PhenStat:
statistical analysis of phenotypic data
User's Guide

Natalja Kurbatova, Natasha Karp, Jeremy Mason

Last revised: 16th September 2015

Contents

1   Introduction

High-throughput phenotyping generates large volumes of varied data including both categorical and continuous data. Operational and cost constraints can lead to a work-flow that precludes 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 with an emphasize on high-throughput data-flows. The package contains dataset checks and cleaning in preparation for the analysis, four statistical frameworks for the phenodeviants identification and additional functions that help to decide the correct method for analysis.

Simple explanation of statistical frameworks is given below. More details can be found in the appropriate sections of the User's Guide.

We also have implemented the IMPCdata package that allows easy access to the phenotyping data produced by the International Mouse Phenotyping Consortium (IMPC) and is available in Bioconductor.

1. Mixed Models framework assumes that base line values of dependent variable are normally distributed but batch (assay date) adds noise and models variables accordingly in order to separate the batch and the genotype. Model optimisation starting with:

$$depVariable = Genotype + Sex + Genotype * Sex + Weight + (1|Batch)$$

Assume batch is normally distributed with defined variance. This framework can be used in case when you have controls measured over multiple batches and you ideally have knockout mice measured in multiple batches. The knockouts do not have to be concurrent with controls.
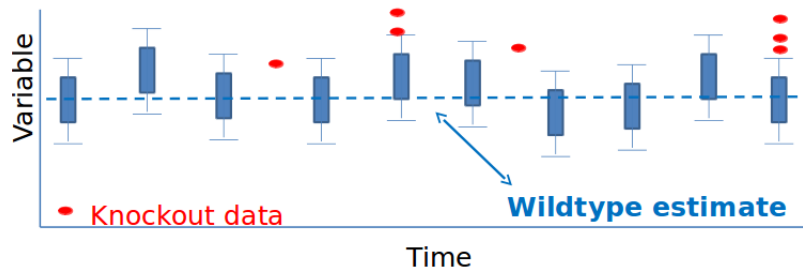
2. Time Fixed Effect framework estimates each batch effect to separate it from genotype. Model optimisation starting with:

$$depVariable = Genotype + Sex + Genotype * Sex + Weight + Batch$$

This framework can be used in case when there are up to 5 batches and concurrent controls approach had been used.

3. Reference Range Plus framework identifies the normal variation form the wildtype animals, classifies dependent variables from the genotype of interest as low, normal or high and compares proportions to assess for movement towards high or low class borders.
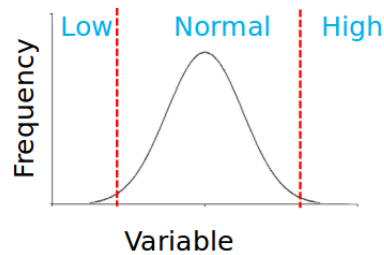
	Low	Normal/High
+/+	3	97
Knockout	6	1

p value: 1.59e-7
Effect Size: 82.7%

	Normal/Low	High
+/+	98	2
Knockout	7	0

p value: 1
Effect Size: 2%

This framework requires sufficient number of controls (more than 60 records) in order to correctly identify normal variation and can be used when other methods are not applicable or as a first simple data assessment method.

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

	Normal	Abnormal
+/+	753	142
Knockout	1	12

p-value: 4.35E-9
Effect Size: 76%
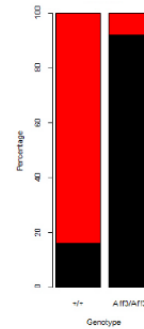
5. Logistic Regression is a framework for categorical data recoded to two levels: 0 (e.g. as expected or reference phenotype) and 1 (abnormal). This method models the relationship between the variable of interest and a number of independent variables assessing the impact of sex, genotype and whether the genotype effect depends on the sex.

Y = Genotype + Sex
Genotype p-value: 1.943e-09
Classification: both sexes equally
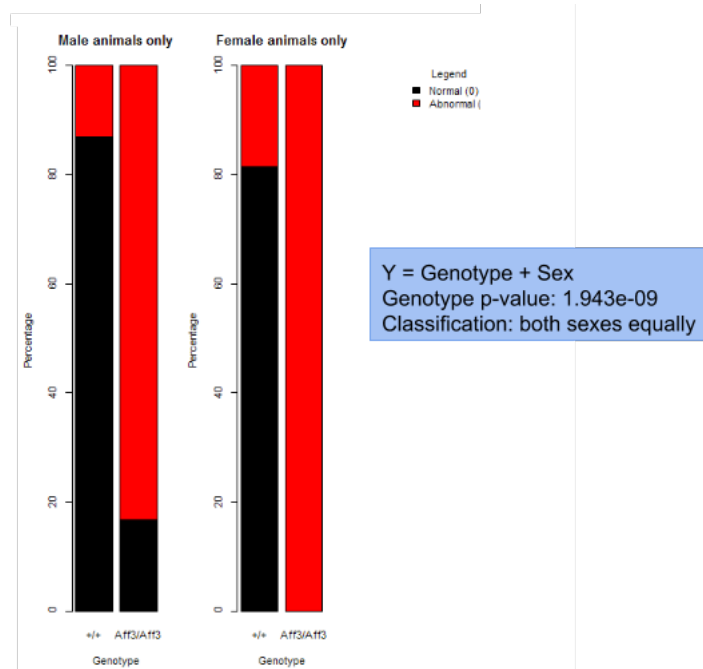
All analysis frameworks output a statistical significance measure, an effect

size measure, model diagnostics (when appropriate), and graphical visualisation of the genotype effect.

Depending on the user needs, the statistical analysis 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 from the Sanger MGP and Europhenome resources. Please note, the testing of the Time Fixed Effect framework has been limited due to the shortage of suitable datasets.
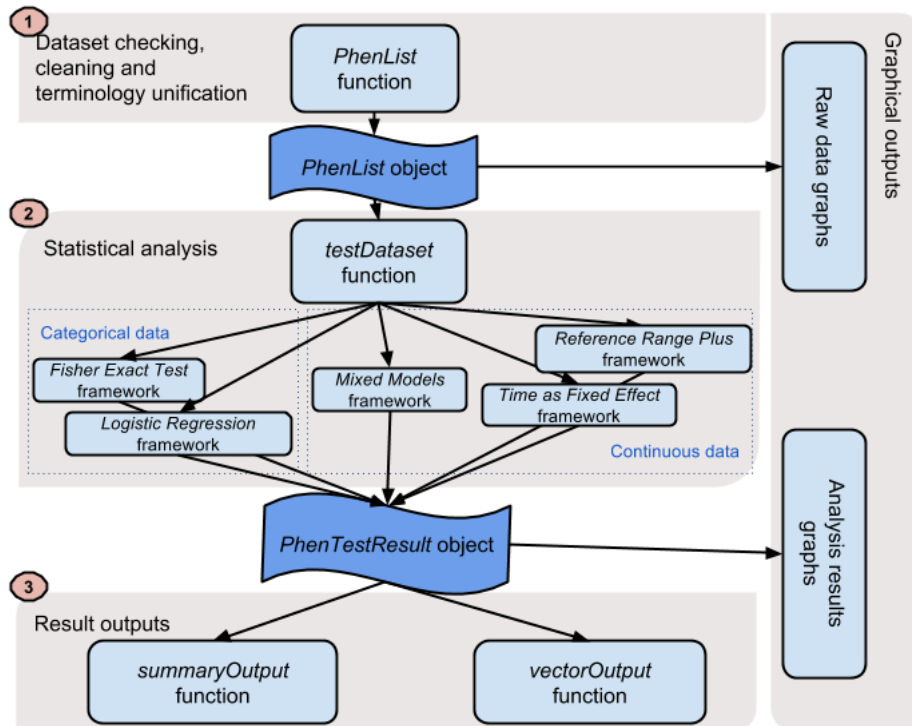
Figure 1: The PhenStat package's three stage structure: dataset processing, statistical analysis, and result output.

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.

8

2. Statistical analysis: is managed by function *testDataset* and consists of Mixed Model, Time Fixed Effect, Reference Range Plus, Fisher Exact framework and Logistic Regression framework implementations. The results are stored in *PhenTestResult* object. Potentially this layer can be extended adding new statistical methods.

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.

Package has a function *recommendMethod* to assess the suitability of the dataset for the various statistical analysis methods.

Package run time depends on a variety of factors including dataset size, computational resources, etc. Average analysis **run time** of the pilot dataset in our local environement 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.

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

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

In Table 1 the example dataset is presented with numerical variables of interest. Table 2 shows the example dataset with categorical data.

In addition to dependent variable column (the variable of interest) mandatory columns are "Genotype" and "Sex". The "Assay.Date" column is used to model "Batch" effect if not specified differently. "Weight" column is used to model body weight effect.

The information provided in the "Assay.Date" column is treated as a categorical variable with different strings as different levels. As such there is no requirement to provide the date in any particular format (ie D\M\Y or

M\D\Y) however please remove any time stamp. Removing the time stamp is necessary such that data will cluster by day appropriately.

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

- terminology unification (see section 2.1 for more details),

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

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

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

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

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

# 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 'Sex' 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!
```

Colony. Prefix	Gene. Name	Mouse	Genotype	Sex	Assay. Date	Age.In. Weeks	Weight	Bone Mineral. Density	Bone. Area	Lean. Mass	...
MBAU	Sparc	M00260466	Sparc/Sparc	Female	07-Aug-09	13.7	26.7	0.0443	8.46	17.29	
MBAU	Sparc	M00260467	Sparc/Sparc	Female	07-Aug-09	13.7	27.6	0.0427	7.95	15.99	
MBAU	Sparc	M00260468	Sparc/Sparc	Female	07-Aug-09	13.7	30.7	0.0451	8.95	17.73	
MBAU	Sparc	M00260475	Sparc/Sparc	Female	07-Aug-09	14.3	24.9	0.0443	8.43	14.84	
MBAU	Sparc	M00330799	Sparc/Sparc	Female	23-Nov-09	14	27.9	0.047	8.79	17.34	
MBAU	Sparc	M00330800	Sparc/Sparc	Female	23-Nov-09	14	25.1	0.0433	8.52	15.84	
MBAU	Sparc	M00330801	Sparc/Sparc	Female	23-Nov-09	14	21.7	0.0419	7.46	15.38	
MBAU	Sparc	M00226962	Sparc/Sparc	Male	23-Jun-09	13.9	32.8	0.0454	9.73	18.31	
MBAU	Sparc	M00226963	Sparc/Sparc	Male	23-Jun-09	13.9	38				
MBAU	Sparc	M00226964	Sparc/Sparc	Male	23-Jun-09	13.9	34.9	0.0471	11	19.64	
MBAU	Sparc	M00354835	Sparc/Sparc	Male	06-Jan-10	14.1	38.5	0.0483	10.28	21.91	
MBAU	Sparc	M00354836	Sparc/Sparc	Male	06-Jan-10	14.1	35.8	0.0486	9.65	20.8	
MBAU	Sparc	M00354837	Sparc/Sparc	Male	06-Jan-10	14.1	40.6	0.0483	10.08	24.25	
MBAU	Sparc	M00405764	Sparc/Sparc	Male	01-Mar-10	14	35.1	0.0465	9.25	20.25	
MBAU	Sparc	M00405766	Sparc/Sparc	Male	01-Mar-10	14	33.2	0.0465	10.17	19.54	
MBAU	Sparc	M00405767	Sparc/Sparc	Male	01-Mar-10	14	32.6	0.0444	8.88	20.07	
MALA	Ccdc57	M00194360	+/+	Female	05-May-09	14.3	23.9	0.0461	8.75	18.93	
MAOA	Yipf1	M00191904	+/+	Female	07-May-09	14	33.8	0.0512	8.8	19.71	
MAOA	Yipf1	M00191913	+/+	Female	07-May-09	14.3	29	0.0519	9.69	18.66	
MBFD	Gatc	M00191104	+/+	Female	08-May-09	14.1	26.5	0.0502	8.8	18.37	
MBFD	Gatc	M00191105	+/+	Female	08-May-09	14.1	40.7	0.0529	10.42	21.71	
MBFD	Gatc	M00195130	+/+	Female	08-May-09	13.1	26.1	0.0474	8.84	19.41	
MBFD	Gatc	M00191134	+/+	Female	08-May-09	13.9	33.1	0.0501	10.03	18.33	
MBHU	Mgst3	M00197544	+/+	Female	14-May-09	13.9	30.5	0.0477	9.08	17.92	
MBHU	Mgst3	M00197565	+/+	Female	14-May-09	14	34	0.0514	8.91	19.12	
MBHU	Mgst3	M00197566	+/+	Female	14-May-09	14	31.9	0.0496	9.01	18.17	
MBHU	Mgst3	M00197567	+/+	Female	14-May-09	14	35.6	0.0524	9.94	19.46	
MBRR	Myo5a	M00197538	+/+	Female	15-May-09	14	32.7	0.0503	9.27	19.12	
MAIT	Setdb1	M00195664	+/+	Female	18-May-09	14.1	31.2	0.0468	8.51	17.67	
...	...	...	...	...	...	...	...	...	...	...	...

Table 1: The continuous dataset example.

Colony. Prefix	Gene. Name	Mouse	Genotype	Sex	Assay. Date	Age.In. Weeks	Caudal. Processes	Cervical. Processes	Lumbar. Processes	Thoracic. Processes	...
MAPP	Smarcal1	M01005699	+/+	Male	12-Mar-12	13.9	Normal	Abnormal	Normal	Normal	
MCUB	Egfr	M01114643	+/+	Male	04-Jul-12	13.9	Normal	Normal	Normal	Normal	
MDYA	Srrm4	M01108371	+/+	Female	04-Jul-12	13.9	Normal	Abnormal	Normal	Normal	
MDYA	Srrm4	M01108373	+/+	Female	04-Jul-12	13.9	Normal	Normal	Normal	Normal	
MEFV	Gpr107	M01167360	+/+	Male	04-Sep-12	14	Normal	Abnormal	Normal	Normal	
MEFV	Gpr107	M01167394	+/+	Male	04-Sep-12	14	Normal	Abnormal	Normal	Normal	
MEFV	Gpr107	M01167391	+/+	Male	04-Sep-12	14	Normal	Abnormal	Normal	Normal	
MEFV	Gpr107	M01167363	+/+	Female	04-Sep-12	14	Normal	Abnormal	Normal	Abnormal	
MDYQ	Fam175b	M01128905	+/+	Female	17-Jul-12	13.9	Normal	Abnormal	Normal	Normal	
MDYQ	Fam175b	M01128906	+/+	Female	17-Jul-12	13.9	Normal	Abnormal	Normal	Normal	
MEGD	Slitrk4	M01140446	+/+	Male	06-Aug-12	14	Normal	Abnormal	Normal	Normal	
MEGD	Slitrk4	M01140447	+/+	Male	06-Aug-12	14	Normal	Normal	Normal	Normal	
MEGD	Slitrk4	M01140448	+/+	Male	06-Aug-12	14	Normal	Abnormal	Normal	Normal	
MDWN	G3bp2	M01051596	+/+	Male	01-May-12	13.9	Normal	Abnormal	Normal	Normal	
MDDY	Map3k1	M01049783	+/+	Male	01-May-12	13.9	Normal	Abnormal	Normal	Abnormal	
MDWN	G3bp2	M01051598	+/+	Male	01-May-12	13.9	Normal	Abnormal	Normal	Normal	
MCUB	Egfr	M01114647	+/+	Male	04-Jul-12	13.9	Normal	Normal	Normal	Normal	
MCUB	Egfr	M01114649	+/+	Male	04-Jul-12	13.9	Normal	Normal	Normal	Normal	
MCND	Aff3	M01268599	Aff3/+	Female	20-Dec-12	13.9	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01268600	Aff3/+	Female	20-Dec-12	13.9	Normal	Abnormal	Normal	Normal	
MCND	Aff3	M01176405	Aff3/+	Female	19-Sep-12	13.9	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01048233	Aff3/+	Female	30-Apr-12	13.9	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01087975	Aff3/Aff3	Male	13-Jun-12	14	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01051403	Aff3/Aff3	Female	30-Apr-12	14	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01127511	Aff3/Aff3	Female	31-Jul-12	14.3	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01127512	Aff3/Aff3	Female	31-Jul-12	14.3	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01024260	Aff3/Aff3	Male	28-Mar-12	14.3	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01257342	Aff3/Aff3	Male	18-Dec-12	14.1	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01257339	Aff3/Aff3	Male	18-Dec-12	14.1	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01069212	Aff3/Aff3	Female	23-May-12	13.7	Normal	Abnormal	Normal	Abnormal	
MCND	Aff3	M01087552	Aff3/Aff3	Female	13-Jun-12	13.7	Normal	Abnormal	Normal	Abnormal	
...	...	...	...	...	...	...	...	...	...	...	...

Table 2: The dataset example with categorical data.

2.1 Terminology Unification

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

The following *PhenList* function's arguments have to be specified to enable terminology unification to match expected columns to user names:

- *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" (user's definition has a priority over "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.sex* allows the user to define column name within dataset for the sex info if this column name is other than "Sex" 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 below dataset's values for females and males are 1 and 2 accordingly. Those values are changed to "Female" and "Male".

```
> file <- system.file("extdata", "test3.csv", package="PhenStat")
> dataset_test <- read.csv(file)
```

```
> 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 'Sex' 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. wildtype versus knockout). Thus the function *PhenList* requires users to define the reference genotype (mandatory argument *refGenotype* with default value "+/+") and test genotype (mandatory argument *testGenotype*). If the *PhenList* function argument *dataset.clean* is set to TRUE then all records with genotype values others than reference or test genotype are filtered out. The user may also specify hemizygotes genotype value (argument *hemiGenotype*) when hemizygotes are treated as the test genotype. This is necessary to manage sex linked genes, where the genotype will be described differently depending on the sex. Consider the following example of a knockout of a X-linked gene. In this situation, Table 3 describes the possible genotype labels and which should be compared biologically.

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

Table 3: Example of the dataset with sex linked genes

With the dataset described in Table 3 where *hemiGenotype* argument of the PhenList function is defined as "KO/Y", the actions of the function are: "KO/Y" genotypes are relabelled to "KO/KO" for males; females "+/KO" heterozygous are filtered out.

If a user would like to switch off filtering, (s)he can set *PhenList* function's argument *dataset.clean* to FALSE (default value is TRUE). In the following example the same dataset is processed successfully passing the checks procedures (see section 2.3) when *dataset.clean* is set to TRUE and fails at checks otherwise.

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

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 'Sex' column has following value(s): 'Female', 'Male'

# Filtering is switched 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 'Sex' 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 sex values.

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

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

Table 4: Example of the dataset with 3 sex 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 sex 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/sex values combination.

- if there is "Weight" column in the dataset then there are at least two weight records for each genotype/sex values combination.

If one of the checks fails, the function stops and the PhenList object is not created. In the following example "Sex" column is missed in the dataset and the checks fail. Note, a dataset can consist of one sex but a sex column is still required to ensure the appropriate model is fitted.

```
> dataset <- read.csv("test_noSexColumn.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 'Sex' column is missed.


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

Next example shows the results of the dataset described in the previous section 2.2 : three sex values and not enough records for the "unsexed" sex and both genotype values.

