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

Natalja Kurbatova, Natasha Karp, Jeremy Mason

Last revised: 11th October 2013

Contents

1 Introduction **4**

2 Data Processing with PhenList Function **6**
2.1 Terminology Unification 7
2.2 Filtering 8
2.3 Dataset Checks 11
2.4 PhenList Object 12

3 Statistical Analysis **14**
3.1 Manager for Analysis Methods – *testDataset* function 14
3.2 Mixed Model Framework 15
3.2.1 Theory 16
3.2.2 Implementation 20
3.2.3 Diagnostics 23
3.2.4 Classification Tag 24
3.3 Fisher Exact Test Framework 24
3.3.1 Classification Tag 28

4 Output of Results **29**
4.1 Summary Output 29
4.2 Vector Format 31
4.3 Count Matrices in Vector Format 35

5 Graphics **37**
5.1 Graphics for Categorical Data 37
5.2 Graphics for Continuous Data 37

6 Case Studies **44**
6.1 PhenStat Usage Example Continuous Data 44
6.1.1 Loading the data and initial steps of analysis 44
6.1.2 Exploring and understanding the output 45
6.1.3 Assessment of raw data and distribution characteristics 46
6.1.4 Assessment of model fit 48
6.1.5 Including weight as a covariate in the model fitting process 53
6.1.6 Additional model diagnostics when weight is included . 55
6.2 PhenStat Usage Example Categorical Data 56
6.2.1 Loading the data and initial steps of analysis 57
6.2.2 Visualisation of data 57
6.2.3 Understanding the *summaryOutput* 58

6.2.4 Understanding the maximum effect size reported 59
6.2.5 A dependent variable with little variation 59
6.3 PhenStat Integration with Database 61
6.4 PhenStat Example Using Cluster 61

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 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 420 lines of historic mouse phenotyping data.

The package consists of three stages as shown in Figure 1:

1. Dataset processing: includes checking, cleaning and terminology unification procedures and is completed by function *PhenList* which creates a *PhenList* object.
2. Statistical analysis: is managed by function *testDataset* and consists of Mixed Model or Fisher Exact framework implementations. The results are stored in *PhenTestResult* object. Potentially this layer can be extended adding new statical methods (dotted box in Figure 1).

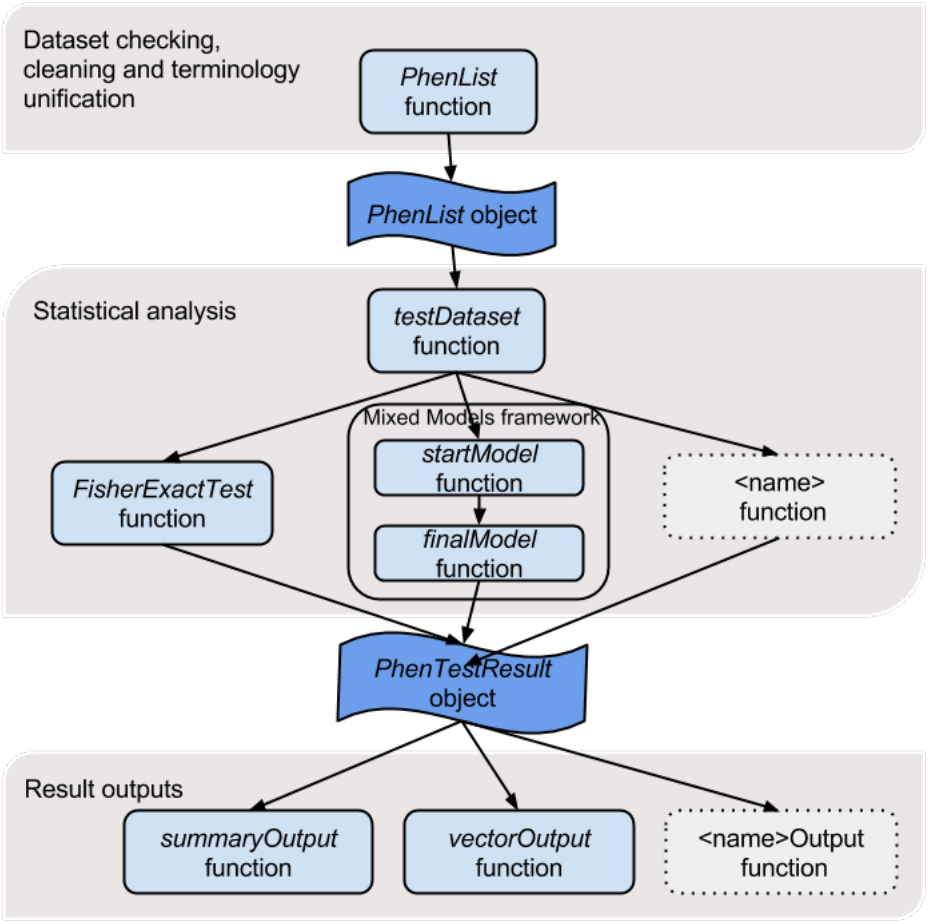

Figure 1: The PhenStat package's three stage structure: dataset processing, analysis, and result output. Dotted boxes show the place-holders for new functions that could implement other methods for data analysis and/or output of results.

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 1).

The estimates of the package run time depends a lot on a size of a dataset used for analysis. In average the analysis **run time** regardless the method used and the size of the dataset is **1.34 seconds**.

2 Data Processing with PhenList Function

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. 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 (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 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 sets 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 creates a copy of the dataset and then uses internal 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 the user 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 the user to define column name within dataset for the genotype info if this column name is other than "Genotype",
- *dataset.colname.gender* allows the user 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 the user 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 the user to specify value used as missing value in the dataset if other than NA,
- *dataset.values.male* allows the user to define value used to label "males" in the dataset if other than "Male",
- *dataset.values.female* allows the user 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 the *PhenList* function argument *dataset.clean* is set then all records with genotype values others than reference or test genotype are filtered out. The user may also 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 1 when *hemiGenotype* argument of the

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

Table 1: Example of the dataset with sex linked genes

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" 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 2.3) 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 also takes place when there are records that do not have at least two records in the dataset with the same genotype and gender values.

