some/folder/where/you/perform/processing". For each variable from the dataset the output file with results in vector format is created.

# References

Gentleman,R., Carey,V., Huber,W., Irizarry,R., Dudoit,S. (2008) Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Springer. ISBN 978-0-387-25146-2.

Gentleman,R. (2008) R Programming for Bioinformatics. Chapman & Hall\CRC. ISBN 978-1-4200-6367-7.

Hahne,F., Huber,W., Gentleman,R., Falcon,S. (2008). Bioconductor Case Studies. Springer. ISBN 978-0-387-77239-4.

Karp,N., Melvin,D., Sanger Mouse Genetics Project, Mott,R. (2012) Robust and Sensitive Analysis of Mouse Knockout Phenotypes, *PLoS ONE*,, **7**(12), e52410, doi:10.1371/journal.pone.0052410.

West,B., Welch,K., Galecki,A. (2007) Linear Mixed Models: A practical guide using statistical software. Chapman & Hall\CRC. ISBN 978-1-584-88480-4.