```
> dataset <- read.csv("test_3sexes.csv")
> test<-PhenList(dataset,
testGenotype="Mysm1/+")
...
Warning:
Since dataset has to have at least two data points for each genotype/sex combination
and there are not enough records for the combination(s): '+/+'/'unsexed' (0),
 'Mysm1/+'/'unsexed' (1), appropriate sex 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 'Sex' column has to have one or two values and currently the data has more than two.

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

Check failed:
Dataset should have at least two data points for each genotype/sex 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 will be avoided when *dataset.clean* argument of the *PhenList* function is set to TRUE (default value). See examples in this and in the previous section 2.2.

3 PhenList Object

The output of the *PhenList* function is the *PhenList* object that contains a cleaned dataset (*PhenList* object's slot *datasetPL*), reference genotype and test genotypes, terminology unification values. There is a number of useful methods available for the object, like *getStat* to obtain a simple statistics about dataset columns, *noSexes* to get a number of sex values available in the dataset, etc.

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 5).

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

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

> test@datasetPL
...
> getStat(test)
...
```

Table 5 shows the content of the *PhenList* method *getStat* output 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's classed continuous (variability is more than 0.5%), number of levels, number of data points and for the numerical variables various summary measures (mean, standard deviation, minimal and maximal values).

PhenList object has stored many characteristics about the data: reference genotype, test genotype, hemizygotes genotype, original column names, etc.

An example is given below.

```
> file <- system.file("extdata", "test2.csv", package="PhenStat")
> dataset2 <- read.csv(file)

> test2 <- PhenList(dataset=dataset2,
  testGenotype="Arid4a/Arid4a",
```

18

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
Sex	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 5: Simple statistics about dataset variables – *dataset.stat* content

```
      dataset.colname.weight="Weight.Value")

> testGenotype(test2)
[1] "Arid4a/Arid4a"


> refGenotype(test2)
[1] "+/+"


> getVariables(test2)
[1] "Age.In.Weeks"  "Alb"          "Alp"          "Alt"          "Amy" ...

> getColumn(test2,"Gluc")
[1] 18.96 16.73 26.61 ...
```

```
> getColumnBatchAdjusted(test2,"Gluc")
[1]-5.610609973  -7.840609973   2.039390027...
```

4   Statistical Analysis

The PhenStat package provides five methods (frameworks) for statistical analysis: Linear Mixed Models (MM), Time as Fixed Effect (TF) and Reference Range method (RR) for continuous data, Fisher Exact Test (FE) and Logistic Regression method (LR) for categorical data. In all frameworks, the statistical significance is assessed, the biological significance measured 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).

4.1   Manager for Analysis Methods – *testDataset* function

The *testDataset* function's argument *phenList* defines the *PhenList* object prepared for the analysis.

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 Time as Fixed Effect method the argument *method* is set to "TF". To perform Fisher Exact Test, the argument *method* is set to "FE". For the Reference Range Plus framework *method* is set to "RR" and finally, for the Logistic Regression framework *method* is set to "LR".

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

Function's argument *transformValues* defines to perform or not data transformation if needed. The default value is TRUE.

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

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

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

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

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

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

    1. *depVariable* column values are numeric.

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

    3. Variability check 2 (variability within a group): there are enough data points in subsets per genotype/sex combinations. The number of values from *depVariable* column should exceed *dataPointsThreshold* in all subsets.

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

- Additional Time as Fixed Effect (TF) framework checks:

    1. Number of batches: there are from 2 to 5 batches (assay dates) in the dataset which contain test genotype.

    2. Control points: there are concurrent controls data in the dataset, meaning the presence of data points for at least one sex in all genotype/batch level combinations.

- Reference Range Plus (RR) framework checks:

    1. *depVariable* column values are numeric.

2. There are data: the number of levels in *depVariable* column after filtering out of null values exceeds zero.

3. Control points: there are enough data points in subsets per reference genotype/sex combinations. The number of values from *depVariable* column should exceed *RR_controlPointsThreshold* in all subsets.

- Fisher Exact Test (FE) framework checks:

  1. There are data: the number of levels in *depVariable* column after filtering out of null values exceeds zero.

  2. Number of levels: number of *depVariable* levels is less than 10.

- Logistic Regression (LR) framework checks:

  1. There are data: the number of levels in *depVariable* column after filtering out of null values exceeds zero.

  2. Number of levels is two and values of the levels are 0 and 1.

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, the more complex MM and TF frameworks both have the functionality to operate in 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. More information about this two stages process is provided in section 4.2.3.

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.

The example how to call MM and FE framework is given below.

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

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

> result_MM_Lean.Mass <- testDataset(test,depVariable="Lean.Mass", method="MM",
  dataPointsThreshold=2)
...
> result_FE_Length <- testDataset(test,depVariable="Nose.To.Tail.Base.Length", method="FE")
..
```

Further details about the MM, TF, RR, FE and LR framework are in the next subsections.

4.2   Mixed Model Framework

First, we will describe the mixed model top-down methodology which starts with a fully loaded model and ends with final reduced model and genotype effect evaluation procedures as described in West *et al.* (2007).

4.2.1   Motivation

Through high throughput phenotyping programs, such as EUMODIC, where data was systematically collected on one genetic background, the significant sources of variation can be identified and it became obvious that batch (defined here as those readings collected on a particular day) can lead to large variation in phenotyping variables Karp *et al.* (2012).

Figure 2, demonstrates variation seen in control data from a standardised phenotyping pipeline.

Figure 3, demonstrates using artificially constructed data how mathematically this can arise from batch variation adding variability to the data. With this batch to batch variation, data collected on the same day will be more similar than other days, hence the readings are correlated and the assumption, of many statistical tests, of independent readings cannot be made. Furthermore, the variation with batch, means that this has to be considered to be able to assign causality i.e. if there is a difference in readings is this due to batch or genotype difference. Therefore these observations have significant implications for the data analysis of both high throughput and secondary phenotyping experiments where use of small batches of animals is common.
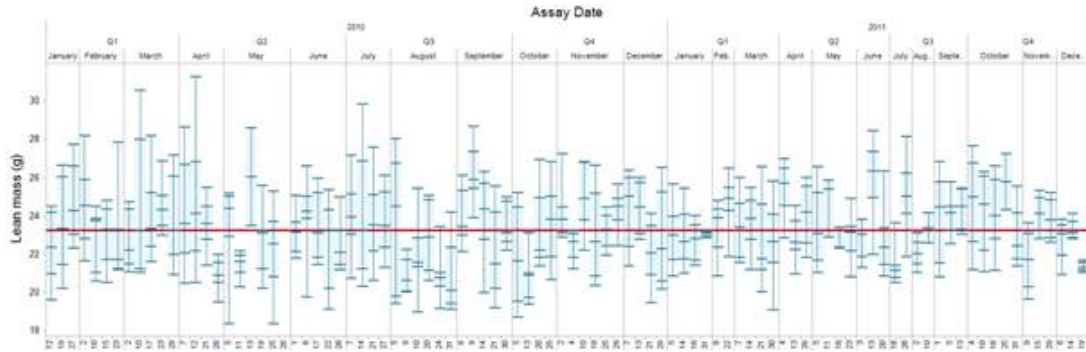
Figure 2: Representative time course plot showing the batch to batch variation in control data for male mice from a B6Brd;B6N-$Tyr^{c\text{-}Brd}$ genetic background from WTSI MGP program. Example shown is the variation seen in the lean mass variable measured in grams. For each day, data is collected a box plot is drawn as a five point summary indicating the minimum, 1st quartile, median, 3rd quartile and maximum. Shown in red is the global median fat mass value.
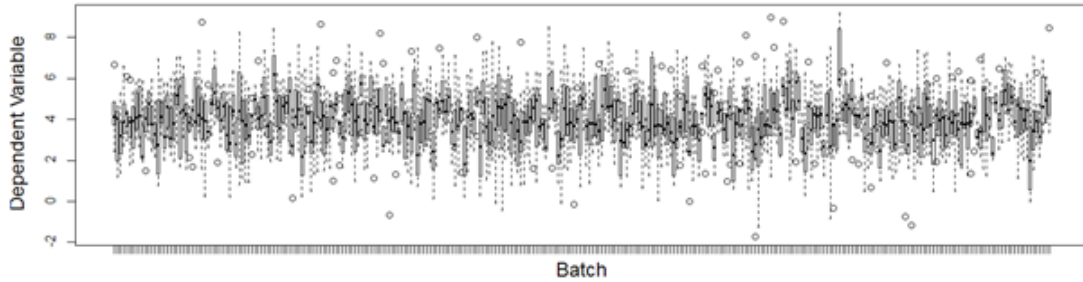
Figure 3: Artificially constructed data demonstrating how batch variation affects data distribution with time. In this artificial data, the variable of interest was assumed to be biologically randomly normally distributed variable with mean=7, standard deviation=1.5. To represent the batch effect, 300 assays days were generated with a randomly distributed batch effect (mean=0, standard deviation 0.5) which was added to the dependent variable biological mean before randomly sampling the dependent variable.

24

One option would be to ensure all animals for a line are processed in one day with concurrent controls. However, it is challenging and costly to produce sufficient animals of the right age within a narrow time point for an experiment. Consider the WTSI Sanger Mouse Genetics Project which requires 7 male and 7 female homozygote mice, generated by a heterozygote cross; a best case scenario would require 14 mating pairs being assembled at the same point in time Pinkert (2002). In order to generate these mating pairs, there would be a staged breeding process to generate the mice which involve several rounds of expansions depending on breeding success. This best case scenario is commonly hampered by fecundity, viability or other phenotypic problems within a line and hence to achieve a one batch pipeline the pairing number needs increasing significantly. In contrast, by accepting smaller numbers of mice in multiple batches, lower breeding pair numbers can be established. The smaller scale allows the generation of mice to answer firstly developmental and breeding issues and secondly to feed the pipeline over time and subsequent litters. As soon as mice are produced at the right age, these are feed into the pipeline. This batch approach, allows the pipeline to utilise animals that would otherwise be discarded as the process had not generated the required experimental sample size which ensure meet the high throughput pipeline needs and also help reduce the breeding cost per line. Furthermore, the operational constraints arising in a high throughput environment make optimal experimental design impractical; typically mutant and control mice are not assayed on the same day, so any phenotypic differences could be due to genotype or to subtle changes in the environment (e.g. temperature fluctuations or pipetting errors).

An alternative method, linear mixed models (MM) are a class of statistical models suited to modelling multiple sources of variability on a phenotype, where some explanatory factors (such as sex, weight and mutant genotype) are assumed to take fixed values that affect the population mean, whilst others such as batch are treated as affecting the covariance structure; animals from the same batch will have correlated phenotypes. Karp *et al.* (2012) demonstrated the utility and benefits of a MM framework for high throughput phenotyping data. The methodology used there has been developed further and refined for this package.

4.2.2 Theory

There are two possible start models, depending on whether weight is included as a factor (see Eq1 for the model without weight and Eq2 for the model

including weight).

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

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

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 Fig. 4).

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 Fig. 5). 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.

- Sex effect. Considered only when there are more than one sex in the dataset.

- Genotype by sex interaction effect. Considered only when there are more than one sex 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 in explaining variation in the dependent variable — and a covariance structure for the residuals that can be homogeneous or heterogeneous (see Fig. 4 Step 1-3). The selected model is then modified by reducing non-significant effects (see Fig. 4 Step 4 and Fig. 5).

When the final model is selected and reduced, the genotype effect is assessed by comparing a genotype and null model fitted with maximum likelihood evaluation method (ML). Finally, the final genotype model is refitted using restricted maximum likelihood evaluation method (REML) to get unbiased estimates of the variance parameters (see Fig. 4 Step 5,6 and Fig. 5).

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

See Models description (Model 1)

See Model Formula decision tree (Start model)

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

See Models description (Model 1 and 1A)

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

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

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

See Model Formula decision tree (Final model)

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

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

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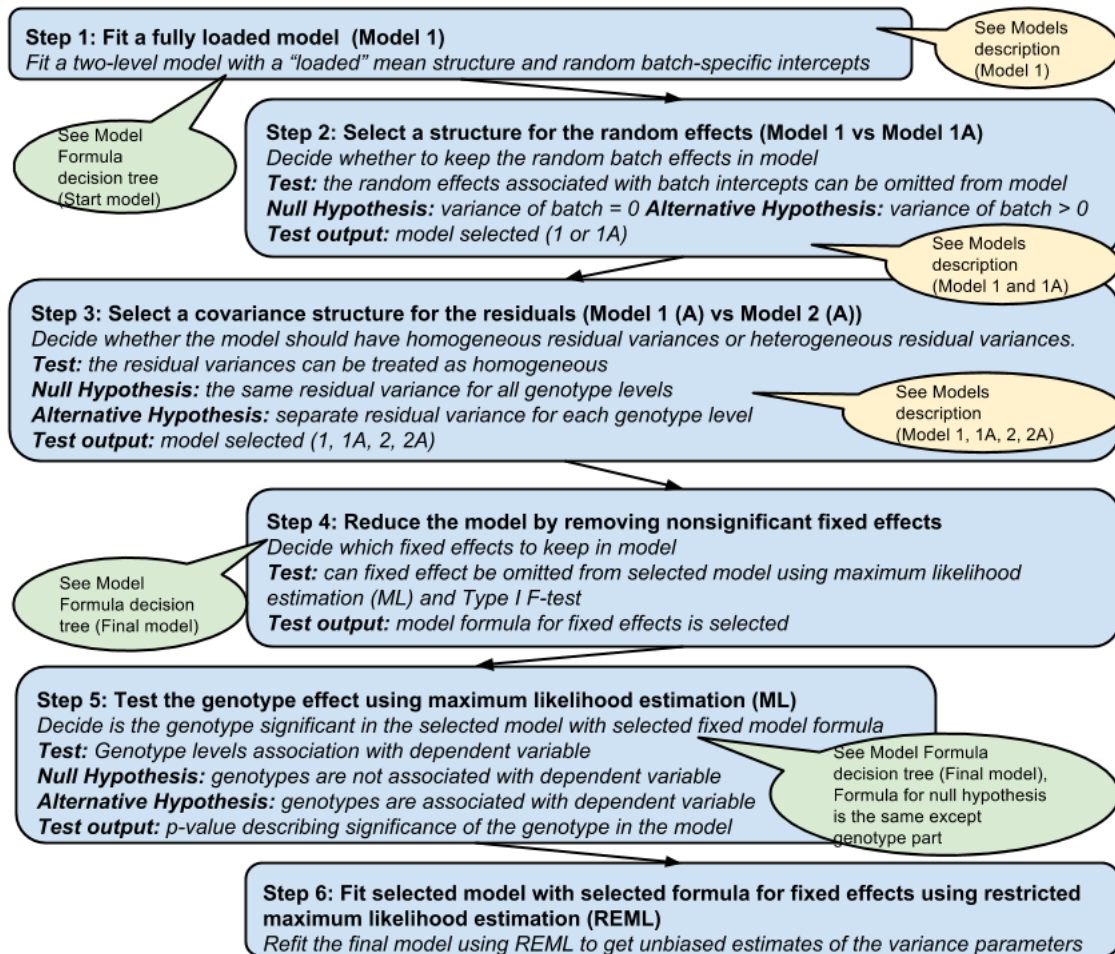*Refit the final model using REML to get unbiased estimates of the variance parameters*

Figure 4: MM framework steps: model selection process and model reducing by using significance of fixed effects.

Start Model Formula

"withoutWeight" — Which equation? — "withWeight" — Is Weight effect significant? — No

Sex levels = 2 (No / Yes)

Is Weight effect significant? — Yes

Sex levels = 2 (Yes / No)

Model Formula =
depVariable ~ Genotype

Model Formula =
depVariable ~ Genotype + Sex + Genotype*Sex

Model Formula =
depVariable ~ Genotype + Sex + Genotype*Sex + Weight

Model Formula =
depVariable ~ Genotype + Weight

Is Interaction* effect significant? (No / Yes)

Is Interaction* effect significant? (No / Yes)

Model Formula =
depVariable ~ Genotype*Male + Genotype*Female + Sex

Model Formula =
depVariable ~ Genotype*Male + Genotype*Female + Sex + Weight

Model Formula =
depVariable ~ Genotype + Sex

Model Formula =
depVariable ~ Genotype + Sex + Weight

Is Sex effect significant? (Yes / No)

Is Sex effect significant? (Yes / No)