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

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

Table 2: Example of the dataset with 3 gender values

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 shows the results of the dataset described in the previous section 2.2 : 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 **********
```

Many checking failures are avoided when *dataset.clean* argument of the *PhenList* function is set to TRUE (default value). A few examples are given in this and in the previous section 2.2.

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 shows how to print out the whole cleaned dataset and how to view the statistics about it (output is shown in Table 3).

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

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

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

Table 3 shows the content of the *dataset.stat* section and describes the data focusing on the columns of the dataset. Each column is 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.

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

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

PhenList object has stored many characteristics about the 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.

An example 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 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".

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 (see Figure 1).

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 the dependent variable.

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

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

1. *depVariable* column 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, the framework can comprise of more than one stage. For instance, MM framework has two stages.

testDataset function's argument *callAll* instructs the package 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 instructs the *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 (such as the Fisher Exact Test case) then *testDataset* function runs that single stage regardless of the *callAll* argument's value. See example of *callAll* argument in the section 3.2.2. 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 Eq1. for the model including weight and Eq2. for the model without weight).

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

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

We reference to the Eq1 and Eq2 as to the models with "loaded" mean structure and random batch-specific intercepts or fully loaded model (see Figure 2).

The final model construct is influenced by a number of criteria. 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 3). 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 is influenced by the batch effect (random effects) — is batch in the dataset, and if so, is it significant — and a covariance structure for the residuals that can be homogeneous or heterogeneous (see Figure 2 Step 1-3). The selected model is modified by reducing non-significant effects (see Figure 2 Step 4 and Figure 3).

When the final model is selected and reduced genotype effect is assessed by comparing a genotype and null model fitted with maximum likelihood

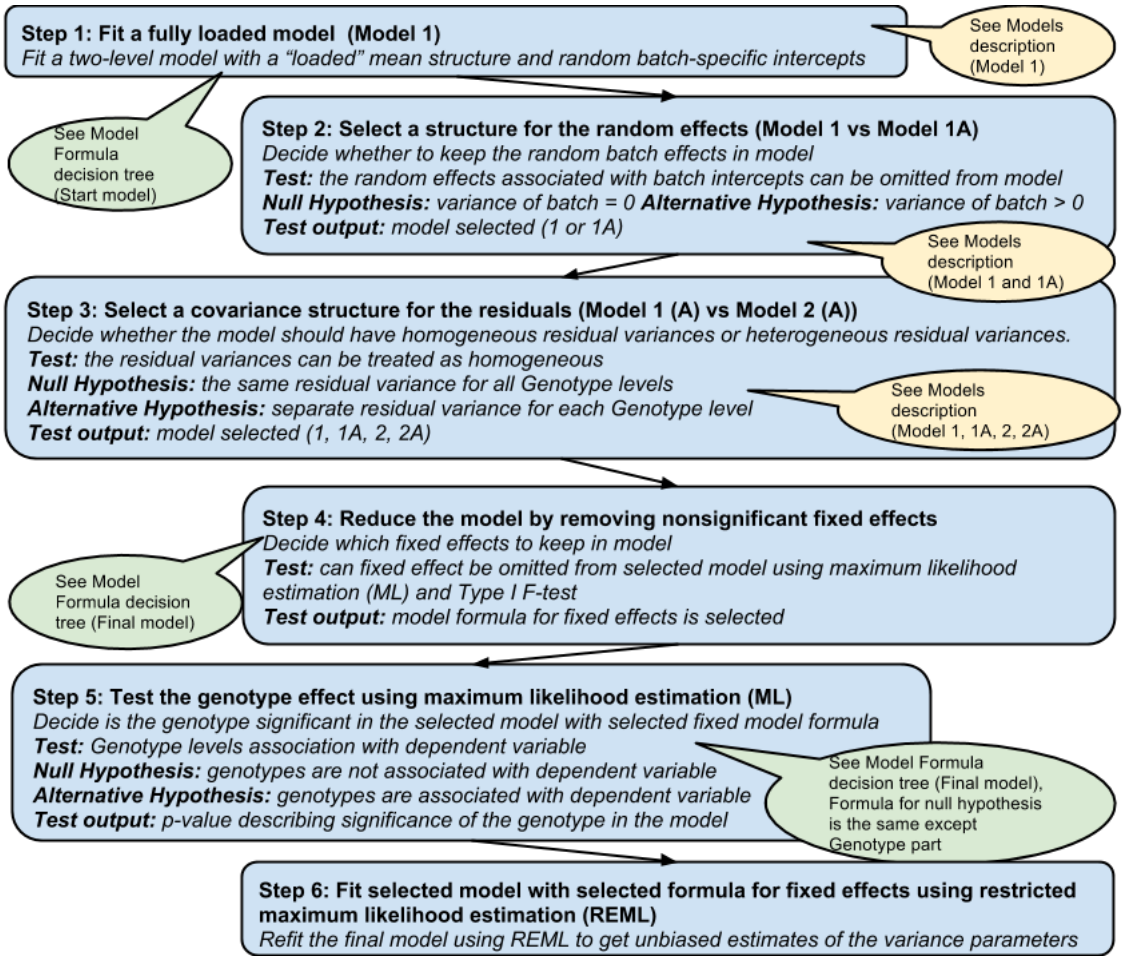

Figure 2: MM framework steps: model selection process and model reducing by using significance of fixed effects.

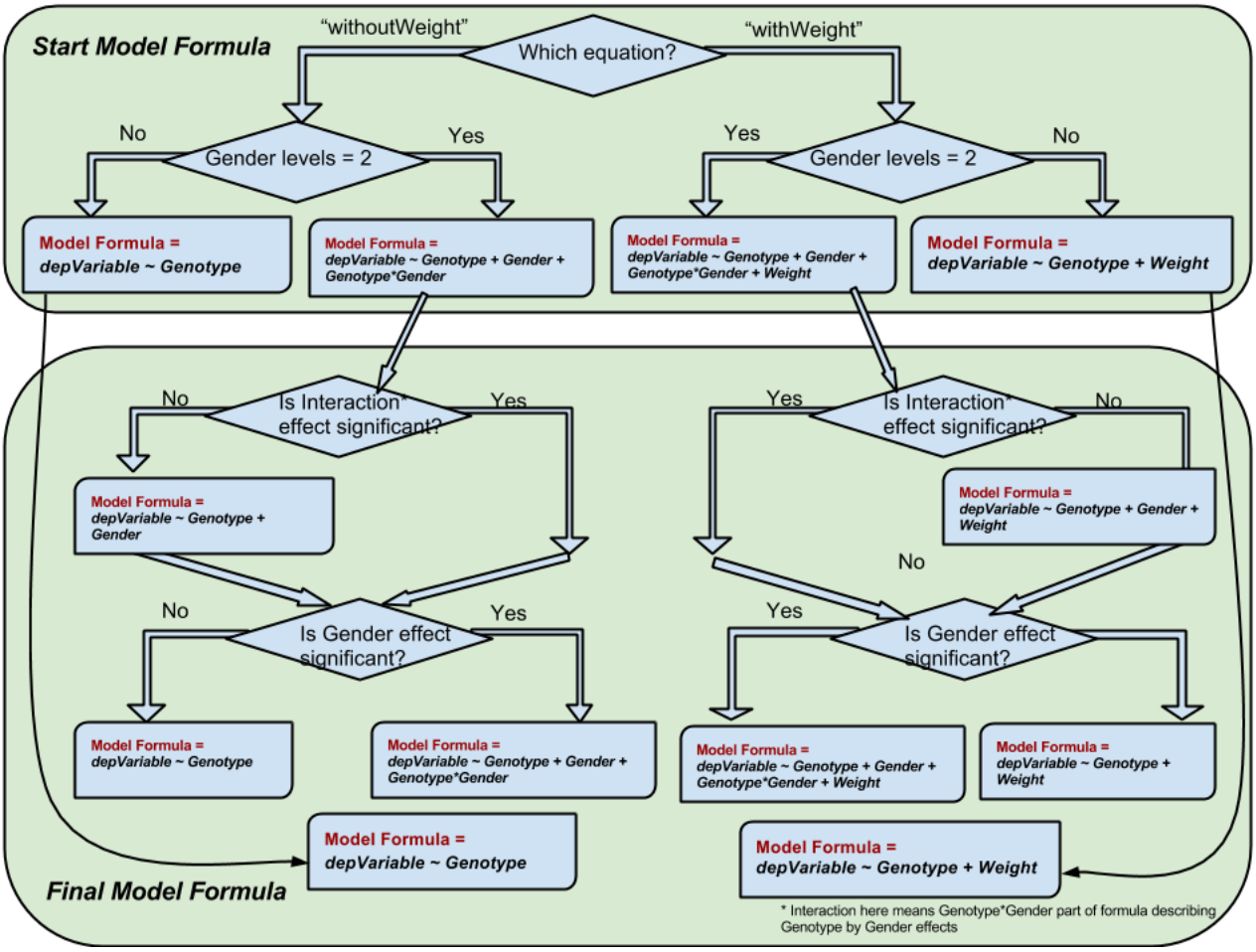

Figure 3: MM framework: start model formula and final model formula creation based on the dataset and significances of the effects (can be estimated or defined by user).

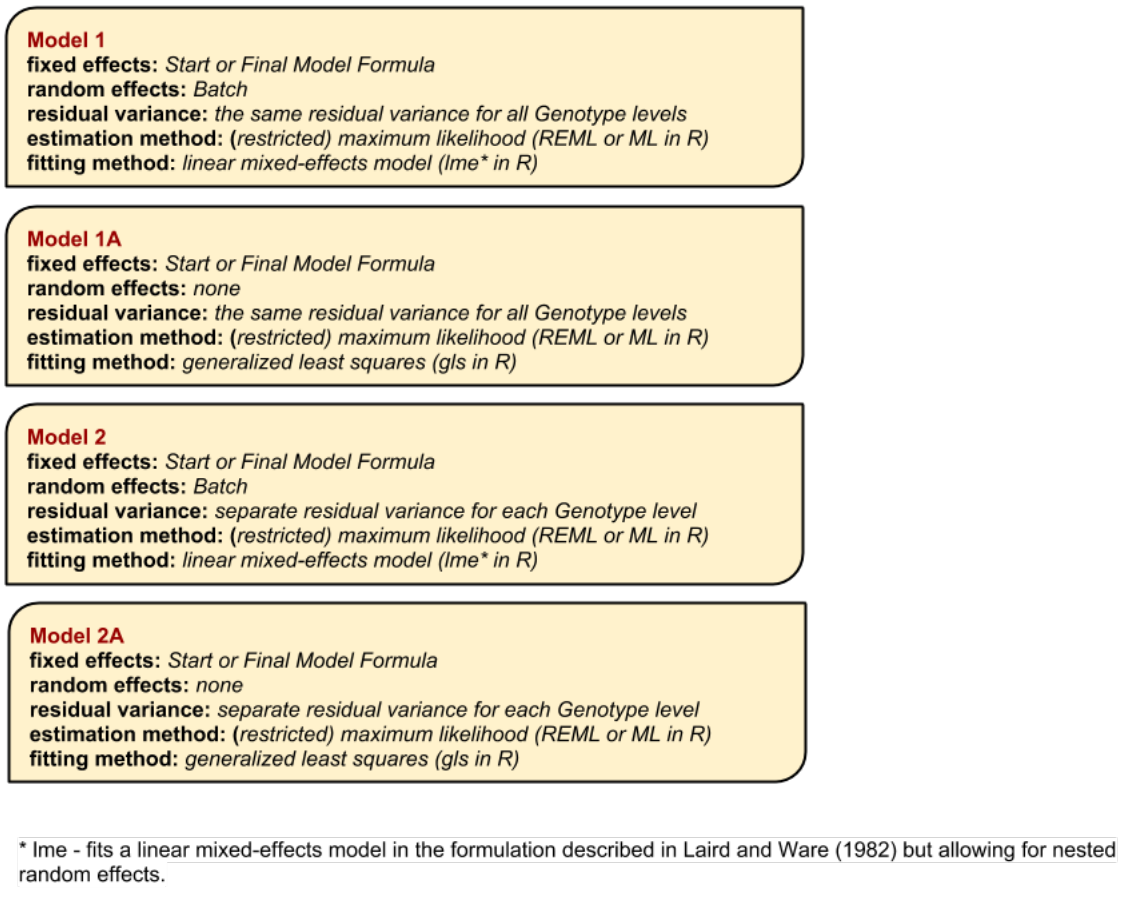

Figure 4: MM framework: different models that are considered.

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 2 Step 5,6 and Figure 3).

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 sequentially, 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 Eq1 and Eq2). 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 criteria or effect (body weight effect, residual variances homogeneity, gender effect, genotype by gender interaction effect, batch effect) is evaluated individually 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. It is important to mention here the convergency problem. If for some reason selected model is failing to converge we simplify it by selecting the similar but simplier model and try to fit again. For instance, if model with heterogeneous residuals is not converging then model with homogeneous residuals will be selected.

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 the user to change some of the arguments of functions and start the model building process from scratch if needed.

We believe that the possibility to change mixed models framework behaviour as described above will help users to go deeper into details of the modelling process, as well as debug 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" residuals (Houseman *et al.* (2004)) normality test (last two only 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, otherwise set to NA.
- "Rotated" residuals normality test result (p-value); applicable only when there is batch random effects in the model, otherwise set to NA.

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.

Both BLUPs and "rotated" residuals tests are used for the model fit estimation.

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 also calculate 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 4 for more details.

If there is only one level for the dependent variable in the dataset e.g. "Normal" then the package will add level "Other" into the count matrices for

consistency. All values for this level will be set to 0. The following is an example of such case:

```
> test2 <- PhenList(dataset=read.csv("test_categorical_normal.csv"),
> result2 <- testDataset(test2,depVariable="Thoracic.Processes", method="FE")
> levels(result2$model.dataset$Thoracic.Processes)
[1] "Normal"
> summaryOutput(result2)

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

Model output:
All data p-val: 1
All data effect size: 0%
Males only p-val: 1
Males only effect size: 0%
Females only p-val: 1
Females only effect size: 0%

Matrix 'all':
       +/+ Aff3/Aff3
Normal 895        13
Other    0         0

Percentage matrix 'all' statistics:
       +/+ Aff3/Aff3 ES change
Normal 100       100         0
Other    0         0         0

...
```

3.3.1 Classification Tag

We've implemented the function *classificationTag* also for the FE framework. However, in the case of Fisher Exact Test it is not sexual dimorphism classification, but rather the overall estimation of the signals significance.

In the Table 4 classification tags and observed p-values classification is based on are presented.

Males only	Females only	All	Classification Tag
< 0.05	< 0.05	< 0.05	Significant in males, females and in combined dataset
< 0.05	≥ 0.05	< 0.05	Significant in males and in combined dataset
< 0.05	< 0.05	≥ 0.05	Significant in males and in females datasets
< 0.05	≥ 0.05	≥ 0.05	Significant in males dataset only
≥ 0.05	< 0.05	< 0.05	Significant in females and in combined dataset
≥ 0.05	< 0.05	≥ 0.05	Significant in females dataset only
≥ 0.05	≥ 0.05	< 0.05	Significant in combined dataset only
≥ 0.05	≥ 0.05	≥ 0.05	Not significant

Table 4: p-values from Fisher Exact Tests and classification tag

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
```

The summary output for MM framework contains metrics indicating the quality of the fit:

- **Value** stands for the estimated coefficient that describes the value of slope calculated by the regression. It might seem a little confusing that the Intercept also has a value, but just think of it as a slope that is always multiplied by 1. This number will obviously vary based on the magnitude of the variable your are inputting into the regression, but it's always good to spot check this number to make sure it seems reasonable.
- **Std.Error** is a standard error of the coefficient estimate – measure of the variability in the estimate for the coefficient. Lower means better but this number is relative to the value for the coefficient.
- **DF** stands for the "Degrees of Freedom" which is the difference between the number of observations included in training sample and the number of variables used in model (intercept counts as a variable).
- **t-value** of the coefficient estimate is a score that measures whether or not the coefficient for this variable is meaningful for the model. It is used to calculate the p-value.
- **p-value** is variable p-value that represents the probability the variable is NOT relevant. The smaller is number the more important is variable (model part). If the number is really small, R will display it in scientific notation.

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" many 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 3.3) 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                          NA
      Value g2_l10              Male Value g1_l1
                NA                        "61"
  Male Value g2_l1              Male Value g1_l2
               "5"                       "392"
  Male Value g2_l2              Male Value g1_l3
               "1"                          NA
  Male Value g2_l3              Male Value g1_l4
                NA                          NA
  Male Value g2_l4              Male Value g1_l5
                NA                          NA
  Male Value g2_l5              Male Value g1_l6
                NA                          NA
  Male Value g2_l6              Male Value g1_l7
                NA                          NA
  Male Value g2_l7              Male Value g1_l8
                NA                          NA
  Male Value g2_l8              Male Value g1_l9
                NA                          NA
  Male Value g2_l9             Male Value g1_l10
                NA                          NA
 Male Value g2_l10            Female Value g1_l1
                NA                        "83"
Female Value g2_l1            Female Value g1_l2
               "7"                       "363"
Female Value g2_l2            Female Value g1_l3
               "0"                          NA
Female Value g2_l3            Female Value g1_l4
                NA                          NA
Female Value g2_l4            Female Value g1_l5
                NA                          NA
Female Value g2_l5            Female Value g1_l6
                NA                          NA
Female Value g2_l6            Female Value g1_l7
                NA                          NA
Female Value g2_l7            Female Value g1_l8
                NA                          NA
Female Value g2_l8            Female Value g1_l9
                NA                          NA
Female Value g2_l9           Female Value g1_l10
                NA                          NA
Female Value g2_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. This graph allows a visual representation of the genotype effect for the variable of interest.

5.2 Graphics for Continuous Data

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

- *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.

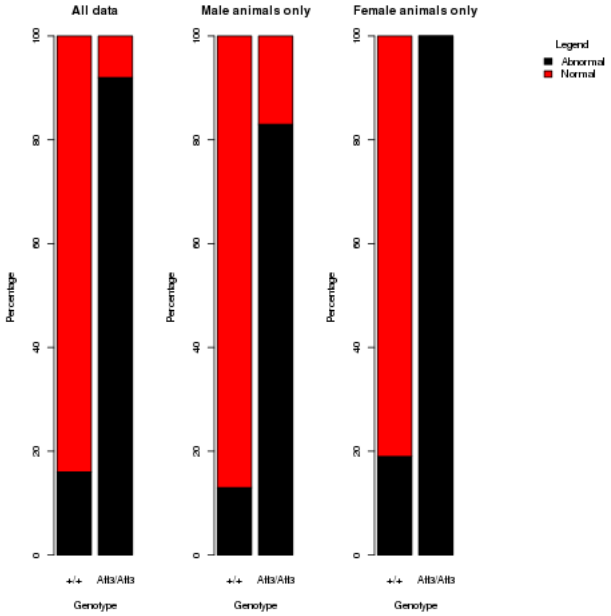

Figure 5: The PhenStat package's graphical output: categorical bar plot.

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

The example of box plot split by gender and genotype is shown in Figure 6. Outliers are shown as independent data points beyond the fences ("whiskers") of the boxplot. An outlier is defined as a data point that is 1.5 times the interquartile range above the upper quartile and bellow the lower quartile.

The example of box plot split by gender, genotype and batch is shown in Figure 7. This allows a visualisation of variation of dependent variable with time. The MM framework assumes this variation is random and conforms the normal distribution. Then the genotype distribution can be of relative to natural variation.