Model Formula =
depVariable ~ Genotype

Model Formula =
depVariable ~ Genotype + Weight

Final Model Formula

* Interaction here means Genotype*Sex part of formula describing Genotype by Sex effects
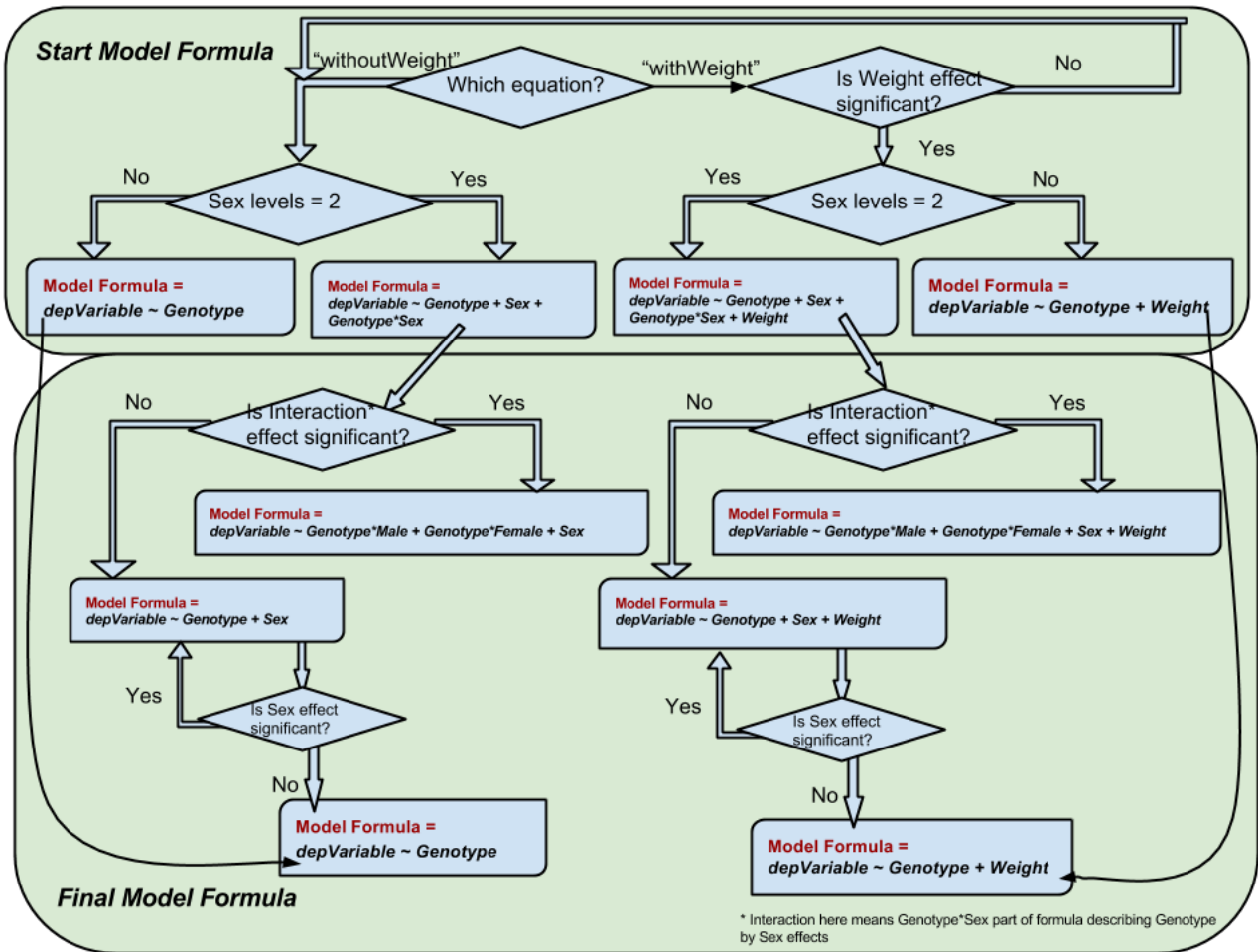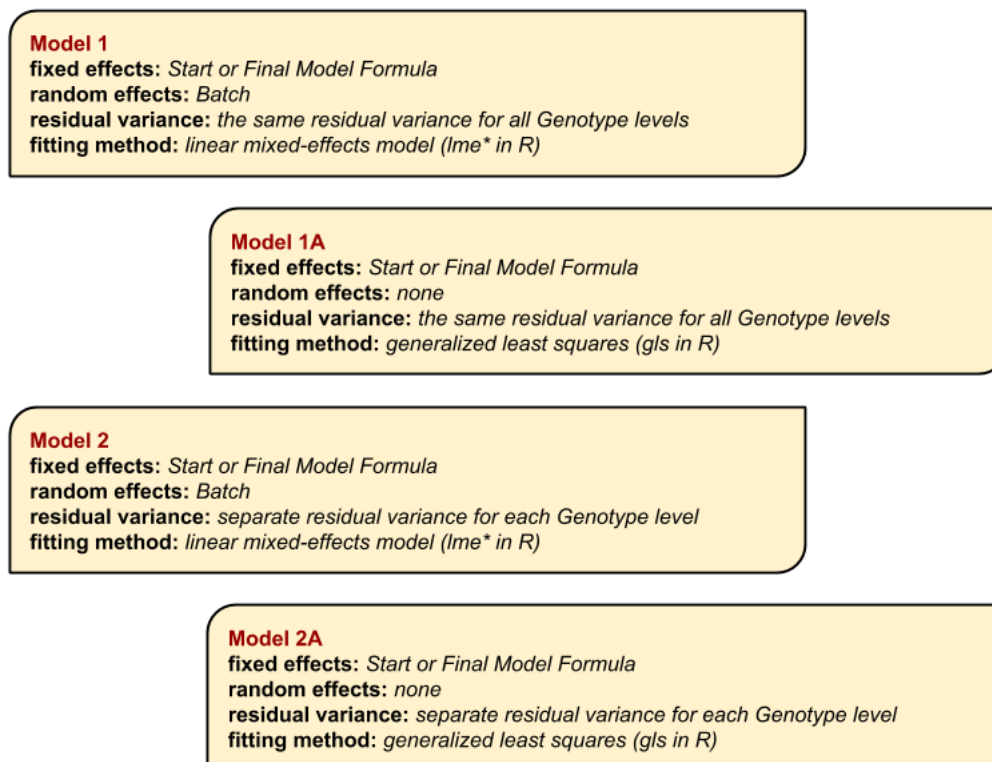
Figure 5: MM framework: start model formula and final model formula creation based on the dataset and significances of the effects (can be estimated or defined by user).

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

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

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

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

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

Figure 6: MM framework: different models that are considered.

4.2.3 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, sex effect, genotype by sex 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 sex effect significant, is sex 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, the selected model is failing to converge we simplify it by selecting the similar but simpler 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:
```

31

```
Method: Mixed Model framework.

Information:
Calculated values for model effects are: keepBatch=TRUE, keepVariance=TRUE,
keepWeight=FALSE, keepSex=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, keepSex=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.sex
[1] TRUE
> result$model.effect.interaction
[1] FALSE

> result$numberSexes
[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, keepSex=TRUE, keepInteraction=TRUE.

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

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

Information:
Equation: 'withoutWeight'.

> result <- finalModel(result)

> summaryOutput(result)
...
```

4.2.4 Diagnostics

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

The first one performs diagnostic tests to assess the MM quality of fit. This includes normality tests for the two genotype levels residuals, BLUPs (best linear unbiased prediction) and "rotated" residuals (Houseman *et al.* (2004)) (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. Consequently if calling directly it 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.

4.2.5   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"
```

When the function is called through *vectorOutput* function, the tag shown in Figure 7 will be proceeded by the phrase "If phenotype is significant", meaning that globally the test has not assessed whether there was a statistical significant difference just that if there was this would be the classification if it was statistically significant. When the function is called through *summaryOutput* function or directly there is an argument *phenotypeThreshold* (default value is 0.01), which sets the significance threshold of whether there is a phenotype of interest. If globally, the analysis indicates there is a statistically significant phenotype then the classification tag is appended to the phrase "With phenotype threshold value XXX".

The Mammalian Phenotype Ontology is under development as a community effort to provide standard terms for annotating mammalian phenotypic data and is housed and managed by the JAXS laboratory (see MP ontology).
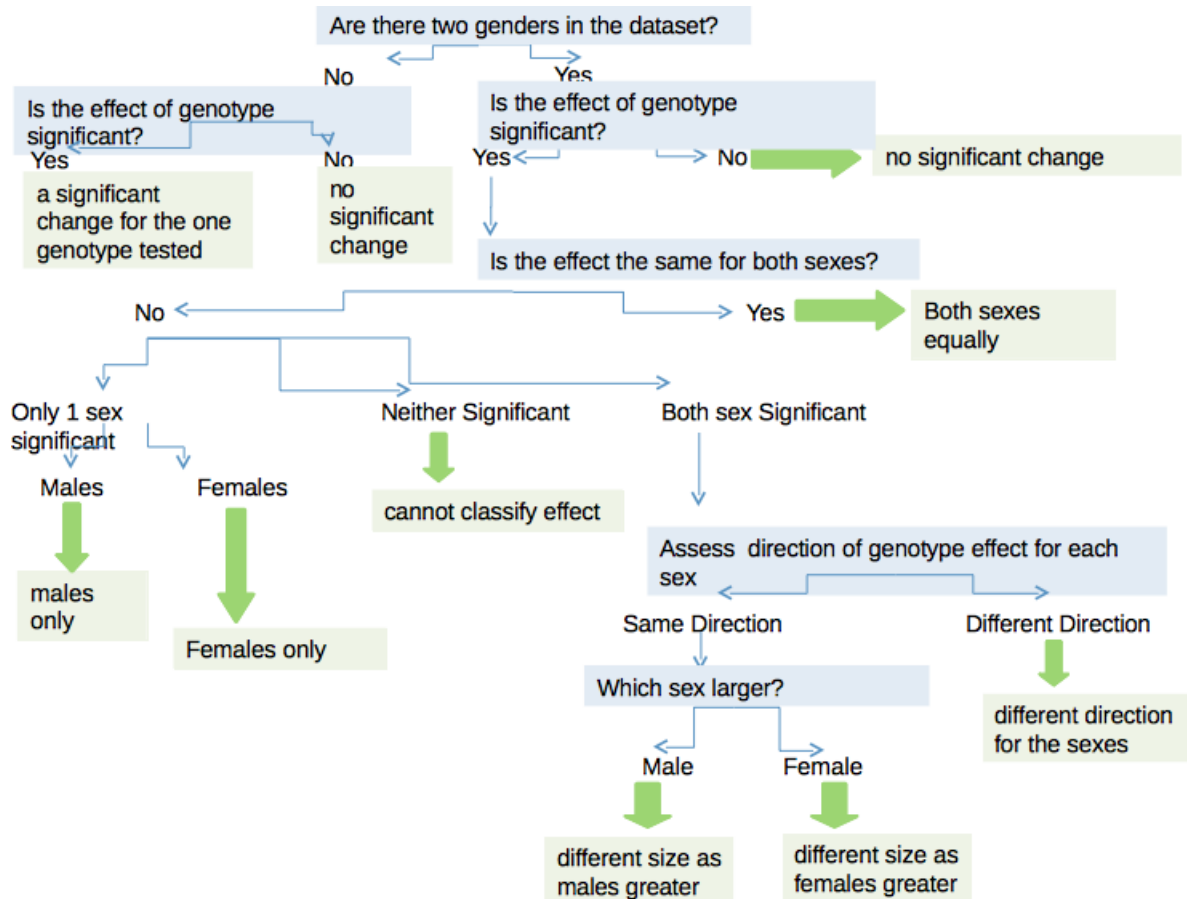
Figure 7: Assigning a classification tag. The output of the mixed model framework is queried to assign a classification tag of how the observed phenotype was observed across the two sexes. Within the decision tree, the question "Is the effect the same for both sexes? " is asking whether mathematically was there an interaction between the genotype and sex. Occasionally the procedure will find that there was a significant interaction but when it comes to identifying how this occurred and quantifying the effect for each sex, there is insufficient power. In this scenario the classification returned states that it "cannot classify the effect".
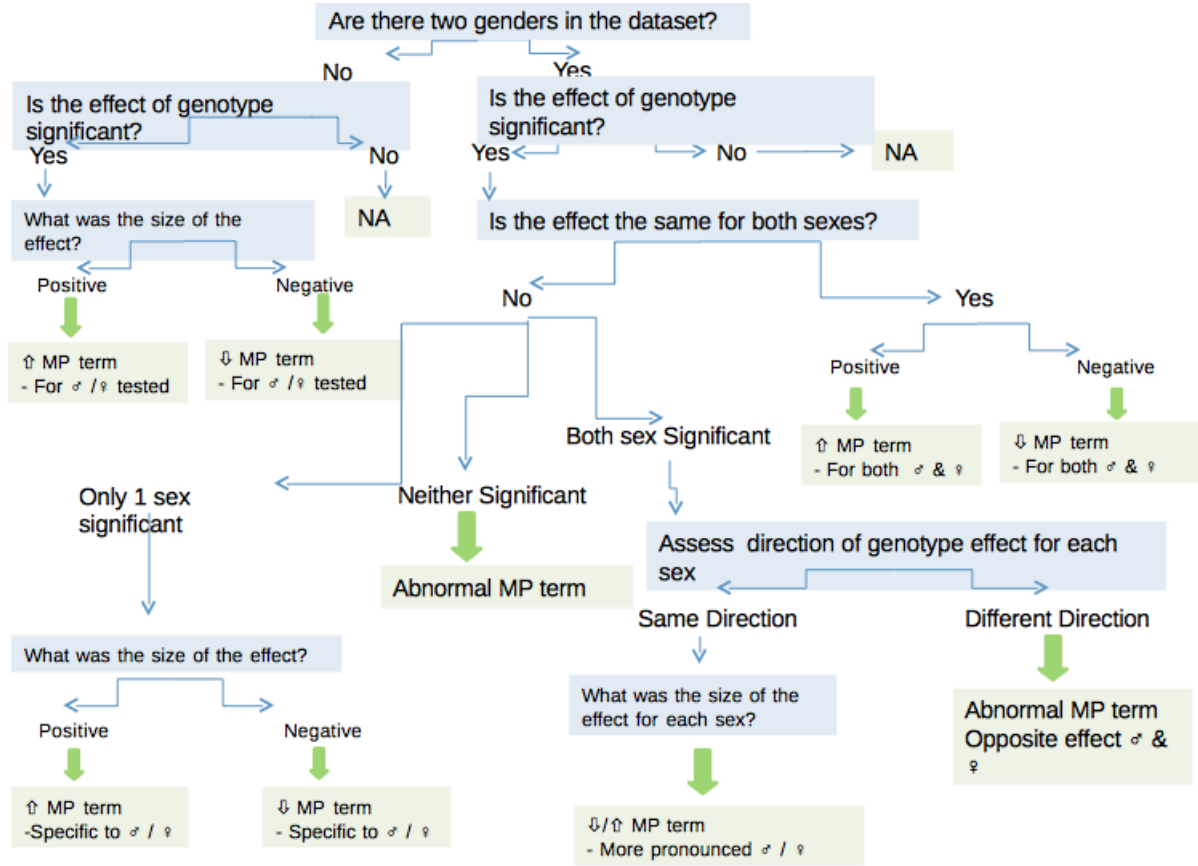
Figure 8: Assigning a Mammalian Phenotype (MP) ontology term in the presence of sexual dimorphism.

With the mixed model implementation, the output is very rich and a classification tag can be appended to the MP term to give richer information on the observed phenotype (e.g. abnormal circulating sodium levels – both sexes equally). Figure 8, details the decision tree that can be used with the MM output to interpret the results such a ontology term can be discerned and an annotation tag added when appropriate.

4.2.6 Model Failures and Jitter

In this section, we would like to discuss failure of model fitting and the potential use of jitter. At times, the model fitting process with the MM methodology would struggle to fit a model. In these instances, the PhenStat method would return an error message stating that the software can't fit
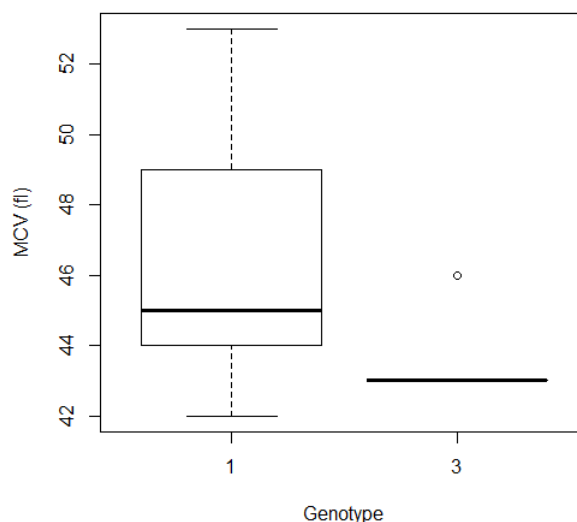
Figure 9: Mean corpuscular volume distribution for a dataset comparing a wildtype group (coded as 1) and a knockout group (coded as 3) of male mice. When PhenStat processes this data, models are not estimated as the software fails to converge on a solution and estimate the parameters, thus the software returns the error "Error: Can't fit the model ... Try to add jitter or RR plus method."

the the model and suggest that you could try jitter or an alternate method such as reference range approach. For example, this was seen with a Mean Corpuscular Volume dataset which had data only for male mice and the knockout mice had little variation as the majority of readings were 43 fl (shown in Fig. 9).

```
> file <- system.file("extdata", "test_jitter.csv", package="PhenStat")
> dataset_jitter <- read.csv(file)

> test_jitter <- PhenList(dataset=dataset_jitter,
    testGenotype="3", refGenotype="1",
    dataset.values.missingValue="null")

> result <- testDataset(test_jitter, depVariable="MCV",
    equation="withoutWeight")
Information:
Dependent variable: 'MCV'.

Information:
Perform all MM framework stages: startModel and finalModel.

Information:
```

```
Method: Mixed Model framework.


Error:
Can't fit the model MCV ~ Genotype. Try to add jitter or RR plus method.
```

Jitter is a function that adds a small amount of noise to a variable. The noise is added randomly at 1000th of the signal difference for that variable. In the example shown, the additional of noise allows the model to converge and estimate the values.

Code to add jitter and process the new variable:

```
> test_jitter@datasetPL$MCVWITHjitter <- jitter(test_jitter@datasetPL$MCV,
        factor =((max(test_jitter@datasetPL$MCV) - min(test_jitter@datasetPL$MCV)) / 1000))


> result_jitter <- testDataset(test_jitter, depVariable="MCVWITHjitter",
        equation="withWeight")


Information:
Dependent variable: 'MCVWITHjitter'.
...
> summaryOutput(result_jitter)


Test for dependent variable:
*** MCVWITHjitter ***


Method:
*** Mixed Model framework ***


------------------------------------------------------------------------
Model Output
------------------------------------------------------------------------
Final fitted model: MCVWITHjitter ~ Genotype
Was batch significant? TRUE
Was variance equal? FALSE
Genotype p-value: 7.872660e-03
Genotype effect: 1.3082 +/- 0.4792
Was there evidence of sexual dimorphism? no (p-value NA)
Genotype percentage change Male: 2.83%


------------------------------------------------------------------------
Classification Tag
------------------------------------------------------------------------
With phenotype threshold value 0.01 - a significant change for the one sex tested


------------------------------------------------------------------------
Model Output Summary
------------------------------------------------------------------------
              Value Std.Error  DF   t-value      p-value
```

```
(Intercept) 46.293656 0.4869087 238 95.076670 2.551272e-191
Genotype3    1.308178 0.4792010 238  2.729914  6.808624e-03
```

4.2.7 Reporting Biological Effect

The biological effect is reported in two formats: estimate and a percentage change.

The first measure is on a scale specific to that variable; whilst the second is relative to the wildtype signal and thus is scaled comparable across variables.

The estimate is in the scale of the variable and is the estimated change arising from being a knockout animal relative to the reference data. In the presence of sexual dimorphism it is estimated separately for the male and female knockout animals. In the *summaryOutput* view these estimates are reported in the ANOVA table. In the *vectorOutput* view these values are captured and reported as either Genotype estimate or Sex FvKO estimate and Sex MvKO estimate depending on whether sexual dimorphism was found to be significant.

The percentage change is the ratio of the genotype effect for a sex relative to the wildtype signal for that variable for that sex. When weight is not included in the model, it is entirely estimated from the coefficients returned from the model. When weight is included in the model, then the estimated genotype effect is reported relative to the average for that sex from the wildtype data. This distinction arises, because when weight is included the model the intercept is an estimate of the wildtype female animal with zero body weight.

The Figures 10 and 11 shows the various models and how the percentage change is estimated such the genotype effect is relative to the wildtype signal for that sex. The calculated values are in the *summaryOutput* and reported in the *vectorOutput* as "Genotype percentage change Male" and/or "Genotype percentage change Female".

See Figures 10 and 11 for the percentage change calculation details.