The example of scatter plot of body weight versus dependent variable is shown in Figure 8. When weight is included in the model MM framework assumes a linear relationship between dependent variable and body weight. This graph allows an assessment of this showing both regression line and a loess line (locally weighted line) fitted for each genotype.

There are five functions in the results based graphs category:

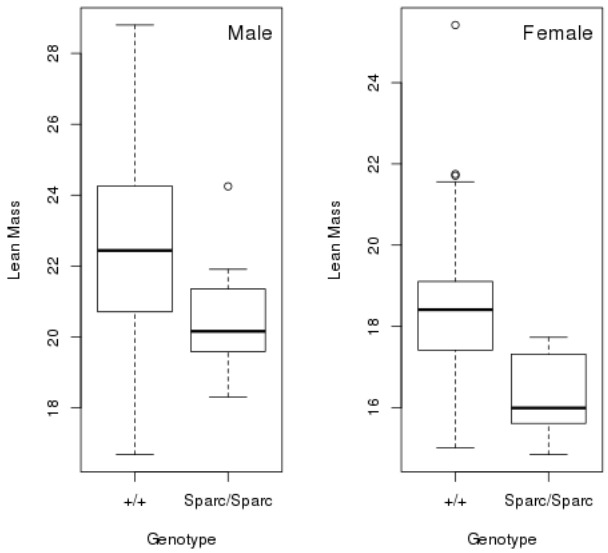

Figure 6: The PhenStat package's graphical output: box plot split by gender and genotype.

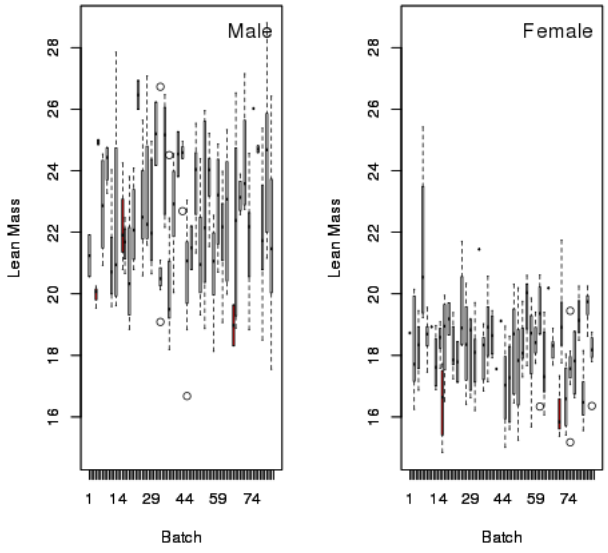

Figure 7: The PhenStat package's graphical output: box plot split by gender, genotype and batch.

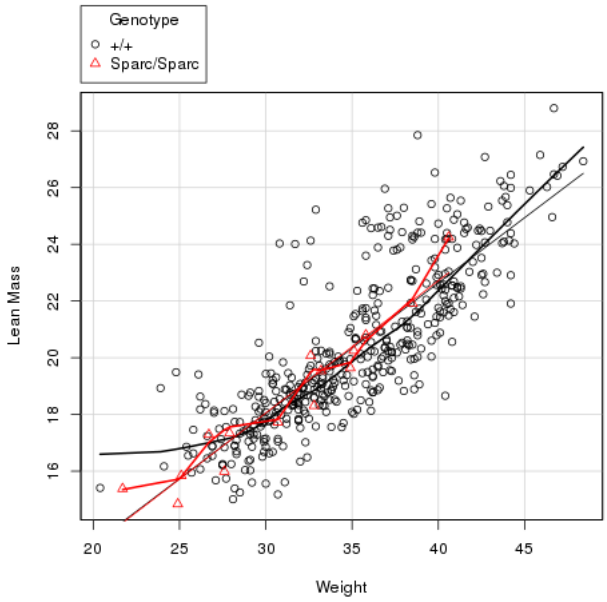

Figure 8: The PhenStat package's graphical output: scatter plot of body weight versus dependent variable.

- *qqplotGenotype* creates a Q-Q plot of residuals for each genotype.
- *qqplotRandomEffects* creates a Q-Q plot of blups (best linear unbiased predictions).
- *qqplotRotatedResiduals* creates a Q-Q plot of "rotated" residuals.
- *plotResidualPredicted* creates predicted versus residual values plots split by genotype.
- *boxplotResidualBatch* creates a box plot with residue versus batch split by genotype.

```
> 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 9. The MM framework assumes residuals are normally distributed. Residuals are the differences between the real values observed for a dependent variable and the fitted values from the model. Q-Q plot assesses this assumption (residuals will be randomly arranged around the line if normally distributed).

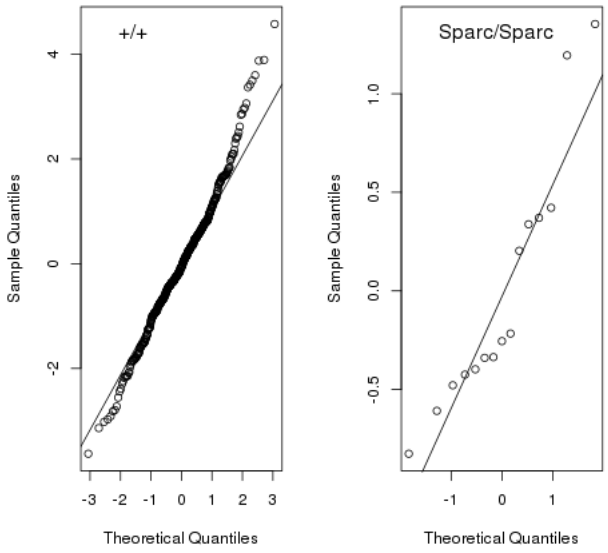

Figure 9: The PhenStat package's graphical output: Q-Q plot of residuals for each genotype.

The example of Q-Q plot of BLUPs (best linear unbiased predictions) is shown in Figure 10. The MM framework assumes the BLUPs are normally distributed. This graph assesses for this assumption by plotting BLUPs and the ideal normal line (large deviations from the line can be an indicator of problems with the model fit). BLUPs are best linear unbiased predictions and are used for the estimation of batch effects. See tutorial BLUPs for more details.

Another method to assess the model fit is to consider the normality of the "rotated" and "unrotated" residuals. The example of Q-Q plot of "rotated" residuals is shown in Figure 11. See section 3.2.3 for the details about "rotated" residuals.

The example of residual-by-predicted plot is shown in Figure 12. 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.

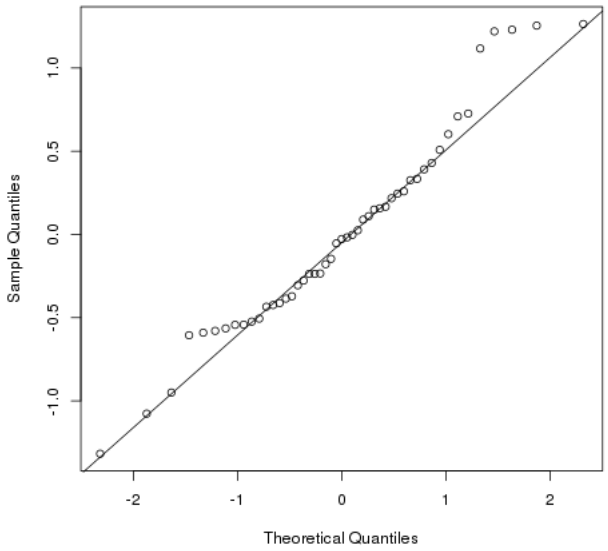

Figure 10: The PhenStat package's graphical output: Q-Q plot of BLUPs (best linear unbiased predictions).

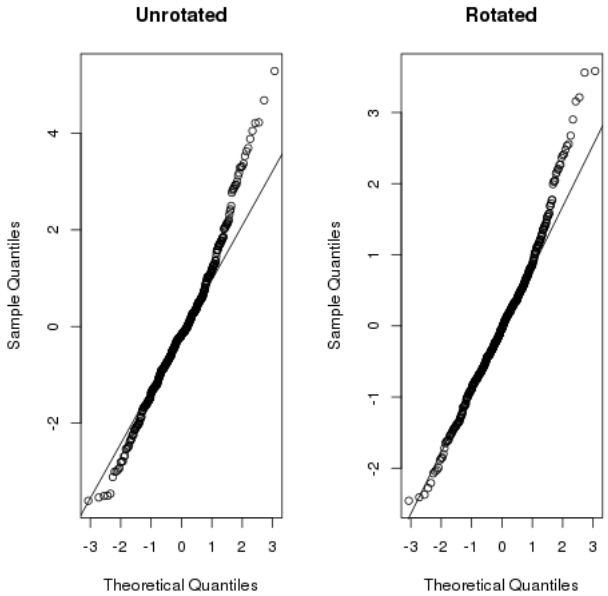

Figure 11: The PhenStat package's graphical output: Q-Q plot of "rotated" residuals.

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.

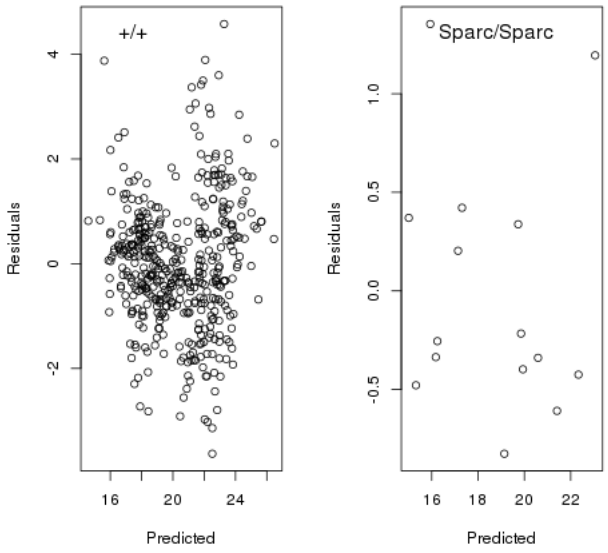

Figure 12: The PhenStat package's graphical output: residual-by-predicted plot split by genotype.

The example of box plot with residue versus batch split by genotype is shown in Figure 13. 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.

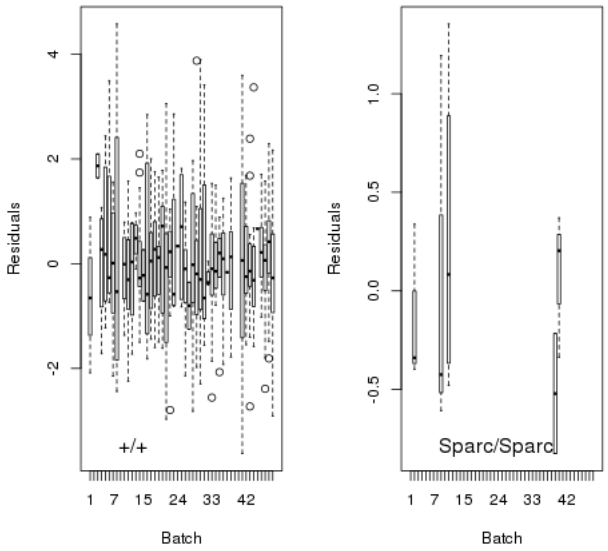

Figure 13: The PhenStat package's graphical output: box plot with residue versus batch split by genotype.

6 Case Studies

6.1 PhenStat Usage Example Continuous Data

The following dataset, provided by Wellcome Trust Sanger Institute (WTSI) Mouse Genetics Project (MPG) is Dual-energy X-ray absorptiometry data obtained for a study on gene knockout mice carrying the Akt2<tm1Wcs> targeted allele which were created by blastocyst injection of targeted ES cells, and bred on the 129S5SvEvBrd genetic background. Data was collected on a high throughput pipeline following a multi-batch workflow, where regular control animals are collected and when knockout animals are the correct age they are issued to the pipeline as they arise.

6.1.1 Loading the data and initial steps of analysis

Initial steps focus on loading the data, using the PhenStat tools to generate the PhenList object, and then the result object. We can then explore the data and fitted results using the visualisation and output functions.

```
> DEXAdata=read.csv("WTSI_AKt2_data.csv")
```

Genotype	Gender	Number Animals	Number Batches
Homozygote	Female	12	3
	Male	14	3
Wild type	Female	574	96
	Male	572	97

Table 5: Number of animals and number of batches in the Akt2 dataset

```
> test <- PhenList(dataset=DEXAdata,testGenotype="Akt2/Akt2",
  refGenotype="WT",
  dataset.colname.batch="dateOfExperiment",
  dataset.values.female="female",
  dataset.values.male="male" )
> result <- testDataset(test,depVariable="Lean.Mass",method="MM",equation="withoutWeight")
```

6.1.2 Exploring and understanding the output

The first two lines of the *summaryOutput* confirm the statistical framework used and the dependent variable studied. The next section of the output clarifies the final fitted model details. As the MM framework is an optimisation process exploring the data to fit the best model to the data, the final fitted model details will vary. We can see that batch variation, the variation in readings between different assay dates, were found to be statistical significant and hence a mixed model will have been fitted where batch is treated as a random effect. If it was not significant, then the model would have reverted to a simpler linear model. The next line of output, "was variance equal?", indicating whether the model assumes equal variance between genotype groups or unequal. In this case, the variance was not found to be equal and therefore the final model estimated the variance for each group separately.

The next stage of the output, indicates whether there was evidence of sexual dimorphism (i.e. the genotype effect was found to be dependent on sexes). In this case, the statistical test of sexual dimorphism was non-significant (p-value = 0.899) hence the final model would not assess the genotypes for each sex separately. The following line indicate which main effects were included in the final model; for this variable and dataset we can see that gender was significant in explaining variation in the data and hence is included in the final model.

The final stage of the output gives information on the final model details. The "Genotype effect" reports the statistical significance for the genotype

effect and is assessed by comparing a treatment model (final fitted model) with a null model where a null model has no genotype effects in the model but all other significant main effects. In this example the null model would thus be Lean.Mass $\sim$ Gender. Looking at the output the "Genotype effect" is highly statistical significant as a 0 value is reported. We can then look at the model fitting details in the final output of the function by examining the table to see how the main effects contributed to the variation in the dependent variables.

```
> 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 0.906)
Final fitted model: Lean.Mass ~ Genotype + Gender
Model output:
Genotype effect: 0
Classification tag: With phenotype threshold value 0.01 - both sexes equally
                     Value Std.Error   DF   t-value       p-value
(Intercept)       18.525264 0.1936360 1048 95.670550  0.000000e+00
GenotypeAkt2/Akt2 -3.527111 0.4866433 1048 -7.247836  8.204401e-13
GenderMale         4.620361 0.1581236 1048 29.219943 8.940893e-138
```

A regression model estimates each component of the model, by isolating how that effect influences the dependent variable (i.e. lean mass) as though all other parts of the model had been fixed and isolated. From the table of output, we can see that the intercept (the expected reading for lean mass for female reference genotype (aka female wild type animals) is 18.5. Then we can see the effect of the genotype is -3.5 in that knockout animals are predicted to have 3.5 grams less lean mass relative to the control animals. Finally we can see that being a male animals leads to an increase of 4.6 grams. As there was no evidence for sexual dimorphism, and thus the genotype effect was estimated independently of gender, then a classification tag "both sexes equally" has been assigned. These estimates are in agreement with the visualisation of the data (see section 6.1.3).

6.1.3 Assessment of raw data and distribution characteristics

To assess model fit, graphical tools are ideal. They focus on two areas:

1. Assessment of raw data and distribution characteristics
2. Assessment of model fit

The function, *boxplotGenderGenotype*, allows the genotype effect to be visualised for each gender group. For the Akt2 example, we can conclude that there looks to be a significant genotype effect that is seen in both genders.

```
> boxplotGenderGenotype(result, "Lean.Mass", "Lean Mass (g)")
```

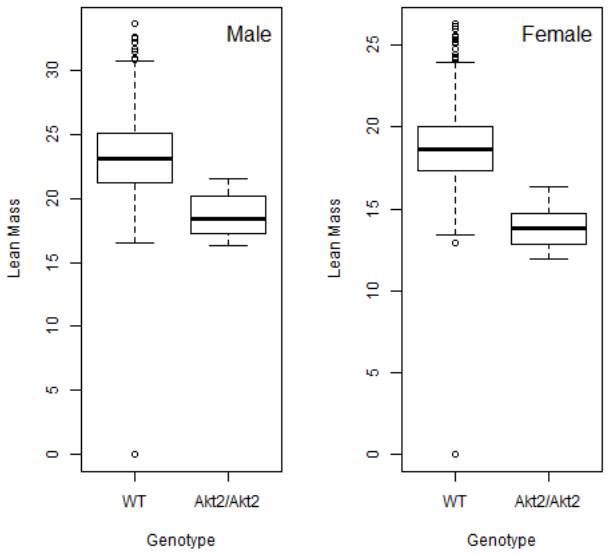

Figure 14: *boxplotGenderGenotype* function for Akt2 example.

A second function, *boxplotGenderGenotypeBatch*, allows the comparison between genotype as a function of batch and gender. This plot allows the user to visualise the batch variation and assess how the treatment measures look relative to the batch variation. It is important to note that as dates can be entered in many forms, the batches are not ordered with time. For the Akt2 lean mass example, we can see that there is significant batch variation, which explains why a mixed model was fitted.