Example calculation:

```
> file <- system.file("extdata", "test1.csv", package="PhenStat")
> test=PhenList(dataset=read.csv(file), testGenotype="Sparc/Sparc")
...
> result <- testDataset(test,depVariable="Lean.Mass",
method="MM", equation="withoutWeight", transformValues=FALSE)
```

39

Model (Without Weight)	Female* Genotype Ratio	Male* Genotype Ratio
$Y = \beta_0 + \beta_1 Genotype$	$(\beta_1/\beta_0)*100$	$(\beta_1/\beta_0)*100$
$Y = \beta_0 + \beta_1 Genotype + \beta_2 Sex$	$(\beta_1/\beta_0)*100$	$(\beta_1/(\beta_0+\beta_2))*100$
$Y = \beta_0 + \beta_1 KO*male + \beta_2 KO*female$	$(\beta_2/\beta_0)*100$	$(\beta_1/\beta_0)*100$
$Y = \beta_0 + \beta_1 KO*male + \beta_2 KO*female + \beta_3 Sex$	$(\beta_2/\beta_0)*100$	$(\beta_1/(\beta_0+\beta_3))*100$

Figure 10: Calculation of the percentage change for model wihtout weight

Model (With Weight)	Female* Genotype Ratio	Male* Genotype Ratio
$Y = \beta_0 + \beta_1 Genotype + \beta_2 Weight$	$(\beta_1/(average\ +/+\ Female))*100$	$\beta_1/(Average\ +/+\ Male))*100$
$Y = \beta_0 + \beta_1 Genotype + \beta_2 Sex + \beta_2 Weight$	$\beta_1/(average\ +/+\ Female))*100$	$\beta_1/(Average\ +/+\ Male))*100$
$Y = \beta_0 + \beta_1 KO*male + \beta_2 KO*female + \beta_3 Weight$	$\beta_2/(average\ +/+\ Female))*100$	$\beta_1/(Average\ +/+\ Male))*100$
$Y = \beta_0 + \beta_1 KO*male + \beta_2 KO*female + \beta_3 Sex + \beta_4 Weight$	$\beta_2/(average\ +/+\ Female))*100$	$\beta_1/(Average\ +/+\ Male))*100$

Figure 11: Calculation of the percentage change for model that includes weight

```
...
> summaryOutput(result)
Test for dependent variable:
*** Lean.Mass ***

Method:
*** Mixed Model framework ***


---------------------------------------------------------------------------
Model Output
---------------------------------------------------------------------------

Final fitted model: Lean.Mass ~ Genotype + Sex
Was batch significant? TRUE
Was variance equal? TRUE
Genotype p-value: 6.751997e-04
Genotype effect: -1.8618 +/- 0.5418
Was there evidence of sexual dimorphism? no (p-value 6.330802e-01)
Genotype percentage change Female: -10.1%
Genotype percentage change Male: -8.22%


---------------------------------------------------------------------------
Classification Tag
---------------------------------------------------------------------------

With phenotype threshold value 0.01 - both sexes equally
```

```
--------------------------------------------------------------------------
Model Output Summary
--------------------------------------------------------------------------
                      Value Std.Error  DF    t-value       p-value
(Intercept)        18.440648 0.1696796 412 108.679253 2.044168e-305
GenotypeSparc/Sparc -1.861759 0.5417867 412  -3.436332  6.495823e-04
SexMale             4.209940 0.1607472 412  26.189818  1.018842e-89
```

$$Genotype\ percentage\ change\ Male = (-1.86/(18.44 + 4.21)) * 100 = -8.2\%$$

$$Genotype\ percentage\ change\ Female = (-1.86/18.44) * 100 = -10.1\%$$

If the weight is included in the fitted model the results are different:

```
> result <- testDataset(test,depVariable="Lean.Mass",
method="MM", equation="withWeight", transformValues=FALSE)
> summaryOutput(result)
Test for dependent variable:
*** Lean.Mass ***

Method:
*** Mixed Model framework ***


--------------------------------------------------------------------------
Model Output
--------------------------------------------------------------------------
Final fitted model: Lean.Mass ~ Genotype + Sex + Weight
Was batch significant? TRUE
Was variance equal? FALSE
Genotype p-value: 3.715089e-01
Genotype effect: -0.2914 +/- 0.3305
Was there evidence of sexual dimorphism? no (p-value 1.022353e-01)
Genotype percentage change Female: -1.59%
Genotype percentage change Male: -1.3%


--------------------------------------------------------------------------
Classification Tag
--------------------------------------------------------------------------
With phenotype threshold value 0.01 - no significant change


--------------------------------------------------------------------------
Model Output Summary
--------------------------------------------------------------------------
                      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
SexMale             1.6407343 0.18080930 411  9.0743913 4.791912e-18
```

```
Weight              0.3430502 0.01808121 411 18.9727422 4.147891e-58
```

```
> wt_subset <- subset(test@datasetPL, Genotype=="+/+")
> wt_subset_females <- subset(wt_subset, Sex=="Female")
> wt_subset_males <- subset(wt_subset, Sex=="Male")
> mean(wt_subset_females$Lean.Mass,na.rm=TRUE)
[1] 18.28705
> mean(wt_subset_males$Lean.Mass,na.rm=TRUE)
[1] 22.49357
```

$$Genotype\ percentage\ change\ Male = (-0.29/22.49) * 100 = -1.3\%$$

$$Genotype\ percentage\ change\ Female = (-0.29/18.28) * 100 = -1.59\%$$

4.3 Time Fixed Effect Framework

Time Fixed Effect framework manages data variation by estimating each batch effect to separate it from genotype.

4.3.1 Motivation

Through high throughput phenotyping programs, such as EUMODIC, where data was systematically collected on one genetic background, the significant sources of variation can be identified and it became obvious that batch (defined here as those readings collected on a particular day) can lead to large variation in phenotyping variables Karp *et al.* (2012). A high throughput phenotyping program will have a well-defined phenotyping pipeline which consists of a sequence of phenotyping procedures carried out at specific ages. To date, standardisation has focused on the experimental methods by which data were collected [5,6]. However the workflow – the practical implementation of a pipeline – varies from center to center. Each center's workflow is a balance of resources, other goals (e.g. allowing for additional phenotyping depending on earlier results) and throughput requirements. Differences in the number and frequency of controls, whether knockout animals are phenotyped at one time or in multiple batches, and blinding methodologies are the most important variables. One strategy implemented is to accept that animals for a knockout line are obtained in small batches due to fertility or fecundity issues, but with each small batch control animals are concurrently phenotyped. This allows an analysis using a regression method where batch

is treated as a fixed effect. We have called this analysis framework "Time Fixed Effect".

4.3.2 Theory

There are two possible start models, depending on whether weight is included as a factor (see Eq3 for the model without weight and Eq4 for the model including weight).

$$depVariable \backsim Genotype + Sex + Genotype * Sex + Batch \qquad \text{(Eq3)}$$

$$depVariable \backsim Genotype + Sex + Genotype * Sex + Weight + Batch \quad \text{(Eq4)}$$

The same top-down approach can be used here as in Mixed Models framework for the model optimization process. When batch is insignificant then the Mixed Model formula is the same as Time Fixed Effect framework's formula.

As in the case of Mixed Model framework the final model construct is influenced by a number of criteria. The only one difference is that there are no random effects in the model.

The following criteria (effects) are considered:

- Batch effect (batch variation). **NB!** In TF framework batch effect is a fixed effect.

- Residual variances homogeneity where homogeneous residual variances means the variance for all genotype levels is tested equivalent.

- Body weight effect. Considered only when Eq4 is used.

- Sex effect. Considered only when there are more than one sex in the dataset.

- Genotype by sex interaction effect. Considered only when there are more than one sex in the dataset.

The selection of model is influenced by the batch effect – is batch in the dataset, and if so, is it significant in explaining variation in the dependent variable – and a covariance structure for the residuals that can be homogeneous or heterogeneous. The selected model is then modified by reducing non-significant effects. When the final model is selected and reduced, the genotype effect is assessed by comparing a genotype and null model fitted

with maximum likelihood evaluation method (ML). Finally, the final geno-type model is refitted using restricted maximum likelihood evaluation method (REML) to get unbiased estimates of the variance parameters.

4.3.3 Implementation

The analysis requires the removal of dates which are not concurrent with knockout animals. This is achieved by the *TFDataset* function that goes through all dataset's records and removes ones that don't have data in test genotype and in reference genotype for the same batch level (assay date) at least for one sex. Summary statistics on the cleaning impact are then provided following the table of data.

See the example below:

```
> file <- system.file("extdata", "test7_TFE.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(file),
                testGenotype="het",
                refGenotype = "WT",
                dataset.colname.sex="sex",
                dataset.colname.genotype="Genotype",
                dataset.values.female="f",
                dataset.values.male= "m",
                dataset.colname.weight="body.weight",
                dataset.colname.batch="Date_of_procedure_start")
...
> test_TF <- TFDataset(test,depVariable="Cholesterol")

Data points containing 'Cholesterol' by batch levels:
```

	WT	WT	het	het
Batch	Female	Male	Female	Male
* 02.09.2013	7	4	0	0
* 03.02.2014	0	1	0	0
...				
* 13.01.2014	4	2	0	0
* 13.05.2013	7	5	0	0
* 14.01.2014	5	7	0	0
14.04.2014	0	4	4	3

```
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 14.10.2013 |          7 |          5 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 15.07.2013 |          5 |          5 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 16.06.2014 |          8 |          8 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 16.09.2013 |          2 |          3 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|   17.02.2014 |          6 |          4 |          3 |          4 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 17.03.2014 |          4 |          5 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|   18.02.2014 |          0 |          3 |          2 |          3 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 18.03.2014 |          4 |          3 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
...


* - removed record(s)

Number of batch levels left: 3
Records removed (reference genotype): 92%
Records removed (test genotype): 0%
...


result <- testDataset(test_TF,depVariable="Cholesterol",method="TF")

Information:
Dependent variable: 'Cholesterol'.


Information:
Perform all TF framework stages: startTFModel and finalTFModel.


Information:
Method: Time as Fixed Effect framework.


Information:
Equation: 'withWeight'.


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

The cleaned dataset contains only three batches which correspond to concurrent control approach.

```
| ---------- | ---------- | ---------- | ---------- | ---------- |
|            |         WT |         WT |        het |        het |
```

```
| ---------- | ---------- | ---------- | ---------- | ---------- |
|     Batch  |    Female  |      Male  |    Female  |      Male  |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|  14.04.2014 |         0  |         4  |         4  |         3  |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|  17.02.2014 |         6  |         4  |         3  |         4  |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|  18.02.2014 |         0  |         3  |         2  |         3  |
| ---------- | ---------- | ---------- | ---------- | ---------- |
```

Similarly to the MM framework there are two functions in the PhenStat package that implements the Time Fixed Effect framework:

- *startTFModel* function evaluates model's criteria and stores the result in the *PhenTestResult* object;

- *finalTFModel* 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.

4.3.4 Diagnostics

The vectorOutput function includes statistical tests for normality on the residuals for the wildtype and residuals for the knockout (see section 4.2.4). These normality tests are provide to assist in the building automated tools for assessing model fit, however when there is a lot of data, the statistical test can be overall sensitive to departures from normality and when the number of data points is low, the test can lack ability to detect deviations from normality.

The normality tests are driven by the function *testFinalModel* and it returns a list of the following values:

1. Reference genotype value.

2. Normality test result (p-value) for the reference genotype's residuals.

3. Test genotype value.

4. Normality test result (p-value) for the test genotype's residuals.

5. NA

6. NA

46

For example:

```
> testFinalModel(result)
[1] "WT"                "0.803172853943264" "het"
[4] "0.788809875034823" NA                  NA
```

Graphical methods are the recommended method for assessing model fits. Model fit quality can also be assessed graphically using *qqplotGenotype*, *boxplotResidualBatch* and *plotResidualPredicted* (see case study example 10.2 for example usage).

4.3.5 Classification Tag

The *classificationTag* function returns a classification tag to assign a sexual dimorphism assessment of the phenotypic change from the results of the TF framework. Figure 7 shows the decision tree that is used to determine the tag assigned.

```
> classificationTag(result)
[1] "With phenotype threshold value 0.01 - both sexes equally"
```

When the function is called through *vectorOutput* function, the tag will be proceeded by the phrase "If phenotype is significant", meaning that globally the test has not assessed whether there was a statistical significant difference just that if there was this would be the classification if it was statistically significant. When the function is called through *summaryOutput* function or directly there is an argument *phenotypeThreshold* (default value is 0.01), which sets the significance threshold of whether there is a phenotype of interest. If globally, the analysis indicates there is a statistically significant phenotype then the classification tag is appended to the phrase "With phenotype threshold value XXX".

4.3.6 Reporting Biological Effect

Within Time Fixed Effect framework the biological effect is also reported in two formats: estimate and a percentage change. The meaning of the calculated values is the same as in Mixed Model framework. Please see section 4.2.7 for details.

4.4 Fisher Exact Test Framework

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

4.4.1 Motivation

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.

4.4.2 Implementation

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 for effect size calculation.

From the chi squared table statistical significance is assessed using a Fisher Exact Test whilst the biological significance is estimated by an effect size (see section 10.4.4 for more details).

These are calculated separately for 3 subsets (if there are multiple sex values in the dataset):

- combined dataset (regardless the sex values),
- males only subset,
- females only subset.

All results are stored in *PhenTestResult* object:

```
> file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> dataset_cat <- read.csv(file)

> test_cat <- PhenList(dataset=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 'Sex' 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"
> getVariable(result_cat)
[1] "Thoracic.Processes"
> result_cat@method
[1] "FE"
> method(result_cat)
[1] "FE"
> noSexes(result_cat)
[1] 2

# Chi squared table for all data
> getCountMatrices(result_cat)$all
         +/+ Aff3/Aff3
Abnormal 142        12
Normal   753         1

# Chi squared table for males only records
> getCountMatrices(result_cat)$male
         +/+ Aff3/Aff3
Abnormal  59         5
Normal   390         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
> getPercentageMatrix(analysisResults(result_cat)[[3]])

             +/+ Aff3/Aff3
Abnormal 15.86592 92.307692
Normal   84.13408  7.692308


# Effect size for all data
> analysisResults(result_cat)[[1]]@ES
[1] 76


# Effect size for females only records
> analysisResults(result_cat)[[3]]@ES
[1] 81


# Fisher Exact Test results for all data
> analysisResults(result_cat)[[1]]@modelOutput

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
> pvalue(analysisResults(result_cat)[[1]])
[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 8 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:

```
> file <- system.file("extdata", "test_categorical_normal.csv", package="PhenStat")
```

```
> dataset_cat_normal <- read.csv(file)
> test2 <- PhenList(dataset=dataset_cat_normal, testGenotype="Aff3/Aff3")
> result2 <- testDataset(test2,depVariable="Thoracic.Processes", method="FE")
> levels(factor(result2@analysedDataset$Thoracic.Processes))
[1] "Normal"
> summaryOutput(result2)


Test for dependent variable:
*** Thoracic.Processes ***

Method:
*** Fisher Exact Test framework ***


-----------------------------------------------------------------------------
Model Output ('*' highlights results with p-values less than threshold 0.01)
-----------------------------------------------------------------------------
All
  p-value:      1.0000
  Effect size: 0%


Males only
  p-value:      1.0000
  Effect size: 0%


Females only
  p-value:      1.0000
  Effect size: 0%


-----------------------------------------------------------------------------
Classification Tag
-----------------------------------------------------------------------------
Not significant


-----------------------------------------------------------------------------
Count Matrices
-----------------------------------------------------------------------------
All
       +/+ Aff3/Aff3
Normal 895        13
Other    0         0

Males only
       +/+ Aff3/Aff3
Normal 449         6
Other    0         0

Females only
       +/+ Aff3/Aff3
Normal 446         7
```

51

```
Other    0        0
```

4.4.3 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 across the different datasets.

Having the Fisher Exact Test results for three datasets: males only, females only, all (combined set) the p-values are compared with *phenotypeThreshold* argument which default value is 0.01. For each one set the function calculates is genotype significant or not (significant if p-value is less than *phenotype-Threshold*) and then assigns classification tag. In the Table 6, the possible classification tags are presented.

No sexes	Genotype significant in				Classification Tag
	general	males only	females only	combined set	
1	yes	-	-	yes	Significant for the sex tested
1	no	-	-	no	Not significant
2	no	no	no	no	Not significant
2	yes	no	no	yes	Significant in males and in combined dataset
2	yes	no	yes	no	Significant in females dataset only
2	yes	yes	no	no	Significant in males dataset only
2	yes	yes	no	yes	Significant in males and in combined dataset
2	yes	yes	yes	no	Significant in males and in females dataset
2	yes	no	yes	yes	Significant in females and in combined dataset
2	yes	yes	yes	yes	Significant in males, females and in combined dataset

Table 6: Classification tag assignment by using Fisher Exact Tests results for three sets: males only, females only, all (combined set).

4.5 Reference Range Plus Framework

At times phenotyping data has been collected in such a way, that typical analysis routes are not considered reliable. The conservative, easy to understand reference range method (RR framework) can have a role in identifying phenodeviants.

4.5.1 Theory

The reference range methodology is considered a conservative method where data are classed normal, low or high depending on whether they lie within the natural variation seen within the control.
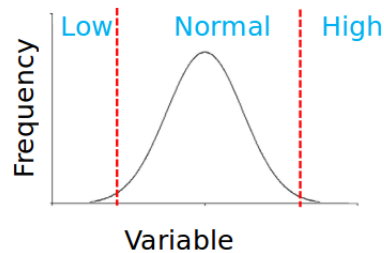

Figure 12: Control data are used to find the ranges for normal, low and high classifications.

Once the data are classified by using the ranges calculated from control data, then a Fisher Exact Test is used to compare the movement towards the low or high classification.

	Low	Normal/High
+/+	3	97
Knockout	6	1

p value: 1.59e-7
Effect Size: 82.7%

	Normal/Low	High
+/+	98	2
Knockout	7	0

p value: 1
Effect Size: 2%

Figure 13: Fisher Exact Test is used to calculate p-values.

4.5.2 Implementation

The RR methodology is driven from the *testDataset* function manager. To analyse the data using the RR methodology the argument *method* needs to be set to "RR". For the RR methodology there are a number of additional arguments that determine how the RR is executed. The *RR_naturalVariation* argument (defaults to 95, minimal value is set to 60) determines how much of the data is classified as normal and thus is used in the calculations of the classification threshold. The thresholds are determined using a percentile method, which avoids any distribution assumptions. For example, when *RR_naturalVariation* defaults to 95, then 95% of the data will be classed as normal, i.e. the 2.5 and 97.5 percentile thresholds of the dependent variable are used to define the boundaries of normal.

The *RR_controlPointsThreshold* argument (defaults to 60) determines the minimum number of data points per sex that are used to determine a reference range. Since the mathematical minimum is recommended as 40 data points (see Solberg (1983) for details) *RR_controlPointsThreshold* minimal value is also set to 40.

Data is analysed in three ways: 1) male only, 2) female only, and 3) all (combined dataset). In the all classification, the classifications of normal, low, and high are combined in the count table but please note the classification to normal, low, or high are run on a sex specific basis. In the situation where only one sex is present the output is returned in the all category.

If we had compared normal vs low and high classification simultaneously there would have been a potential problem of having a highly significant p-value yet an ES that is low. This can arise when there is movement away from the normal classification to both a low and high classification at the same time. As people want to use one filter this was a problem.

Our solution is to compare movement towards low separately from movement towards high classifications. As a result an adjustment for the multiple testing is needed (we use simple multiplication of p-value by two) since there are two tests performed: low vs normal/high (low classification) and high vs normal/low (high classification).

The effect size calculated is the maximum percentage change seen in the low and high classification. 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 control.

All results are stored in *PhenTestResult* object:

```
> file <- system.file("extdata", "test_Akt2.csv", package="PhenStat")
> DEXAdata <- read.csv(file)
> test <- PhenList(dataset=DEXAdata,
  testGenotype="Akt2/Akt2",
  refGenotype="WT",
  dataset.colname.batch="dateOfExperiment",
  dataset.values.female="female",
  dataset.values.male="male",
  dataset.colname.genotype="Genotype",
  dataset.colname.sex="Gender")
...
> result <- testDataset(test,
depVariable="Lean.Mass",
method="RR")
...
> summaryOutput(result)

Test for dependent variable:
*** Lean.Mass ***

Method:
*** Reference Ranges Plus framework ***

1) High vs Normal/Low
          All Females only Males only
p-value 1.0000        1.0000     1.0000
ES         3%            3%         3%

2) Low vs Normal/High
          All Females only Males only
p-value 0.0000        0.0000     0.0000
ES         59%           72%        47%


----------------------------------------------------------------------
Classification Tag
----------------------------------------------------------------------
With phenotype threshold value 0.01 - significant in males (Low), females (Low) and in combined dataset (Low)


----------------------------------------------------------------------
Thresholds
----------------------------------------------------------------------
Natural variation:           95
Min control points:          60
Normal values 'males only':   18.10925 to 30.0315
Normal values 'females only': 14.38975 to 23.52675


----------------------------------------------------------------------
```

```
Count Matrices
--------------------------------------------------------------------------
All
            WT Akt2/Akt2
Low         30        16
Normal/High 1116      10

All
            WT Akt2/Akt2
High        30         0
Normal/Low 1116       26

Females only
            WT Akt2/Akt2
Low         15         9
Normal/High 559        3

Males only
            WT Akt2/Akt2
Low         15         7
Normal/High 557        7

Females only
            WT Akt2/Akt2
High        15         0
Normal/Low 559        12

Males only
            WT Akt2/Akt2
High        15         0
Normal/Low 557        14
```

4.5.3   Classification Tag

We've also implemented the function *classificationTag* for the RR framework with the same restrictions as for Fisher Exact Test – it is not sexual dimorphism classification, but rather the overall estimation of the signals significance across various datasets.