```
> boxplotGenderGenotypeBatch(test, "Lean.Mass", "Lean Mass (g)")
```

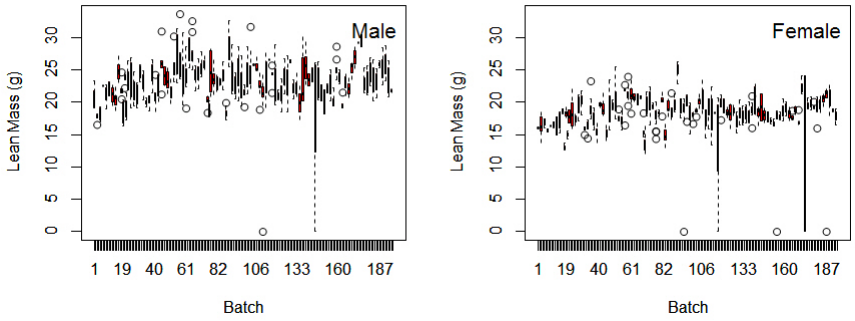

Figure 15: *boxplotGenderGenotypeBatch* function for Akt2 example.

6.1.4 Assessment of model fit

Five functions are available and they focus on looking at residual behaviour. A residual, is the difference between the estimated dependent variable from the final model estimates and the actual measured dependent variable response. A model is a good fit, when the residuals are normally distributed and there is no systematic pattern in the distribution of the residuals relative to the dependent variable.

The *vectorOutput* function includes statistical tests for normality on the residuals for the wild type, residuals for the knockout, the blups and "rotated" residuals (see section 3.2.3). These normality tests are provide to assist in the building automated tools for assessing model fit, however when there is a lot of data (e.g. in a dataset where the wild type arises from a high throughput program with a running baseline), the statistical test can be overall sensitive to departures from normality and when the number of data points is low (e.g. in many knockout groups), the test can lack ability to detect deviations from normality.

- **qqplotGenotype**

  This function assesses the normality of the residuals are assessed for each genotype through plotting a normal Q-Q plot. Q-Q plots are a means of comparing two distributions. To test normality, we plot the residuals against a normal distribution and see if they match. If the two distributions are similar the points on a QQ plot will fall along the y=x line (unity). Thus we are looking for a random distribution of points along the lines.

Looking at the Akt2 lean mass example, the residuals on the HOM group are near perfect showing the model is fitting this data well. The residuals on the WT group are deviating in a way (systematic below at one end and systematic above at the other) which indicates that we have long tails to our distribution. This is not concerning as we have a very large control dataset and we do have outliers in the data.

```
> qqplotGenotype(result)
```

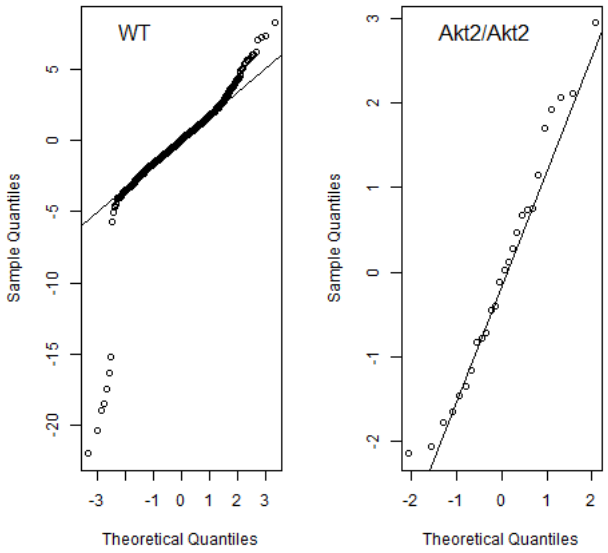

Figure 16: *qqplotGenotype* function for Akt2 example.

- **boxplotResidualBatch**

  This function allows visualisation to assist the user to assess whether the deviation in the residual is consistent across all the batches and similar in size between the wild type and knockout line. For the Akt2 example, we can see that the variation in residual is consistent across all the batches and similar in size between the knockout and wild type group.

```
> boxplotResidualBatch(result)
```

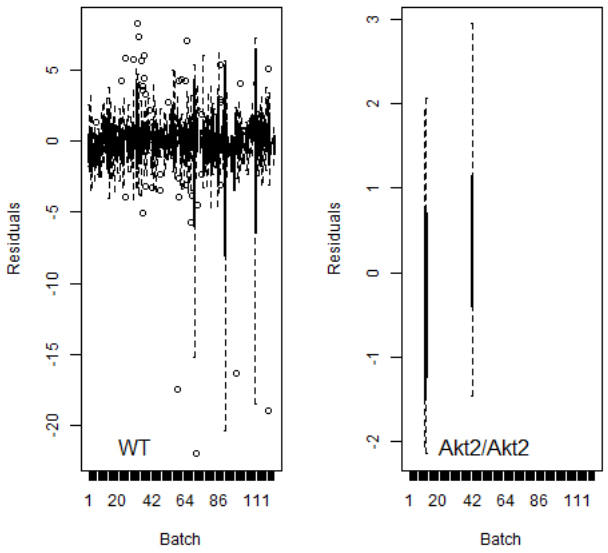

Figure 17: *boxplotResidualBatch* function for Akt2 example.

- **plotResidualPredicted**

  This function plots the residuals against the predicted readings for both genotypes. The predicted readings are the values the model would estimate for the dependent variable. As a user, you are looking to see that the model is fitting the data well over the entire data range. Looking at the akt2 data, we can see that there spread of the residuals is fairly consistent, however there are some data points that are not being fit well by the model, the good news is that they are in the control set but they should be considered further to see if a reason for their poor fit can be ascertained.

```
> plotResidualPredicted(result)
```

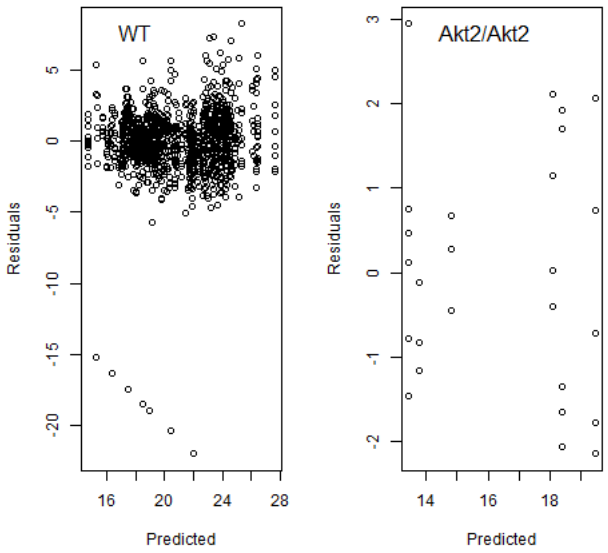

Figure 18: *plotResidualPredicted* function for Akt2 example.

- **qqplotRandomEffects**

  This function is assessing the assumption that the batch effects are normally distributed. The estimates of the random effects, aka the estimates of the batch effects in this scenario, are called best linear unbiased prediction BLUPs. Here a normal Q-Q plot is used to plot the estimated BLUPs against a normal distribution. So looking at the Akt2 lean mass example, the majority of the data points are distrbuted along the line. There is some systematic deviation at the tails but it is a small percentage of the points and as it is above and below the line it indicates long tails (ie outliers) and so we can conclude the distribution is not too far from the ideal and the model is a good representation of the data.

```
> qqplotRandomEffects(result)
```

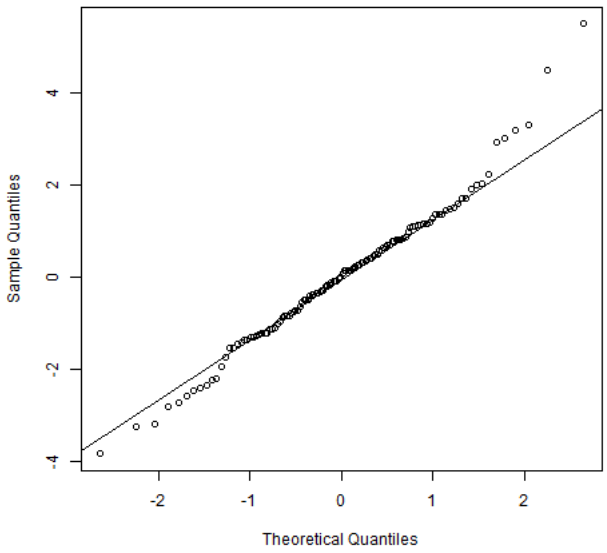

Figure 19: *qqplotRandomEffects* function for Akt2 example.

- **qqplotRotatedResiduals**

  This function, allows the user to consider the normality of the "rotated" and "unrotated" residuals and have been recommended to assess model fit success with mixed models (Houseman *et al.* (2004)). See section 3.2.3 for more details. So looking at the Akt2 lean mass example, the majority of the data points are distributed along the line. There is some systematic deviation at the tails but it is a small percentage of the points and as it is above and below the line it indicates long tails (i.e. outliers) and so we can conclude the distribution is not too far from the ideal and the model is a good representation of the data.