The table of possible classification tags is very similar to the Fisher Exact Test classification table (see Table 6) with one additional element: next to each one dataset name (males only, females only, combined) we added the classification movement value. The possible values are "Low, "High" and "NA". "NA" value is assigned when there are both low and high classifications significant for the particular subset.

For example, if in all three subsets the genotype is significant and movement is classified as "high" the classification tag is "Significant in males (High), females (High) and in combined dataset (High)".

4.6 Logistic Regression Framework

Logistic Regression Framework is a framework suitable for categorical data when the variable of interest has been recoded to 0 and 1. This method models the relationship between the variable of interest and a number of independent variables assessing the impact of sex, genotype and whether the genotype effect depends on the sex.

4.6.1 Motivation

The logistic regression allows the assessment of the genotype impact on phenotype for both sexes simultaneously and allows a statistical assessment of whether the genotype effect has a sexual dimorphic element. The inclusion of both sexes simultaneously increases statistical power and by the inclusion of a sexual dimorphic assessment allows a more refined assessment of the genotype effect.

The approach implemented uses the "logistf" package of R, which is a biased reduction logistic regression which provides a refinement to the logistic regression method to allow studies with rare event categorical data Heinze (2006).

The "logistf" package does not support the inclusion of random effects which was used in the MM framework to model batch in high throughput studies. This is not an issue for the majority of genotype-phenotype studies on categorical data as the phenotype traits are typically rare events studies looking for rare abnormal phenotypes and day to day variation would not influence this phenotype.

4.6.2 Theory

The method starts with equation X

depVariable = Genotype + Sex + Genotype * Sex (EqLR)

$$depVariable \frown Genotype + Sex + Genotype * Sex \qquad (Eq5)$$

The same top-down approach is used here as in Mixed Models framework for the model optimization process, though the model optimisation steps are fewer compared to the mixed model method. The optimisation considers two aspects assessing whether they statistically significant (p-value less than 0.05) in explaining variation in the model before selecting the final model:

- Sex effect. Considered only when there are more than one sex in the dataset.

- Genotype by sex interaction effect. Considered only when there are more than one sex in the dataset.

As an exploration validation test, we have built a routine in to assess whether batch was significant however it cannot influence the final fitted model as the biased reduction methodology used here cannot work with random effects.

When the final model is selected and reduced, the genotype effect is assessed by comparing a genotype and null models. Finally, the final genotype model is fitted to estimate the effects associated with each term included in the model.

Unlike standard regression which returns estimates (coefficients) that predict the change in the dependent variable for one unit change in the independent variable, Logistic Regression is looking at probabilities. The Logistic Regression model estimates are still measures of the contribution to variations in the dependent variable but as a log odd estimate. To increase interpretability you can relate this back to the experiment by converting the estimate into an odds ratio by calculating the exponential of the value (i.e. E^β or $\exp(^\beta)$). Where an odds ratio is an indicator of the change in odds resulting from a unit change in the predictor. For example, with an odds ratio of 2 which tells us that being a testGenotype animal leads to a 7.4 higher odds of being abnormal.

4.6.3 Implementation

The analysis requires the recoding of the data to 0 and 1 where 0 is the reference phenotype and 1 is the abnormality. This is achieved by the *LRDataset* function that recodes all abnormal phenotypes as 1 and the remainder as 0.

Similarly to the MM framework there are two functions in the PhenStat package that implements the Logistic Regression Effect framework:

- *startLFModel* function evaluates model's criteria and stores the result in the PhenTestResult object;

- *finalLFModel* function builds the final model using the model's criteria from PhenTestResult object and assesses the genotype effect and fits the final model.

By default, both functions will be called from *testDataset* manager sequentially.

See the example below:

```
> file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(file),
                   testGenotype="Aff3/Aff3")
> test2 <- LRDataset(test, depVariable="Thoracic.Processes",
  abnormalValues="Abnormal")
> result2 <-testDataset(test2,depVariable="Thoracic.Processes",method="LR")
> summaryOutput(result2)


Test for dependent variable:
*** Thoracic.Processes ***

Method:
*** Logistic Regression ***


-----------------------------------------------------------------------
Model Output
-----------------------------------------------------------------------
Final fitted model: Thoracic.Processes ~ Genotype + Sex
Was batch significant? FALSE
Genotype p-value: 1.943939e-09
Genotype effect: 3.7983 +/- 0.9034
Was there evidence of sexual dimorphism? no (p-value 5.441219e-01)


-----------------------------------------------------------------------
Classification Tag
-----------------------------------------------------------------------
With phenotype threshold value 0.01 - both sexes equally


-----------------------------------------------------------------------
Model Output Summary
-----------------------------------------------------------------------

                   Value Std.Error   ci.lower    ci.upper      p-value
(Intercept)     -1.4646494 0.1209814 -1.7077333 -1.23321580 0.000000e+00
GenotypeAff3/Aff3  3.7983232 0.9033813  2.3619588  6.02332990 1.943939e-09
SexMale         -0.4251768 0.1840563 -0.7891747 -0.06661705 2.002838e-02
```

4.6.4 Classification Tag

The *classificationTag* function returns a classification tag to assign a sexual dimorphism assessment of the phenotypic change from the results of the LR framework. Figure 7 shows the decision tree that is used to determine the tag assigned.

```
> classificationTag(result)
[1] "With phenotype threshold value 0.01 - both sexes equally"
```

When the function is called through *vectorOutput* function, the tag will be proceeded by the phrase "If phenotype is significant", meaning that globally the test has not assessed whether there was a statistical significant difference just that if there was this would be the classification if it was statistically significant. When the function is called through *summaryOutput* function or directly there is an argument phenotypeThreshold (default value is 0.01), which sets the significance threshold of whether there is a phenotype of interest. If globally, the analysis indicates there is a statistically significant phenotype then the classification tag is appended to the phrase "With phenotype threshold value XXX".

4.6.5 Understanding the estimates

The model estimates represent change in the probability of being a member of the modeled categories (e.g. female versus male or refGenotype versus testGenotype). The estimated coefficients are expressed in log units and are not directly interpretable. They can be converted to an odds ratio by taking the coefficient and using it as the power to which the base of the natural logarithm (2.71828) is raised, the result represents the change in the odds of the modeled event associated with a one-unit change in the independent variable (i.e. for sex going from female to male or for genotype going from refGenotype to the testGenotype). If a coefficient is positive, its transformed log value will be greater than one, meaning that the modeled event is more likely to occur. If a coefficient is negative, its transformed log value will be less than one, and the odds of the event occurring decrease. A coefficient of zero (0) has a transformed log value of 1.0, meaning that this coefficient does not change the odds of the event one way or the other.

5 Data Transformation

The *testDataset* function has an argument "transformValues" which can be set to TRUE or FALSE. By default the value is set to FALSE meaning NOT to perform transformation.

The "MM" and "TF" regression methods make an assumption of normally distributed data. Unfortunately at times, a non-linear transformation is needed to improve the distribution of the data to improve the model fitting reliability. The Box-Cox transformation (Box and Cox (1964)) represents a family of power transformations with a procedure to find the normalizing transformation for each variable. The procedure returns the optimal lambda value that specifies the transformation necessary and the procedure identifies the value by testing a variety of lambda values. The Box-Cox power transformation is not a guarantee for normality as the procedure looks to minimise standard deviation based on the assumption that the transformed data has the highest likelihood to be normally distributed when the standard deviation is the smallest. The Box-Cox transformation of the variable is defined as:

$$y'_i = \begin{cases} \frac{(y_i + scaleShift)^{\lambda} - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(y_i + scaleShift), & \text{otherwise} \end{cases} \tag{1}$$

The parameter λ is estimated using the profile-likelihood function. The parameter *scaleShift* is a scaling correction for negative data. The Box-Box transformation incorporates many tradition transformations (table 7).

λ	Transformation equivalent	
	Name	Specification
0.5	Square root transformation	$\sqrt{Y}$
0.33	Cube root transformation	$\sqrt[3]{Y}$
0.25	Fourth root transformation	$\sqrt[4]{Y}$
0	Log transformation	$\ln(Y)$
-0.5	Reciprocal square root transformation	$\frac{1}{\sqrt{Y}}$
-1	Reciprocal transformation	$\frac{1}{Y}$
-2	Reciprocal power 2 transformation	$\frac{1}{Y^2}$

Table 7: Relationship between lambda and common transformations.

PhenStat has an implementation of the Box-Cox transformation to determine

if a transformation is needed and if so how the data should be transformed. Implementation steps:

1. **Scaling assessment**

   The Box-Cox power transformation only works if all the data is positive and greater than 0. This can be achieved by adding a scaling correction (*scaleShift*) to all data before it is transformed. This value if required is returned in both the *summaryOutput* and the *vectorOutput* functions.

2. **Determine Lambda**

   The Box-Cox procedure is applied to the data taking into account structure in the data by specifying them as fixed effects and use profile-likelihood techniques to derive the optimal λ and the 95% confidence interval for λ.

3. **Convert output to transformation requirement**

   - If the statistical method is "FE", "LR" or "RR" or if the user has chosen to perform analysis without data transformation (*test-Dataset* function's argument *transformValues* is set to FALSE) then no transformation is needed ("lambda=NA, scaleShift=NA, transformed=FALSE, code=0")

   - If the 95% confidence interval for λ includes 1 then no transformation is needed ("lambda=value, scaleShift=value, transformed=FALSE, code=1").

   - If the λ value is between -0.1 and 0.1, at the same time 1 is not included into the 95% confidence interval for λ then the value is classed as 0 and a log transformation is performed ("lambda=0, scaleShift=value, transformed=TRUE, code=2").

   - If the optimal λ value exceeds 5 or -5 then we consider that transformation is not appropriate ("lambda=value, scaleShift=value, transformed=FALSE, code=4"). This has been implemented as with these large λ values the resulting transformed variable would have values that were too large/small for meaningful regression analysis. This can happen with small datasets or datasets that are poorly suited to regression analysis (e.g. datasets with little variation).

   - For all other lambda values the implementation performs power transformation using the optimal λ value ("lambda=value, scaleShift=value, transformed=TRUE, code=3").

VectorOutput results concerning transformation are classified in table 8.

Code	λ	Values transformed	Comment
0	NA	No	Statistical method doesn't required data transformation or user requested to perform without transformation.
1	value	No	According to test results transformation is not needed.
2	0	Yes	Log transformation is applied.
3	value	Yes	Power transformation is applied.
4	value	No	Optimal lambda value is not found within a range of -5 to 5.

Table 8: VectorOutput values explaining transformation.

As shown in figure 14, the transformation recommended by PhenStat package lead to an improvement in the residual distribution improving the model fit.

After transformation the model estimates are on a new scale and hence for interpretability it can be useful to convert these values back to the original scale. In the *summaryOutput* function, the genotype estimate and standard error is converted and this is reported as "Genotype effect (original scale)". For a database users, or if you are interested in other model variables, below are the conversion functions:

$$y_i = \begin{cases} sign(y_i' * \lambda + 1) * |y_i' * \lambda + 1|^{\frac{1}{\lambda}} - scaleShift, & \text{if } \lambda \neq 0 \\ e^{y_i'} - scaleShift, & \text{otherwise} \end{cases} \quad (2)$$

$$sign(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

6 PhenTestResult Object

The output of the *testDataset* function is the *PhenTestResult* object regardless the statistical method used. The following data are stored in this object:
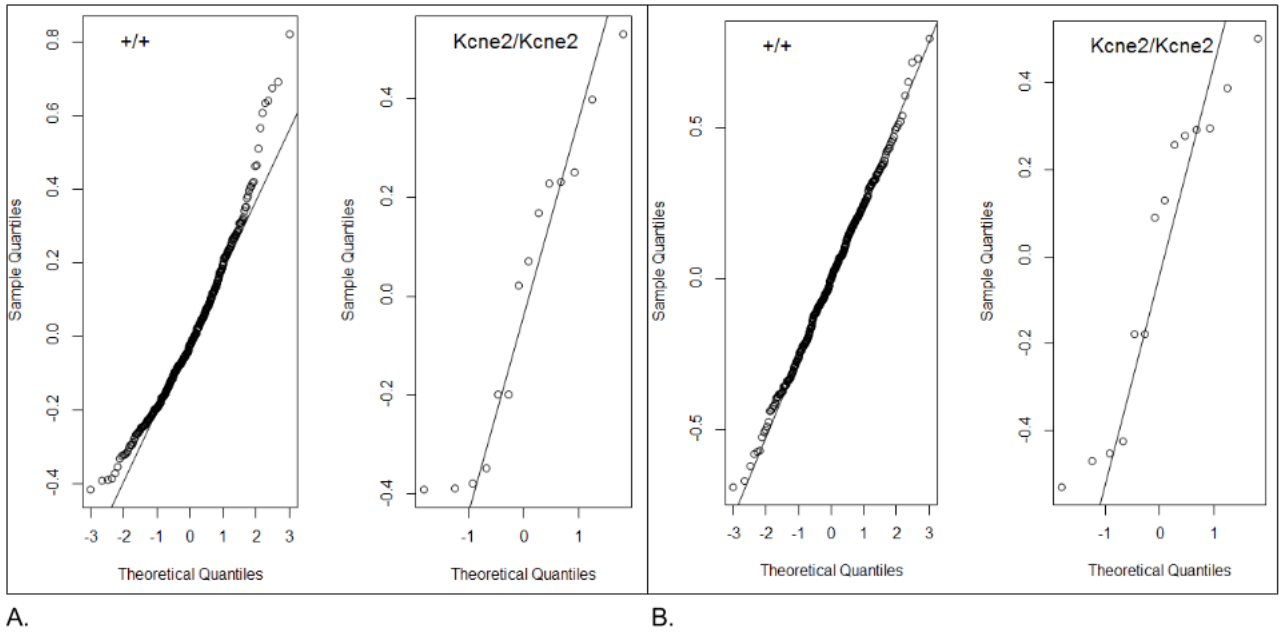
Figure 14: Impact of transformation on model residuals. Mixed model analysis of the *Kcne2* knockout of blood plasma triglyceride data obtained at the Wellcome Trust Sanger Institute. PhenStat transformation assessment recommended a log transformation. **A:** Residuals obtained on fitting a model to the raw data. **B:** Residuals obtained on fitting a model to the PhenStat transformed data.

- a subset that actually was analysed (slot *analysedDataset*) with the following columns: original values of dependent variable, sex values and genotype values, transformed values of dependent variable if transformation was applied, batch values and batch adjusted values of dependent variable if batches are present in the dataset, weight values if weight is in the dataset;

- a name of the variable that was analysed (slot *depVariable*);

- reference and test genotypes (slots *refGenotype* and *testGenotype*),

- information about transformation if it was applied (slots *transformationRequired*, *lambdaValue* and *scaleShift*);

- statistical method used and method's parameters (slots *method* and *parameters*);

- analysis results (slot *analysisResults*).

There is a number of useful methods available for the *PhenTestResult* object, like *batchIn* and *weightIn* to figure out the batch/weight presence, *noSexes* to get a number of sex values available in the dataset, etc.

There is also a method specific for regression analysis methods (MM, TF, LR): *getGenotypeEffect* to get report of biological effect and a method specific for the Fisher Exact Test based frameworks (FE, RR): *getCountMatrices* to obtain all count matrices.

The content of the *analysisResults* slot depends on the statistical method that has been used. It contains output of regression analysis for MM, TF and LR frameworks and number of "htest" objects with additional meta data (like subset, classification, effect size, count matrix) for Fisher Exact Test based frameworks (FE, RR). "htest" class is used by *fisher.test* function to return Fisher Exact Test results.

"FE" framework returns *analysisResults* slot with a list of 3 (1 in case of one sex only) extended "htest" objects since there are three subsets: "all", "males only", "females only". "RR" framework returns *analysisResults* slot that contains a list of 6 (2 in case of one sex only) extended "htest" object since there are three subsets ("all", "males only", "females only") and two classifications (low and high).

There are special methods available for the extended "htest" objects. For example: *matrixCount* to get count matrix, *getPercentageMatrix* to obtain percentage matrix, *getColumnView* to obtain p-value and ES value together with meta data.

```
> file <- system.file("extdata", "test_Akt2.csv", package="PhenStat")
> DEXAdata <- read.csv(file)
> test <- PhenList(dataset=DEXAdata,
  testGenotype="Akt2/Akt2",
  refGenotype="WT",
  dataset.colname.batch="dateOfExperiment",
  dataset.values.female="female",
  dataset.values.male="male",
  dataset.colname.genotype="Genotype",
  dataset.colname.sex="Gender")
> result <- testDataset(test,
depVariable="Lean.Mass",
method="RR")


# print out count matrices and percentage matrices for male only subset
> for (i in seq_along(analysisResults(result))) {
      val <- analysisResults(result)[[i]]
      if (analysedSubset(val)=="males"){
        print(comparison(val))
        print("Count matrix")
          print(matrixCount(val))
          print("Percentage matrix")
          print(getPercentageMatrix(val))
      }
 }
[1] "Low vs Normal/High"
[1] "Count matrix"
            WT Akt2/Akt2
Low          15         7
Normal/High 557         7
[1] "Percentage matrix"
                  WT Akt2/Akt2
Low          2.622378        50
Normal/High 97.377622        50
[1] "High vs Normal/Low"
[1] "Count matrix"
            WT Akt2/Akt2
High         15         0
Normal/Low  557        14
[1] "Percentage matrix"
                  WT Akt2/Akt2
High         2.622378         0
Normal/Low  97.377622       100
```

7 Method Recommendation for Dataset

Sometimes the analysis path to follow is obvious. For example, the variable "head shape" with options "Normal" or "Abnormal" is clearly categorical and therefore the Fisher Exact Test (FE) framework appropriate. At times, it is however less obvious. An example, could be the variable "number of vertebrates". Initial as this is a number this suggest continuous methods (e.g. MM, TF or RR) however on inspection there is no variation and in fact this variable is best treated as categorical. Based on our experience, we have prepared a function called *recommendMethod* which explores the data and makes recommendations on analysis that can be conducted. Please note, at times multiple analysis paths could be suitable.

```
file <- system.file("extdata", "test1.csv", package="PhenStat")

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

recommendMethod(test,"Lean.Mass")
...

[1] "MM and RR"
```

Function *recommendMethod* goes through all common and framework's specific checks and makes recommendations according to the check failure/success results.

8 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.

8.1 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
> file <- system.file("extdata", "test1.csv", package="PhenStat")

> dataset1 <- read.csv(file)

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

> result <- testDataset(test,
    depVariable="Lean.Mass",outputMessages=FALSE, transformValues=FALSE)

> summaryOutput(result)

Test for dependent variable:
*** Lean.Mass, power transformed with lambda value = -0.7  ***

Method:
*** Mixed Model framework ***


----------------------------------------------------------------------------
Model Output
----------------------------------------------------------------------------
Final fitted model: Lean.Mass ~ Genotype + Sex + Weight
Was batch significant? TRUE
Was variance equal? TRUE
Genotype p-value: 1.141215e-01
Genotype effect (original scale): 0.9962 +/- 1.0025
Was there evidence of sexual dimorphism? no (p-value 8.917040e-01)
Genotype percentage change Female: -0.31%
Genotype percentage change Male: -0.3%


----------------------------------------------------------------------------
Classification Tag
----------------------------------------------------------------------------
With phenotype threshold value 0.01 - no significant change


----------------------------------------------------------------------------
Model Output Summary
----------------------------------------------------------------------------
                       Value    Std.Error  DF    t-value      p-value
(Intercept)          1.178594383 0.0033490036 411 351.923894 0.000000e+00
GenotypeSparc/Sparc -0.003843757 0.0024479122 411  -1.570219 1.171337e-01
SexMale              0.010122217 0.0010264122 411   9.861747 9.871536e-21
Weight               0.002009066 0.0001029706 411  19.511070 1.757719e-60
```

The summary output for MM and TF frameworks contains metrics about the fitted model:

- **Intercept** is the reference level after all other factors are accounted for. For example, for equation with weight (Eq2) in fully loaded model intercept is reference genotype's female with zero weight.

- **Value** stands for the estimated coefficient. 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 is 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 lower the more important is the 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, p-values and effect size measures.

```
> file <- system.file("extdata", "test_categorical.csv", package="PhenStat")

> dataset_cat <- read.csv(file)

> test2 <- PhenList(dataset=dataset_cat,
    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 ('*' highlights results with p-values less than threshold 0.01)
--------------------------------------------------------------------------
All
* p-value:     0.0000
* Effect size: 76%


Males only
* p-value:     0.0003
* Effect size: 70%


Females only
* p-value:     0.0000
* Effect size: 81%


--------------------------------------------------------------------------
Classification Tag
--------------------------------------------------------------------------
With phenotype threshold value 0.01 - significant in males, females and in combined dataset


--------------------------------------------------------------------------
Count Matrices
--------------------------------------------------------------------------
All
         +/+ Aff3/Aff3
Abnormal 142        12
Normal   753         1

Males only
         +/+ Aff3/Aff3
Abnormal  59         5
Normal   390         1

Females only
         +/+ Aff3/Aff3
Abnormal  83         7
Normal   363         0
```

The output of the "RR" framework *summaryOutput* function includes count matrices, p-values and effect size measures separately for High vs Normal/Low and Low vs Normal/High classifications and thresholds used for the calculations.

```
> result <- testDataset(test,depVariable="Lean.Mass", method="RR")
Information:
Dependent variable: 'Lean.Mass'.
```

```
Information:
Method: Reference Ranges Plus framework.


> summaryOutput(result)


Test for dependent variable:
*** Lean.Mass ***


Method:
*** Reference Ranges Plus framework ***


1) High vs Normal/Low
            All Females only Males only
p-value 1.0000       1.0000     1.0000
ES         3%           3%         3%


2) Low vs Normal/High
            All Females only Males only
p-value 0.0120       0.0198     0.2510
ES        17%          26%         9%


---------------------------------------------------------------------------
Classification Tag
---------------------------------------------------------------------------
Not significant


---------------------------------------------------------------------------
Thresholds
---------------------------------------------------------------------------
Natural variation:            95
Min control points:           60
Normal values 'males only':   18.830 to 26.630
Normal values 'females only': 15.586 to 20.6035


---------------------------------------------------------------------------
Count Matrices
---------------------------------------------------------------------------
All
            +/+ Sparc/Sparc
Low          13            3
Normal/High 435           12


All
            +/+ Sparc/Sparc
High         12            0
Normal/Low  436           15


Females only
              +/+ Sparc/Sparc
```

```
Low             6           2
Normal/High 221             5


Males only
            +/+ Sparc/Sparc
Low             7           1
Normal/High 214             7


Females only
            +/+ Sparc/Sparc
High            6           0
Normal/Low 221              7


Males only
            +/+ Sparc/Sparc
High            6           0
Normal/Low 215              8
```

The output of the "LR" framework *summaryOutput* function includes summary model information and the model output:

```
> file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> test <- PhenList(dataset=read.csv(file),
                   testGenotype="Aff3/Aff3")
> test2 <- LRDataset(test, depVariable="Thoracic.Processes",
  abnormalValues="Abnormal")
> result2 <-testDataset(test2,depVariable="Thoracic.Processes",method="LR")
> summaryOutput(result2)


Test for dependent variable:
*** Thoracic.Processes ***

Method:
*** Logistic Regression ***


----------------------------------------------------------------------------
Model Output
----------------------------------------------------------------------------
Final fitted model: Thoracic.Processes ~ Genotype + Sex
Was batch significant? FALSE
Genotype p-value: 1.943939e-09
Genotype effect: 3.7983 +/- 0.9034
Was there evidence of sexual dimorphism? no (p-value 5.441219e-01)


----------------------------------------------------------------------------
Classification Tag
----------------------------------------------------------------------------
With phenotype threshold value 0.01 - both sexes equally


----------------------------------------------------------------------------
```

```
Model Output Summary
-------------------------------------------------------------------------
                      Value Std.Error   ci.lower    ci.upper      p-value
(Intercept)       -1.4646494 0.1209814 -1.7077333 -1.23321580 0.000000e+00
GenotypeAff3/Aff3  3.7983232 0.9033813  2.3619588  6.02332990 1.943939e-09
SexMale           -0.4251768 0.1840563 -0.7891747 -0.06661705 2.002838e-02
```

8.2   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 of the analysis framework.

Example of the *vectorOutput* function results for the MM framework:

```
> vectorOutput(result)
Method
"MM framework, linear mixed-effects model, equation with weight"
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"
Sex estimate
"1.64073430331594"
Sex standard error
"0.180809296427475"
Sex p-val
"4.79191190571249e-18"
Weight estimate
"0.343050209791982"
Weight standard error
"0.0180812139273457"
Weight p-val
"4.1478905048872e-58"
...
```

Results of *vectorOutput* function for TF framework looks very similar to the MM output.

In the Table 9 vector output values are described.

As was mentioned above *vectorOutput* format is the same for all frameworks. However, in case of "FE", "RR" and "LR" many values are not defined. For example, *vectorOutput* results for "FE" framework:

```
> vectorOutput(result_cat)
Method
"Fisher Exact Test framework"
Dependent variable
"Thoracic.Processes"
Batch included
"NA"
Residual variances homogeneity
"NA"
Genotype contribution
"NA"
Genotype estimate
"76"
Genotype standard error
"NA"
Genotype p-Val
"4.35745946092922e-09"
...
Gp1 genotype
"+/+"
...
Gp2 genotype
"Aff3/Aff3"
...
Sex FvKO estimate
"81"
...
Sex FvKO p-val
"1.00779809539594e-05"
Sex MvKO estimate
"70"
Sex MvKO standard error
"NA"
Sex MvKO p-val
"0.00025633944344021"
Classification tag
"With phenotype threshold value 0.01 - significant in males, females and in combined dataset"
Additional information
"NA"
```

#	Name	Value	Framework	Description
1	Method	String	MM, TF, RR, FE, LR	Possible values: "Mixed Model/Time as Fixed Effect framework", linear mixed-effects model/generalized least squares, equation with weight/equation without weight", "Reference Range Plus framework", "Fisher Exact Test framework", "Logistic Regression".
2	Dependent variable	String	MM, TF, RR, FE, LR	Name of the dependent variable.
3	Batch included	TRUE/FALSE	MM, TF, LR	Was batch significant in the model?
4	Residual variances homogeneity	TRUE/FALSE	MM, TF	Was variance equal in the model?
5	Genotype contribution	Numeric	MM, TF, RR, FE, LR	For the MM and TF it is the p-value from the test of genotype contribution. Whilst for the FE and RR it is the p-value from either a test of one sex (if only one sex exists) or it is the output of the combined datasets.
6	Genotype estimate	Numeric	MM, TF, RR, FE, LR	Estimated coefficient that describes genotype value calculated by MM, TF, LR or effect size estimates in FE and RR.
7	Genotype standard error	Numeric	MM, TF, LR	Standard error of the coefficient estimate for genotype.
8	Genotype p-val	Numeric	MM, TF, RR, FE, LR	Genotype p-value that represents the probability the genotype is NOT relevant. p-value for combined set in FE and RR. In case of RR framework there are two values reported: low classification p-value, high classification p-value.
9	Genotype percentage change	Numeric	MM, TF	The ratio of the genotype effect for a sex relative to the wild-type signal for that variable for that sex. Format: "Female: *value*, Male: *value*". If sex is not present in the dataset value is set to "NA".
10	Sex estimate	Numeric	MM, TF, LR	Estimated coefficient that describes sex value calculated.
11	Sex standard error	Numeric	MM, TF, LR	Standard error of the coefficient estimate for sex.
12	Sex p-val	Numeric	MM, TF, LR	Sex p-value that represents the probability the sex is NOT relevant.
13	Weight estimate	Numeric	MM, TF	Estimated coefficient that describes weight value calculated.
14	Weight standard error	Numeric	MM, TF	Standard error of the coefficient estimate for weight.
15	Weight p-val	Numeric	MM, TF	Weight p-value that represents the probability the weight is NOT relevant.
16	Gp1 genotype	String	MM, TF, RR, FE, LR	Value of reference genotype.
17	Gp1 Residuals normality test	Numeric	MM, TF	p-value that represents the probability the residuals in reference genotype subset are normally distributed.
18	Gp2 genotype	String	MM, TF, RR, FE, LR	Value of test genotype.
19	Gp2 Residuals normality test	Numeric	MM, TF	p-value that represents the probability the residuals in test genotype subset are normally distributed.
20	Blups test	Numeric	MM, TF	p-value that represents the probability the blups are normally distributed.

75

#	Name	Value	Framework	Description
21	Rotated residuals normality test	Numeric	MM, TF	p-value that represents the probability the rotated residuals are normally distributed.
22	Intercept estimate	Numeric	MM, TF, LR	Estimated coefficient that describes intercept value.
23	Intercept standard error	Numeric	MM, TF, LR	Standard error of the coefficient estimate for intercept.
24	Interaction included	TRUE/FALSE	MM, TF, LR	Indicates the inclusion of genotype by sex interaction effect into model.
25	Interaction p-val	Numeric	MM, TF, LR	Interaction p-value that represents the probability the interaction is NOT relevant in model.
26	Sex FvKO estimate	Numeric	MM, TF, RR, FE, LR	Estimated coefficient that describes value calculated by the MM, TF and LR for females in test genotype subset or effect size estimate calculated from chi table for females only subset in FE and RR. In case of RR framework there are two values reported: low, high classification estimate.
27	Sex FvKO standard error	Numeric	MM, TF, LR	Standard error of the "Sex FvKO estimate".
28	Sex FvKO p-val	Numeric	MM, TF, RR, FE, LR	p-value that represents the probability that females group from test genotype subset contribution into model is NOT relevant. In case of RR framework there are two values reported: low, high classification p-value.
29	Sex MvKO estimate	Numeric	MM, TF, RR, FE, LR	Estimated coefficient that describes value calculated by the MM, TF and LR for males in test genotype subset or effect size estimate calculated from chi table for males only subset in FE and RR. In case of RR framework there are two values reported: low, high classification estimate.
30	Sex MvKO standard error	Numeric	MM, TF, LR	Standard error of the "Sex MvKO estimate".
31	Sex MvKO p-val	Numeric	MM, TF, RR, FE, LR	p-value that represents the probability that males group from test genotype subset contribution into model is NOT relevant. In case of RR framework there are two values reported: low, high classification p-value.
32	Classification tag	String	MM, TF, RR, FE, LR	A sexual dimorphism assessment of the phenotypic change from the results of MM, TF, LR or the overall estimation of the signals significance in FE and RR.
33	Transformation	String	MM, TF	Transformation parameters in format: lambda=*value*, scaleShift=*value*. If transformation was not performed or needed values are NA.
34	Additional information	String	MM, TF, RR	Additional info concerning dataset, for instance, subsets sizes. String in JSON format. See table 10 for "RR".

Table 9: Vector output description.

Tag name	Sub-tag	Output example	Meaning
Natural variation	-	95	Percentage of data classed as 'normal'
Min control points	-	60	Minimum number of data points required to construct reference range
Normal values	males only	19.19875 to 27.12625	The thresholds used to define normal
	females only	15.786 to 21.520	
	all	15.786 to 21.520	only provided if only one sex tested

Table 10: The "RR" framework's information stored within the additional information index in the vector output.

Additional information for "RR" framework contains data that help to understand how the RR methodology was implemented. For example the thresholds that define normal are stored within the additional information field. See table 10 for details.

Example of *vectorOutput* results for "RR" framework:

```
> vectorOutput(result)
Method
"Reference Ranges Plus framework"
Dependent variable
"Lean.Mass"
Batch included
"NA"
Residual variances homogeneity
"NA"
Genotype contribution
"NA"
Genotype estimate
"59%,3%"
Genotype standard error
"NA"
Genotype p-Val
"0.0000,1.0000"
...
Sex FvKO estimate
"72%,3%"
```

```
Sex FvKO standard error
"NA"
Sex FvKO p-val
"0.0000,1.0000"
Sex MvKO estimate
"47%,3%"
Sex MvKO standard error
"NA"
Sex MvKO p-val
"0.0000,1.0000"
Classification tag
"With phenotype threshold value 0.01 - significant in males (Low), females (Low) and in combined dataset (Low)"
Additional information
"{\"Natural variation:95\",\"Min control points:60\",
\"Normal values 'males only':18.10925 to 30.0315\",
\"Normal values 'females only':14.38975 to 23.52675\"}"
```

8.3   Count Matrices in Vector Format

There is an additional function to support the FE and RR frameworks: *vectorOutputMatrices*. This function returns values from count matrices in the vector format. We've limited the number of levels for dependent variable to 10. In the vector, the first three positions represent: dependent variable, genotype level 1 (reference genotype) and genotype level 2 (test genotype). The next 10 positions are used to define the dependent variable levels. When there are less than 10 levels, "NA" value is used. The next 20 positions represent combined count matrix values. Thereafter the vector contains the males only count matrix values and females only count matrix values. Again "NA" is used when the values are not present. The positions are labelled with the group and level.

For the chi squared tables from example described in "Fisher Exact Test framework" subsection (see 4.4) 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
```

9   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.

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

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

9.1   Graphics for FE, RR and LR frameworks

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

```
> categoricalBarplot(result_cat)
```

The example of bar plot is shown in Fig. 15. This graph allows a visual
representation of the genotype effect for the variable of interest.
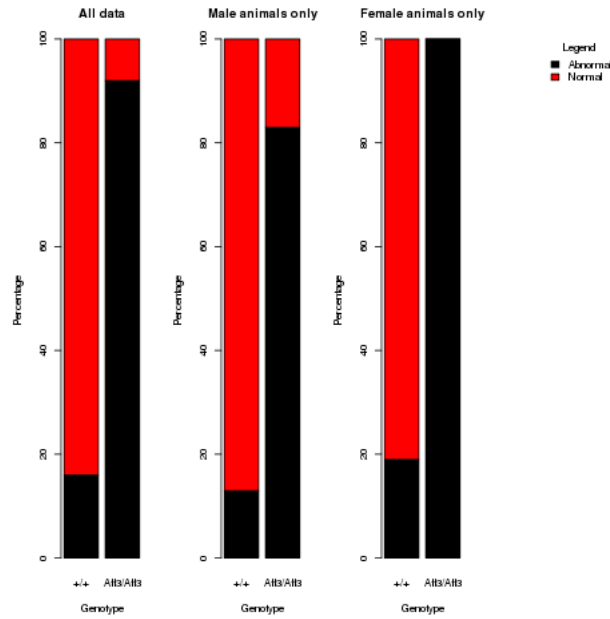
Figure 15: The PhenStat package's graphical output: categorical bar plot.

9.2 Graphics for MM and TF frameworks

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

9.2.1 Dataset based graphs

There are four functions in the dataset based graphs category:

- *boxplotSexGenotype* creates a box plot split by sex and genotype.

- *boxplotSexGenotypeBatchAdjusted* creates a box plot split by sex and genotype after the variation arising from batch has been accounted for as a random effect.

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

- *scatterplotGenotypeWeight* 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.

```
> boxplotSexGenotype(test,depVariable="Lean.Mass",graphingName="Lean Mass")
> boxplotSexGenotypeBatchAdjusted(test,depVariable="Lean.Mass",graphingName="Lean Mass Adjusted for Batch")
> scatterplotSexGenotypeBatch(test,depVariable="Lean.Mass",graphingName="Lean Mass")
> scatterplotGenotypeWeight(test,depVariable="Bone.Mineral.Content",graphingName="BMC")
```

An example of box plot split by sex and genotype is shown in Fig. 16. 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.
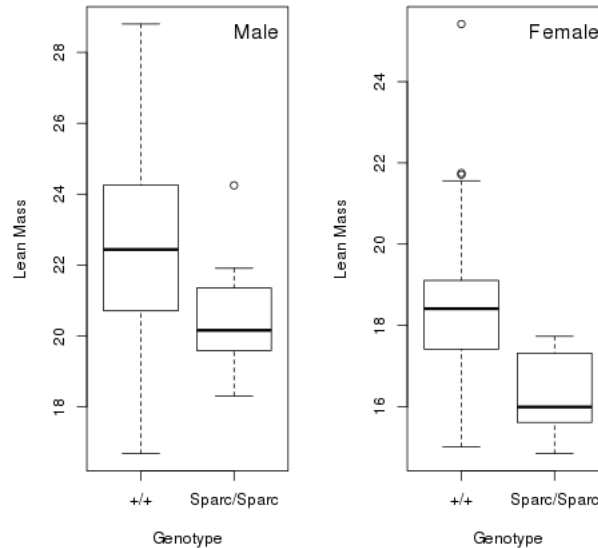
Figure 16: The PhenStat package's graphical output: box plot split by sex and genotype.

The example of scatter plot split by sex, genotype and batch is shown in Fig. 17. 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 compared relative to natural variation.