```
> qqplotRotatedResiduals(result)
```

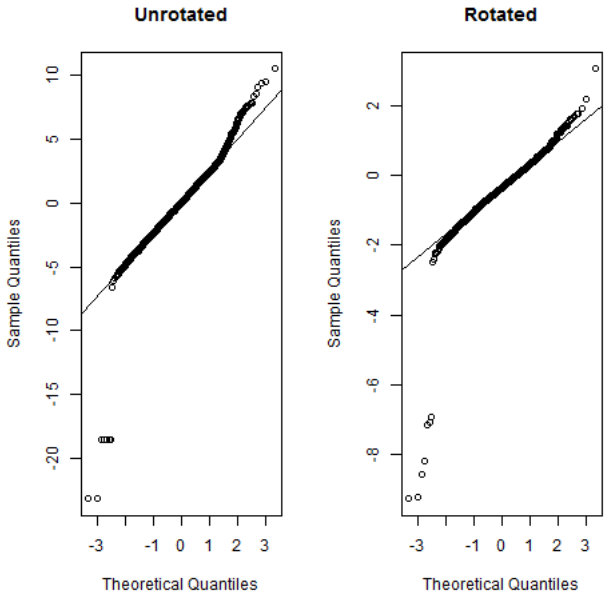

Figure 20: *qqplotRotatedResiduals* function for Akt2 example.

6.1.5 Including weight as a covariate in the model fitting process

What is the value to including body weight in the model?

Weight is included in the initial model via the *testDataset* function equation argument being set to either "withWeight" or "withoutWeight". When weight is included as a covariate, the model is assuming that the dependent variable (e.g. lean mass) has a linear relationship with body weight. If weight is not found to be statistical significant in explaining the variation in the dependent variable, then weight as a covariate will drop out of the final model and the equation will automatically revert to an equation type "withoutWeight".

There are two advantages to including weight:

1. **Increase in sensitivity.** If differences in animal weight lead to greater variability in the dependent variable, then by adding weight and accounting for this variability then the statistical model will be more sensitive to a genotype effect.

2. **Adjusting for weight differences between the knockout and**

control group. When there is a weight difference between the knockout and wild type animals, the genotype effect is confounded by the weight effect in that there is a difference in the dependent variable but you cannot assess whether it is due to the differences in body weight of the knockout and control animals or genotype differences between the knockout and control animals. When weight is included in the equation, the genotype effect is then testing for a genotype difference after adjusting for the weight difference.

We have found that the majority of continuous phenotypic variables monitored in the WTSI Mouse Genetics Project (MPG) have a relationship with body weight. Looking at the Akt2 dataset we can see that there is a difference in body weight between the wild type and knockout group and thus body weight can be a confounding factor to isolating the genotype effect.

```
> boxplotGenderGenotype(test, "Weight", "Body Weight (g)")
```

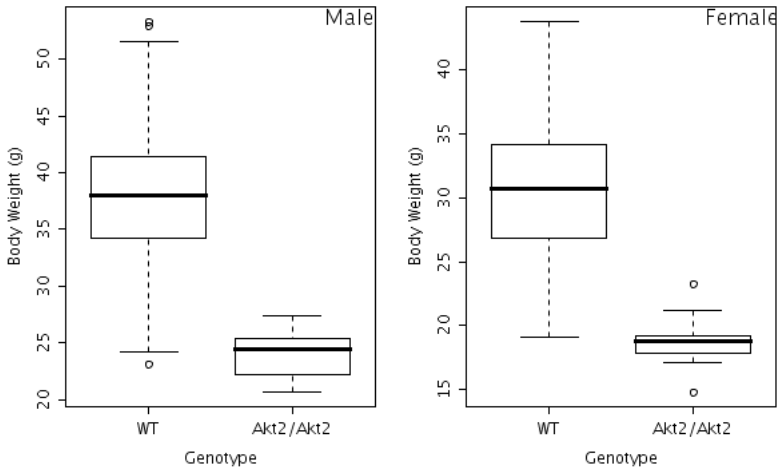

Figure 21: *boxplotGenderGenotype* function for Akt2 example to show the body weight impact.

What is the impact of including weight on the *summaryOutput*?

When body weight is included, the inclusion can be seen in the final fitted model as weight is listed as a covariate and then in the final model output table, where the influence of weight on the fitted model is shown. With weight included the genotype effect is estimating the impact of genotype after adjusting for weight differences in the animals.

Looking at the *summaryOutput* for the Akt2 example, the table shows that for each gram of body weight the lean mass increased by 0.32g. When weight is included in the equation it changes the definition of the intercept from the lean mass value of a wild type female animal to the lean mass value of a wild type female animal of zero body weight. This happens as the model is estimating each of these terms influences in isolation of the other terms. In contrast to the earlier fitted results (section 6.1.2), when weight is included in the model, the classification tag identifies the change as "no significant change" as the global genotype test is now not significant with a p-value of 0.64. This means the statistically difference observed with the fitted model "withoutWeight" was entirely due to a body weight differences between the knockout and control animals. So whilst there is a fundamental differences in lean mass between the knockout and control this is due to the knockout animals being smaller than the control animals.

```
> 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 0.105)
Final fitted model: Lean.Mass ~ Genotype + Gender + Weight
Model output:
Genotype effect: 0.639989841
Classification tag: With phenotype threshold value 0.01 - no significant change
                      Value  Std.Error   DF    t-value      p-value
(Intercept)        7.6201046 0.52150631 1047 14.6117207 3.827390e-44
GenotypeAkt2/Akt2 -0.1924952 0.41201772 1047 -0.4672012 6.404531e-01
GenderMale         2.1550868 0.17386173 1047 12.3954067 5.056608e-33
Weight             0.3541522 0.01623164 1047 21.8186332 2.718321e-87
```

6.1.6 Additional model diagnostics when weight is included

In addition to the diagnostic discussed in previous section, when weight is included in the model, it is important to consider whether the linear relationship between body weight and the dependent variable is a valid assumption. This can be assessed with the *scatterplotGenotypeWeight* function. In this plot, for each genotype a regression line is fitted to assess the relationship between the dependent variable and body weight. Then a locally weight line (loess line) is plotted. The loess line allows assessment that the regression

line fits all the data well. Note the loess line can be distorted by a few data points so if it deviates strongly but for only a few data points, this is not concerning. This graph is used to assess whether a linear relationship exists and whether it is the same for both genotypes.

In the Akt2 example, it can be clearly seen that a common linear relationship exists between lean mass and body weight, such that as the weight increases so does the lean mass. It can also be seen that the knockout animals have a lower body weight and subsequently lower lean mass but the drop in lean mass is entirely in accordance with the drop in body weight.

```
> scatterplotGenotypeWeight(test, depVariable="Lean.Mass")
```

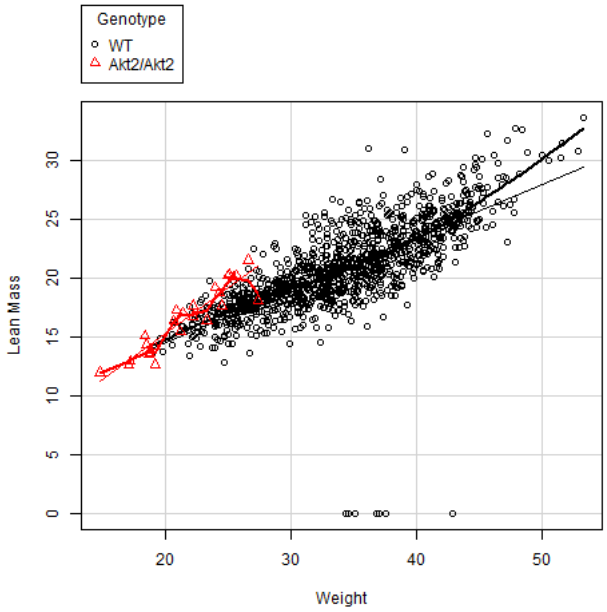

Figure 22: *scatterplotGenotypeWeight* function for Akt2 example.

6.2 PhenStat Usage Example Categorical Data

The following dataset, provided by Wellcome Trust Sanger Institute (WTSI) Mouse Genetics Project (MPG) of high resolution X-ray data obtained from a study on gene knockout mice carrying the Aff3tm1a(EUCOMM)Wtsi targeted allele which were created by blastocyst injection of targeted ES cells, and bred on the B6N genetic background. Data was collected on a high throughput pipeline following a multi-batch workflow, where regular control

animals are collected and when knockout animals are the correct age they are issued to the pipeline as they arise. At WTSI, batch to batch variation has not been found to be significant for these rare event categorical variables. Consequently, we ignore batch and combine data for the same genetic background when collected with the same protocol and housing and husbandry conditions. This increases the sensitivity of the analysis as we have more accuracy on the estimate of the prevalence of the condition in the wild type population.

Genotype	Gender	Number Animals	Number Batches
Homozygote	Female	7	4
	Male	6	4
Wild type	Female	446	70
	Male	451	70

Table 6: Number of animals and number of batches in the Aff3 dataset

6.2.1 Loading the data and initial steps of analysis

Initial steps focus on loading the data, using the PhenStat tools to generate the PhenList object, and then the result object. We can then explore the data and fitted results using the visualisation and output functions.