The example of scatter plot of body weight versus dependent variable is shown in Fig. 18. When weight is included in the model MM framework, it
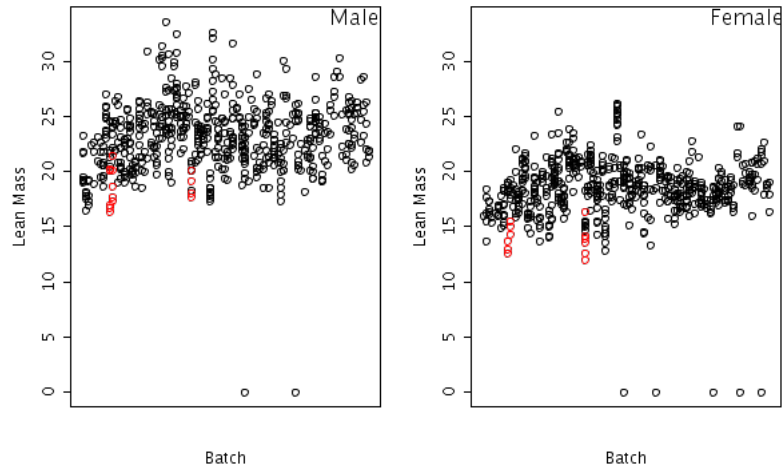
Figure 17: The PhenStat package's graphical output: scatter plot split by sex, genotype and batch.

assumes a linear relationship between dependent variable and body weight. This graph allows an assessment of this assumption.
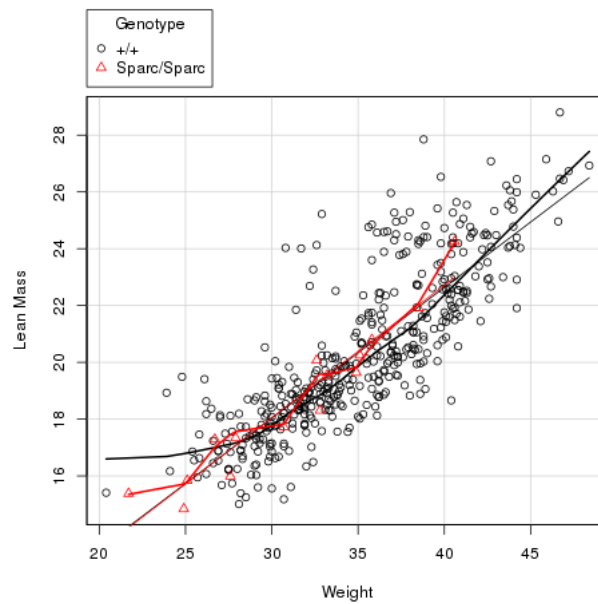
Figure 18: The PhenStat package's graphical output: scatter plot of body weight versus dependent variable.

9.2.2 Adjustment for batch

In order to demonstrate the power of *boxplotSexGenotypeBatchAdjusted* function the following dataset has been used: provided by the Toronto Centre for Phenogenomics of blood plasma calcium data obtained from a study on gene knockout mice carrying the *Tox3* *tm1b(KOMP)Mbp* targeted allele which were created by blastocyst injection of targeted ES cells, and bred on the C57BL/6NCrl genetic background. Data was collected on a standardized high throughput phenotyping pipeline with regular control animals.

A statistically significant genotype effect was reported from a mixed model analysis of the data (p-value = 6.836e-7) with an estimated effect size of -0.3535 ± 007 mg/dl affecting both sexes equally. However, visually inspection of the raw data suggests that no genotype effect is present (Figure 19:A). The conflict arises as the batch differences are masking the genotype effect in the raw data resulting in a boxplot of the raw data where no genotype effect can be observed. It is a strong example of why an analysis that removes batch effects (as the mixed model method does) is needed. The *boxplotSexGenotypeBatchAdjusted* function provides a graphing of the data after modelling has accounted for batch as a random effect and with this approach the genotype effect is then revealed (Figure 19:B). It is worth noting that when the graph is plotted with *scatterplotSexGenotypeBatch* function (Figure 20) where data is plotted by batch with close inspection the genotype effect can be seen without adjustment.

```
> data <- system.file("extdata", "Tox3CaDataset.csv", package="PhenStat")
> test <- PhenList(dataset=data, testGenotype="heterozygote",
 refGenotype="null",dataset.colname.batch="dateOfExperiment",
 dataset.colname.genotype="zygosity", dataset.colname.sex="sex")
> result <- testDataset(phenList=test, depVariable="dataPoint", equation="withoutWeight", method="MM")
> summaryOutput(result)
> boxplotSexGenotype(test, "dataPoint", "Calcium (mg/dl)")
> boxplotSexGenotypeBatchAdjusted(test, "dataPoint", "Calcium (mg/dl) adjusted for batch")
> scatterplotSexGenotypeBatch(test, "dataPoint", "Calcium (mg/dl)")
```

9.2.3 Results based graphs

There are five functions in the results based graphs category:

- *qqplotGenotype* creates a Q-Q plot of residuals for each genotype.

- *qqplotRandomEffects* creates a Q-Q plot of blups (best linear unbiased predictions). Only relevant in MM framework.
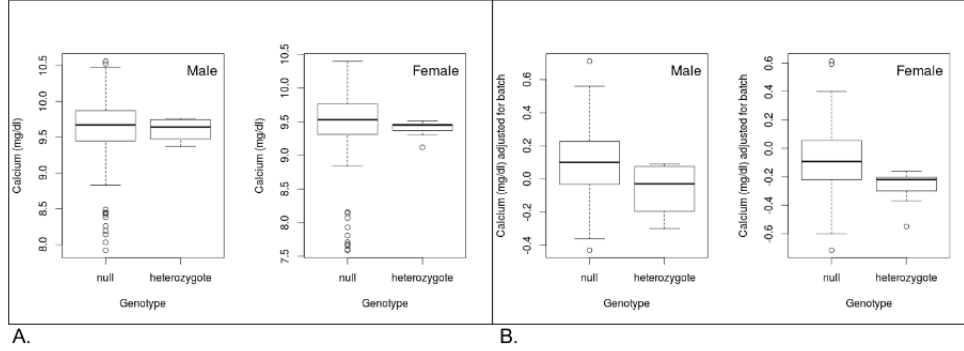
Figure 19: Exploration of the variation in calcium with genotype. Data shown is the calcium data for the *Tox3* knockout provided by Toronto Centre for Phenogenomics. **A:** *boxplotSexGenotype* function is showing the variation in calcium by sex and genotype. **B:** *boxplotSexGenotypeBatchAdjusted* function is showing the variation in calcium by sex and genotype after removing the effect of batch by modelling batch as a random effect.
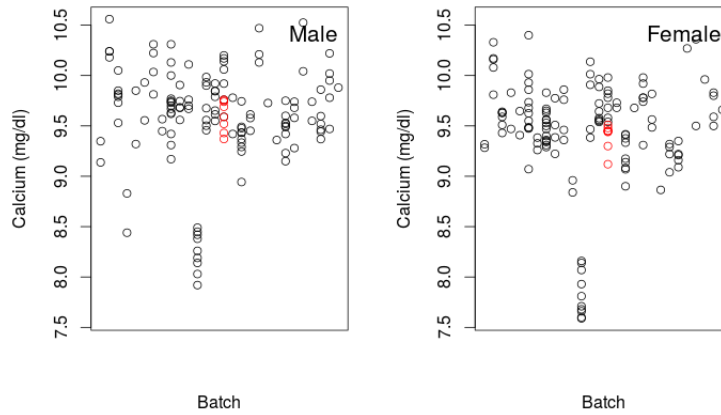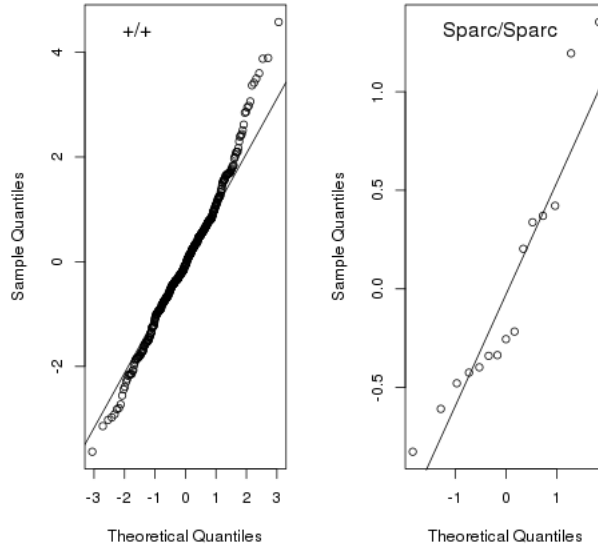

Figure 20: Exploration of the variation in calcium with genotype by batch using the function *scatterplotSexGenotypeBatch*. Data shown is the calcium data for the *Tox3* knockout provided by Toronto Centre for Phenogenomics.

Figure 21: The PhenStat package's graphical output: Q-Q plot of residuals for each genotype.

- *qqplotRotatedResiduals* creates a Q-Q plot of "rotated" residuals. Only relevant in MM framework.

- *plotResidualPredicted* creates predicted versus residual values plots split by genotype.

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

```
> qqplotGenotype(result)
# MM framework specific graphic function
> qqplotRandomEffects(result)
# MM framework specific graphic function
> qqplotRotatedResiduals(result)
> plotResidualPredicted(result)
> boxplotResidualBatch(result)
```

The example of Q-Q plot of residuals for each genotype is shown in Fig. 21. 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. A Q-Q plot assesses this assumption (residuals will be randomly arranged around the line if normally distributed).

86

The example of Q-Q plot of BLUPs (best linear unbiased predictions) is shown in Fig. 22. The MM assumes the BLUPs are normally distributed. This graph assesses 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 random batch effects. As such this is specific to MM framework only. See tutorial BLUPs for more details.
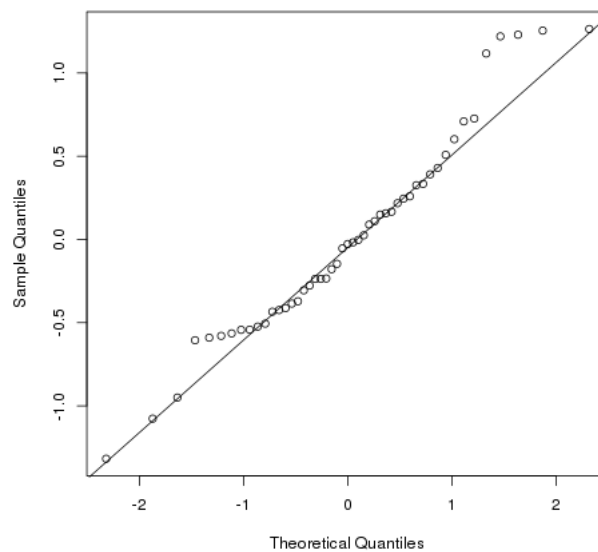
Figure 22: The PhenStat package's graphical output: Q-Q plot of BLUPs (best linear unbiased predictions).

Another method to assess the model fit of MM framework is to consider the normality of the "rotated" and "unrotated" residuals. The example of Q-Q plot of "rotated" residuals is shown in Fig. 23. See section 4.2.4 for the details about "rotated" residuals.

The example of residual-by-predicted plot is shown in Fig. 24. Residuals, differences between fitted and real values, are plotted against the predicted (fitted) values of dependent variable. A residual-by-predicted plot can be used to diagnose nonlinearity or nonconstant error variance. It is also can be used to find outliers.

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

87
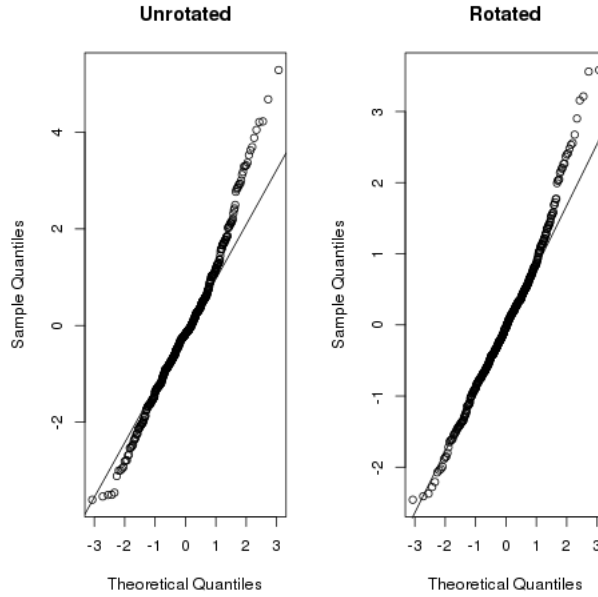
Figure 23: The PhenStat package's graphical output: Q-Q plot of "rotated" residuals.

- 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.

The example of box plot with residue versus batch split by genotype is shown in Fig. 25. This allows assessment that the residual behaviour for all batches is within natural deviation and the model is fitting the data well.

10   Case Studies

10.1   PhenStat MM framework Usage Example

The following dataset, provided by Wellcome Trust Sanger Institute (WTSI) Mouse Genetics Project (MGP) is Dual-energy X-ray absorptiometry data
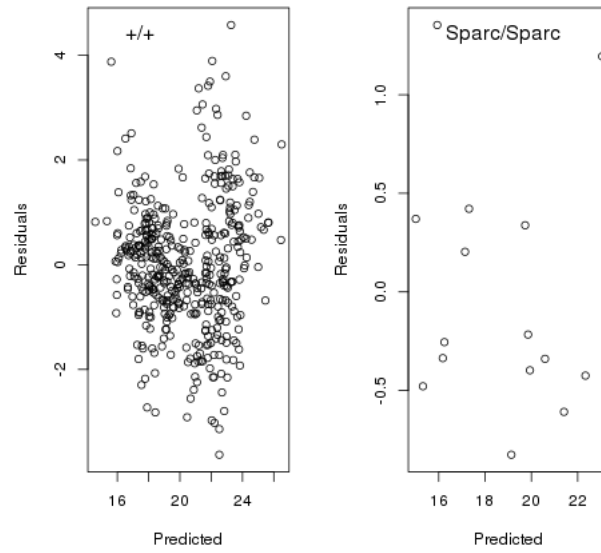
Figure 24: The PhenStat package's graphical output: residual-by-predicted plot split by genotype.
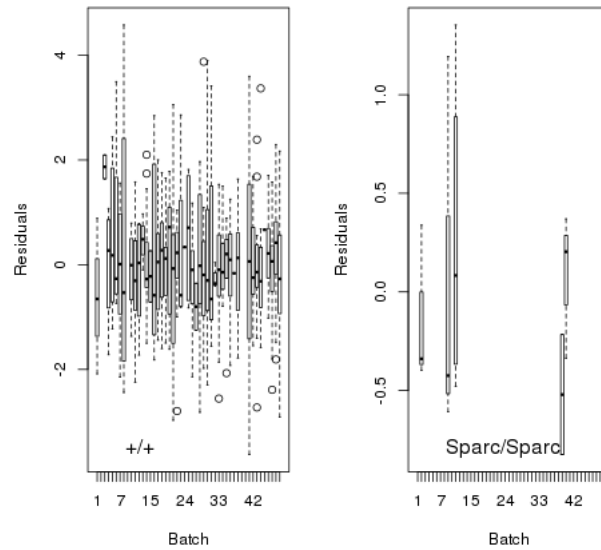
Figure 25: The PhenStat package's graphical output: box plot with residual versus batch split by genotype.

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 129S5/SvEvBrd genetic background. Data were collected
doing a standardized high throughput phenotyping pipeline following a multi-
batch workflow, where regular control animals are collected and knockout
animals of the correct age are issued to the pipeline as they arise.

Genotype	Sex	Number Animals	Number Batches
Akt2/Akt2	Female	12	3
	Male	14	3
+/+	Female	574	96
	Male	572	97

Table 11: Number of animals and number of batches in the *Akt2* dataset

10.1.1 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.

```
> file <- system.file("extdata", "test_Akt2.csv", package="PhenStat")
> DEXAdata <- read.csv(file)
> 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",transformValues=FALSE)
```

10.1.2 Exploring and understanding the summary output

The first two lines of the *summaryOutput* confirm the statistical framework
used and the dependent variable studied. If the values of dependent vari-
able have been transformed (see section 5) then information about trans-
formation parameters appears together with dependent variable name. For
example:

```
> resultTransformed <- testDataset(test,depVariable="Lean.Mass",
    method="MM",equation="withoutWeight",transformValues=TRUE)
> summaryOutput(resultTransformed)

Test for dependent variable:
```

90

```
*** Lean.Mass, power transformed with lambda value = 1.7  and scale shift = 1 ***


Method:
*** Mixed Model framework ***


--------------------------------------------------------------------------
Model Output
--------------------------------------------------------------------------
Final fitted model: Lean.Mass ~ Genotype + Sex
Was batch significant? TRUE
Was variance equal? FALSE
Genotype p-value: 2.661943e-10
Genotype effect (original scale): -10.5581 +/- 2.2910
Was there evidence of sexual dimorphism? no (p-value 1.467981e-01)
Genotype percentage change Female: -30%
Genotype percentage change Male: -21.01%


--------------------------------------------------------------------------
Classification Tag
--------------------------------------------------------------------------
With phenotype threshold value 0.01 - both sexes equally


--------------------------------------------------------------------------
Model Output Summary
--------------------------------------------------------------------------
                    Value Std.Error   DF  t-value       p-value
(Intercept)      92.95823  1.656484 1048 56.11778 4.921042e-318
GenotypeAkt2/Akt2 -27.88930  3.868329 1048 -7.20965  1.072359e-12
SexMale          39.79585  1.243963 1048 31.99118 3.071828e-157
```

Lean Mass values have been transformed with parameter $\lambda = 1.7$ and parameter $scaleShift = 1$. In the form of equation:

$$\text{Lean.Mass}_i = \frac{(\text{Lean.Mass\_original}_i + 1)^{1.7} - 1}{1.7}, \text{for } i = 1,..,1172$$

The function's *testDataset* output is *PhenTestResult* object that contains slot called *analysedDataset* from which original values (here "Lean.Mass_original"), transformed values (here "Lean.Mass") and batch adjusted (here "Lean.Mass_adjusted") values can be obtained:

```
> head(resultTransformed@analysedDataset)
  Lean.Mass    Sex  Genotype             Batch Weight Lean.Mass_original Lean.Mass_adjusted
1  45.34405 Female Akt2/Akt2 2008-10-22T23:00:00Z   14.8              11.98          -5.085269
2  49.13590 Female Akt2/Akt2 2008-03-18T00:00:00Z   17.1              12.60          -6.152621
3  51.26762 Female Akt2/Akt2 2008-03-18T00:00:00Z   17.2              12.94          -5.812621
4  65.58713 Female Akt2/Akt2 2008-03-20T00:00:00Z   18.4              15.09          -4.509509
5  60.56454 Female Akt2/Akt2 2008-03-20T00:00:00Z   18.5              14.36          -5.239509
6  55.24919 Female Akt2/Akt2 2008-10-22T23:00:00Z   18.7              13.56          -3.505269
```

91

Coming back to the output of *summaryOutput* function, the next section of it shows the final fitted model and clarifies model's 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 ("Was batch significant? TRUE") 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 "Genotype p-value" 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 $\sim$ Sex. Looking at the output the "Genotype p-value" is highly statistical significant as a $2.661943e-10$ value is reported.

After transformation the model estimates are on a new scale and hence for interpretability it is useful to convert these values back to the original scale. In the *summaryOutput* function, the genotype estimate and standard error is converted and this is reported as "Genotype effect (original scale)". In our example the genotype effect is estimated at -10.5581 ± 2.2910.

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 $= 0.147$) hence the final model assessed a genotype effect rather than the genotypes for each sex separately.

The next two lines of the output show percentage changes – the ratio of the genotype effect for a sex relative to the wildtype signal for that variable for that sex (see section 4.2.7 for more details).

Genotype percentage change Female $= (-27.89/92.96) * 100 = -30\%$

Genotype percentage change Male $= (-27.89/(92.96+39.8))*100 = -21.01\%$

As there was no evidence for sexual dimorphism, and thus the genotype effect was estimated independently of sex, then a classification tag "both sexes equally" has been assigned.