```
> aff3data=read.csv("categorical_example 1_aff3_Xray.csv")
> test <- PhenList(dataset=aff3data,testGenotype="Aff3/Aff3",
  refGenotype="+/+", dataset.colname.batch="Assay.Date")
> result <- testDataset(test, depVariable="Thoracic.Processes", method="FE")
```

6.2.2 Visualisation of data

The function *categoricalBarplot* has been provided to visualise the categorical data as summary percentage data. It reports the percentage of each classification observed for up to three datasets: all data, male only and female only. It is important to note that percentage accuracy is very dependent on the number of readings so it is important to consider the dataset size when interpreting these graphs. Therefore tables showing both the percentage and count values are included in the *summaryOutput*.

```
>  categoricalBarplot(result)
```

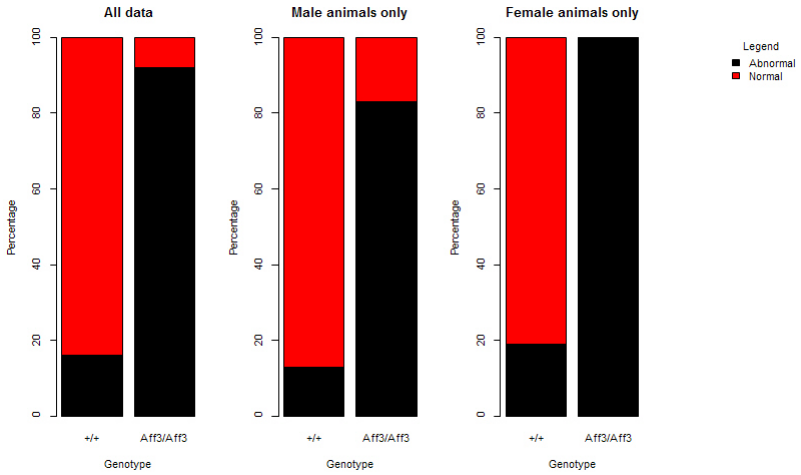

Figure 23: *categoricalBarplot* function for Aff3 example.

6.2.3 Understanding the *summaryOutput*

The first two lines of the *summaryOutput* confirm the statistical framework used and the dependent variable studied. The next section of the output reports the summary of statistical assessment. Two measures are provided for each dataset considered. First a test of statistical significance assessed using a Fisher Exact Test and then a measure of biological significance, the maximum effect size. The final section of the output provides tables showing the counts and percentage calculated for each group and possible level.

The following is reported for dependent variable thoracic processes for the Aff3 dataset and we can see that for all datasets, there is a statistically significant change with a large effect size.

The *summaryOutput* function shows the model fitted and the model output for the analysis of the thoracic processes for the aff3 dataset. The example is showing the summary tables provided for the composite (male and female data) for the thoracic processes for the aff3 dataset.

```
>  summaryOutput(result)

...
Model output:
All data p-val: 4.35745946092922e-09
All data effect size: 76%
Males only p-val: 0.00025633944344021
Males only effect size: 70%
```

```
Females only p-val: 1.00779809539594e-05
Females only effect size: 81%

Matrix 'all':
         +/+ Aff3/Aff3
Abnormal 142        12
Normal   753         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 36.678  1 1.3931e-09
Pearson          53.164  1 3.0675e-13

Phi-Coefficient   : 0.242
Contingency Coeff.: 0.235
Cramer's V        : 0.242
...
```

6.2.4 Understanding the maximum effect size reported

The maximum effect size is the maximum percentage change seen for an observation type. Below is a table showing an artificial example where the majority of the wild type are normal but the abnormality is spread across multiple levels in the knockout. For each trait level (i.e. the observed phenotype), the change in percentage effect size is seen by subtracting the percentage observed in the knockout from the wild type. Then across all the observed levels, the maximum percentage change is selected after ignoring the direction of the change. Thus in the example below, the maximum effect size would be 86.5% which indicates that there has been 86.5% change in how often a level is observed.

6.2.5 A dependent variable with little variation

Within the IMPC pipeline, there are a number of dependent variables which have little variation but are numeric. For example, no of digits, or number of caudal vertebrae etc. If you try to process these variables through a mixed

Trait level	wild type		knockout		Effect size calculation	Effect size
	count	%	count	%		
Normal	198	99	1	12.5	99-12.5	86.5
Abnormal left eye	0	0	0	0	0	0
Abnormal right eye	1	0.5	4	50	0.5-50	49.5
Abnormal both eye	1	0.5	3	37.5	0.5-37.5	37

Table 7: Number of animals and number of batches in the Aff3 dataset

model framework the analysis will stop and an error will report that there is insufficient variability in the data.

For our example dataset, the number of ribs is a dependent variable with little variation as shown by plotting the data with the *categoricalBarplot* function.

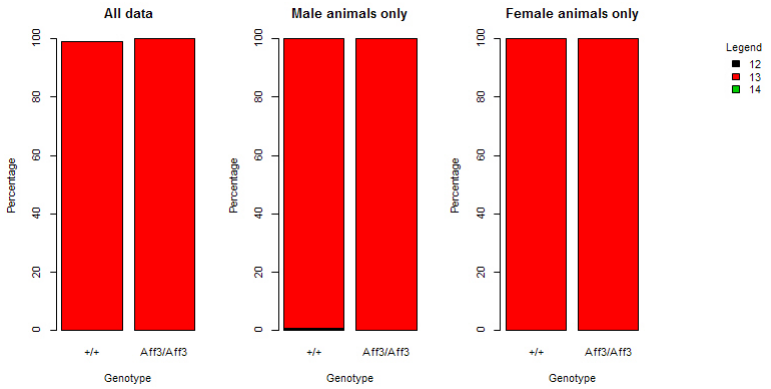

Figure 24: *categoricalBarplot* function for Aff3 example with number of ribs as dependent variable.

If we try and process this variable through a mixed model framework the following output is obtained:

```
> result<-testDataset(test,depVariable="Number.Of.Ribs.Left",method="MM")

Error:
Insufficient variability in the dependent variable 'Number.Of.Ribs.Left' for MM framework.
Fisher Exact Test can be better way to do the analysis.
Error:
Insufficient variability in the dependent variable 'Number.Of.Ribs.Left' for genotype/gender
combinations to allow the application of Mixed Model.
```

Instead the dependent variable should be treated as a categorical variable

which each numeric output possible treated as a level allowing a statistical comparison of how the levels are distributed between the knockout and wild type groups. Thus for our example the following summary output and count table is obtained:

```
> summaryOutput(result)

Test for dependent variable: Number.Of.Ribs.Left
Method: Fisher Exact Test framework

Model output:
All data p-val: 1
All data effect size: 0%
Males only p-val: 1
Males only effect size: 0%
Females only p-val: 1
Females only effect size: 0%

Matrix 'all':
   +/+ Aff3/Aff3
12   1         0
13 893        13
14   1         0

Percentage matrix 'all' statistics:
   +/+ Aff3/Aff3 ES change
12   0         0         0
13 100       100         0
14   0         0         0
```

6.3 PhenStat Integration with Database

6.4 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.

```
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 every individual variable of the dataset 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(), "./", 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.

E. Andrés Houseman, Louise M. Ryan and Brent A. Coull (2004) Cholesky Residuals for Assessing Normal Errors in a Linear Model with Correlated Outcomes, *Journal of the American Statistical Association*, **99** (466), 383-394