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.

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 wildtype animals) is 92.96. Then we can see the effect of the genotype is -27.89 (-10.56 in original scale) in that knockout animals are predicted to have 27.89 grams (10.56 in original scale) less lean mass relative to the control animals. Finally we can see that being a male animals leads to an increase of 39.8 grams (11.03 grams in original scale). These estimates are in agreement with the visualisation of the data (see section 10.1.3).

Here is the function example for reverse transformation in order to obtain original scale values:

```
> reverseTransformValues(c(-27.89,39.8),lambda = 1.7,scaleShift = 1)
[1] -10.55820  11.03411
```

Function's *summaryOutput* results for the not transformed lean mass values:

```
> summaryOutput(result)

Test for dependent variable:
*** Lean.Mass ***

Method:
*** Mixed Model framework ***


-------------------------------------------------------------------------
Model Output
-------------------------------------------------------------------------
Final fitted model: Lean.Mass ~ Genotype + Sex
Was batch significant? TRUE
Was variance equal? FALSE
Genotype p-value: 1.901362e-10
Genotype effect: -3.5271 +/- 0.4866
Was there evidence of sexual dimorphism? no (p-value 9.055876e-01)
Genotype percentage change Female: -19.04%
Genotype percentage change Male: -15.24%


-------------------------------------------------------------------------
Classification Tag
-------------------------------------------------------------------------
```

```
With phenotype threshold value 0.01 - both sexes equally


----------------------------------------------------------------------------
Model Output Summary
----------------------------------------------------------------------------

                    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
SexMale           4.620361 0.1581236 1048 29.219943 8.940893e-138
```

10.1.3   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, *boxplotSexGenotype*, allows the genotype effect to be visualised for each sex group. For the *Akt2* example, we can suggest that there looks to be a significant genotype effect that is seen in both sexes.

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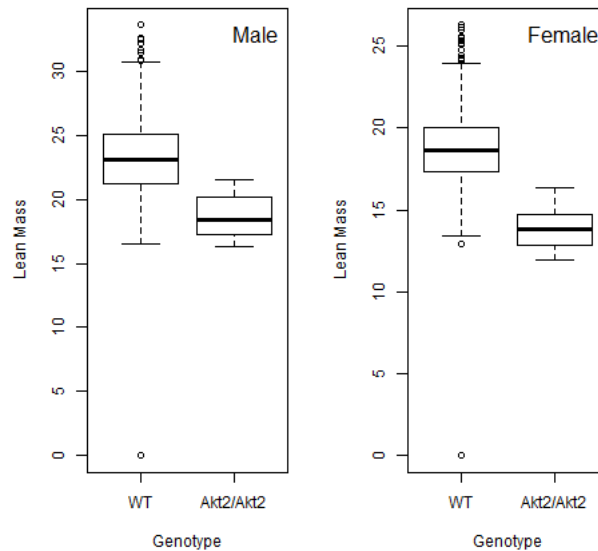

Figure 26: *boxplotSexGenotype* function for *Akt2* example.

94

A second function, *scatterplotSexGenotypeBatch*, allows the comparison between genotype as a function of batch and sex. 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.

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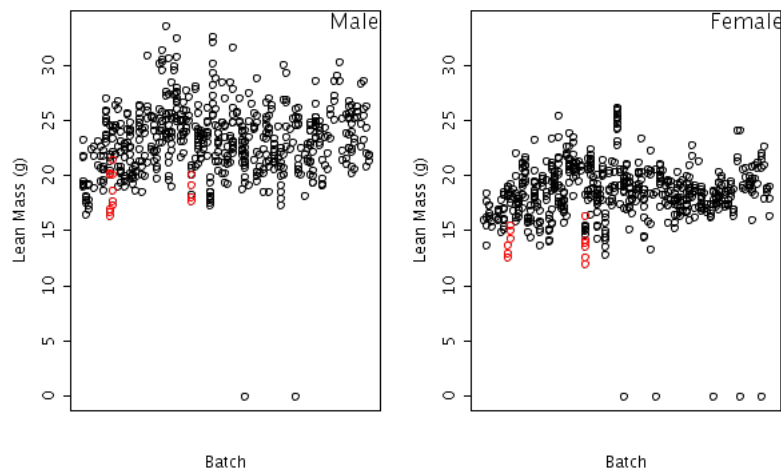

Figure 27: *scatterplotSexGenotypeBatch* function for *Akt2* example.

10.1.4   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 wildtype, residuals for the knockout, the blups and "rotated" residuals (see section 4.2.4). 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 wildtype 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 homozygous knockout 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)
  ```
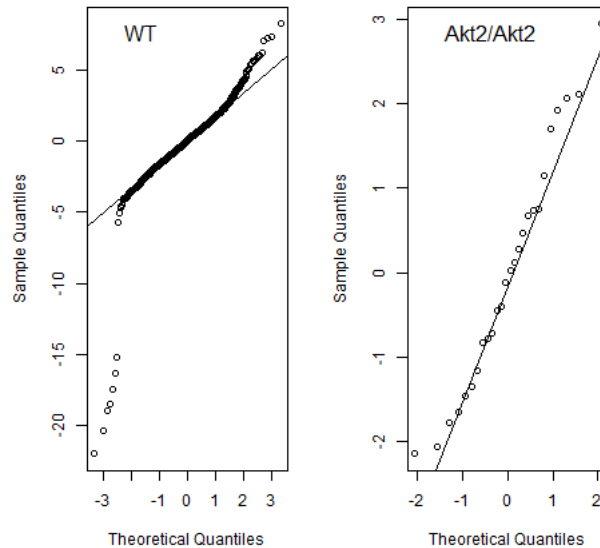
Figure 28: *qqplotGenotype* function for *Akt2* example.

- **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 wildtype 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 wildtype group.

  A few of the wildtype (control) dataset points have large residuals and it would be worth looking at these data points further to see why these are outliers. They do not suggest the model should be discarded because as a proportion of the dataset they are few and scattered through the dataset.

  ```
  > boxplotResidualBatch(result)
  ```

97
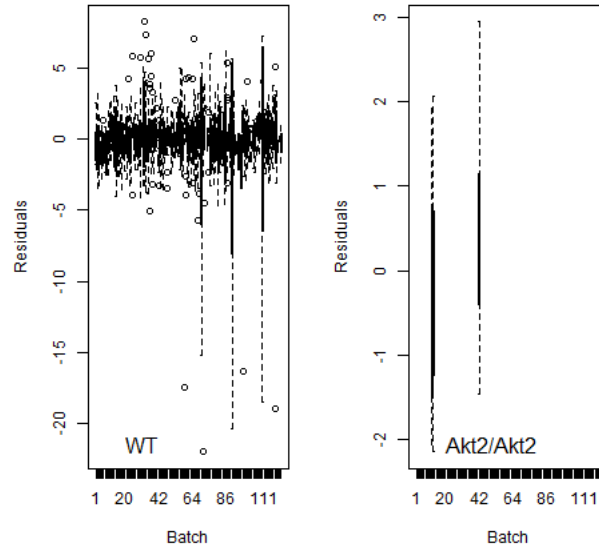
Figure 29: *boxplotResidualBatch* function for *Akt2* example.

- **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)
    ```
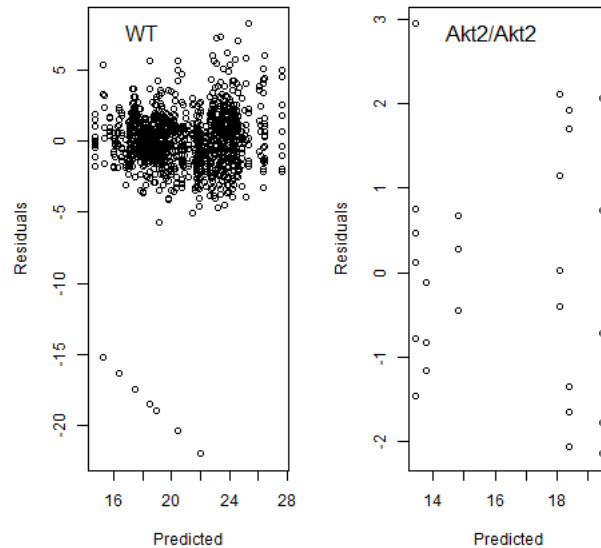
Figure 30: *plotResidualPredicted* function for *Akt2* example.

- **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)
```

99
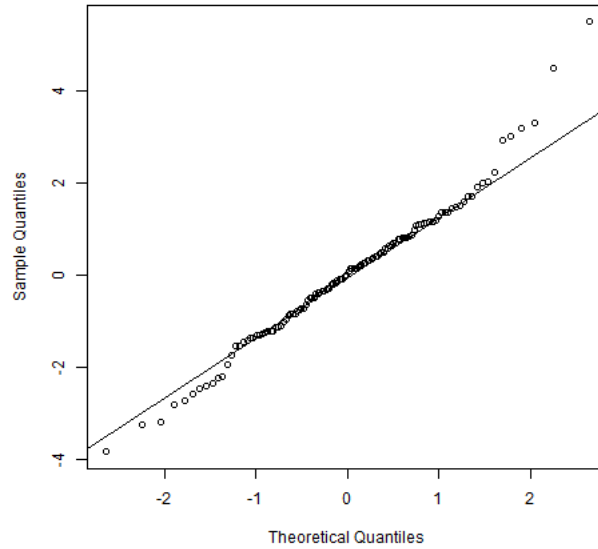
Figure 31: *qqplotRandomEffects* function for *Akt2* example.

- **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 4.2.4 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)
```
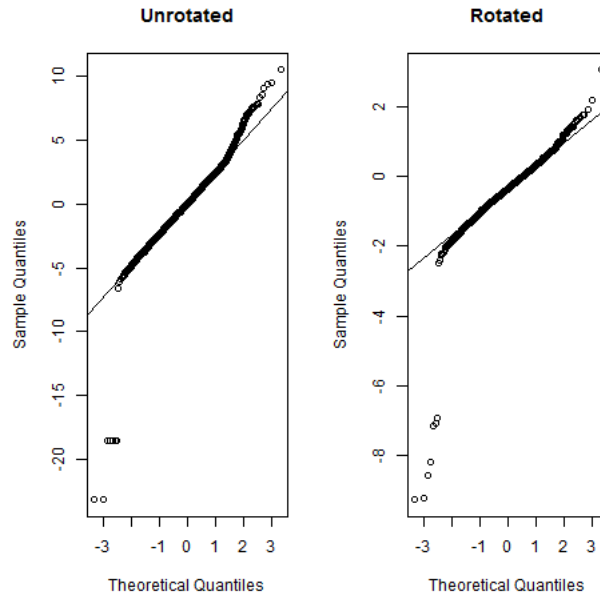
Figure 32: *qqplotRotatedResiduals* function for *Akt2* example.

10.1.5 Including weight as a covariate in the model fitting process

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 wildtype 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 (MGP) have a relationship with body weight. Looking at the *Akt2* dataset we can see that there is a difference in body weight between the wildtype and knockout group and thus body weight can be a confounding factor to isolating the genotype effect.

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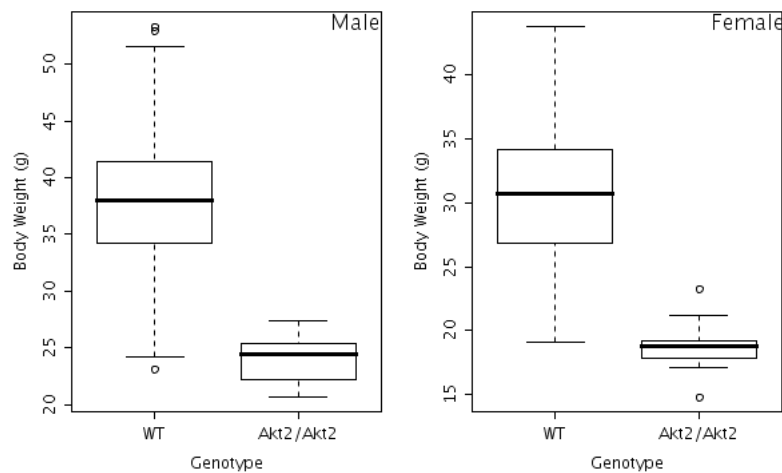

Figure 33: *boxplotSexGenotype* function for *Akt2* example to show the body weight impact.

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

102

lean mass value of a wildtype female animal to the lean mass value of a wildtype 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 10.1.2), 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.

```
> result <- testDataset(test,depVariable="Lean.Mass",
    method="MM",equation="withWeight",transformValues=FALSE)
> summaryOutput(result)

Test for dependent variable:
*** Lean.Mass ***

Method:
*** Mixed Model framework ***

---------------------------------------------------------------------------
Model Output
---------------------------------------------------------------------------
Final fitted model: Lean.Mass ~ Genotype + Sex + Weight
Was batch significant? TRUE
Was variance equal? FALSE
Genotype p-value: 6.399898e-01
Genotype effect: -0.1925 +/- 0.4120
Was there evidence of sexual dimorphism? no (p-value 1.053833e-01)
Genotype percentage change Female: -1.04%
Genotype percentage change Male: -0.83%


---------------------------------------------------------------------------
Classification Tag
---------------------------------------------------------------------------
With phenotype threshold value 0.01 - no significant change


---------------------------------------------------------------------------
Model Output Summary
---------------------------------------------------------------------------
                   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
SexMale           2.1550868 0.17386173 1047 12.3954067 5.056608e-33
Weight            0.3541522 0.01623164 1047 21.8186332 2.718321e-87
```

10.1.6 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")
```
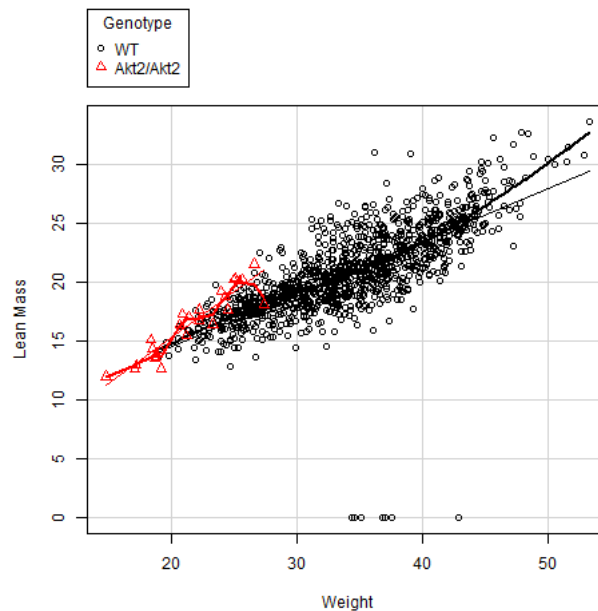
Figure 34: *scatterplotGenotypeWeight* function for *Akt2* example.

10.2 PhenStat TF framework Usage Example

The following dataset, provided by the German Mouse Clinic from a plasma chemistry study on gene knockout mice carrying the *Dbn1* targeted allele on the C57BL/6NTac(USA) genetic background. Data was collected via a standardised high throughput phenotyping pipeline following a multi-batch workflow, with concurrent controls. The dataset contains baseline controls of which a proportion is concurrent.

Genotype	Sex	Number Animals	Number Batches
Het	Female	9	3
	Male	10	3
WT(concurrent controls)	Female	6	1
	Male	10	3
WT(baseline controls)	Female	189	NA
	Male	218	NA

Table 12: Number of animals and number of batches in the *Dbn1* dataset

10.2.1 Loading the data and initial steps of analysis

Initial steps focus on loading the data ready for statistical analysis. For the TF framework there are two steps; first the use of PhenList and a second specific function to the TF framework using the *TFDataset* function to generate a dataset cleaned to remove control data that isn't concurrent. The resulting cleaned data can be analysed and the results explored using the visualisation and output functions.

```
>file <- system.file("extdata", "test7_TFE.csv", package="PhenStat")

>test <- PhenList(dataset=read.csv(file), refGenotype="WT", testGenotype="het",
dataset.colname.sex="sex", dataset.values.male="m",dataset.values.female="f",
dataset.colname.batch="Date_of_procedure_start",dataset.colname.weight="body.weight")
...
>testTF_new <- TFDataset(test,depVariable="HDL")
Data points containing 'HDL' by batch levels:
| ---------- | ---------- | ---------- | ---------- | ---------- |
|            |         WT |         WT |        het |        het |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|      Batch |     Female |       Male |     Female |       Male |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 02.09.2013 |          7 |          4 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
```

```
| * 03.02.2014 |          0 |          1 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 03.03.2014 |          6 |          6 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
...
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 14.01.2014 |          5 |          7 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|   14.04.2014 |          0 |          4 |          4 |          3 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 14.10.2013 |          7 |          5 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 15.07.2013 |          5 |          5 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 16.06.2014 |          8 |          8 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 16.09.2013 |          2 |          3 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|   17.02.2014 |          6 |          4 |          3 |          4 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 17.03.2014 |          4 |          5 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
|   18.02.2014 |          0 |          3 |          2 |          3 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 18.03.2014 |          4 |          3 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
...
| ---------- | ---------- | ---------- | ---------- | ---------- |
| * 31.03.2014 |          3 |          3 |          0 |          0 |
| ---------- | ---------- | ---------- | ---------- | ---------- |
* - removed record(s)

Number of batch levels left: 3
Records removed (reference genotype): 92%
Records removed (test genotype): 0%

>result <- testDataset(testTF_new, depVariable="HDL", dataPointsThreshold=2, method="TF",
transformValues=FALSE)
...
```

The cleaning of the data is reported in the output of the *TFDataset* function. First the output shows the number of data points by batch for each sex and genotype combination. For data to be retained for a date, there needs to be both test and ref genotype data for a sex. Summary statistics on the cleaning impact are then provided following the table of data.

10.2.2 Exploring and understanding the summary 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 TF 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 model including batch as a fixed effect was fitted. 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.

Furthermore, we can see that there was statistical evidence of sexual dimorphism and therefore the final model would estimate the genotype effect for each sex separately. The next stage of the output, classification tag, details the classification of the effect seen as discussed in section 4.3.5 and in this case the genotype effect was specific to the males at the default (0.01) threshold.

```
>summaryOutput(result)


Test for dependent variable:
*** HDL ***

Method:
*** Time as Fixed Effect framework ***


----------------------------------------------------------------------------
Model Output
----------------------------------------------------------------------------
Final fitted model: HDL ~ Sex + Genotype:Sex + Weight + Batch
Was batch significant? TRUE
Was variance equal? FALSE
Genotype p-value: 1.072892e-04
Genotype by male effect: -0.3063 +/- 0.0781
Genotype by female effect: -0.0170 +/- 0.0856
Was there evidence of sexual dimorphism? yes (p-value 3.044927e-02)
Genotype percentage change Female: -1.44%
Genotype percentage change Male: -19.15%


----------------------------------------------------------------------------
Classification Tag
----------------------------------------------------------------------------
With phenotype threshold value 0.01 - males only
```

```
--------------------------------------------------------------------------
Model Output Summary
--------------------------------------------------------------------------

                            Value  Std.Error   t-value     p-value
(Intercept)              0.18543523 0.24670119  0.7516593 4.583132e-01
SexMale                  0.15246298 0.07485017  2.0369090 5.087890e-02
Weight                   0.05286366 0.01036426  5.1005716 1.917964e-05
Batch17.02.2014         -0.25235168 0.05892587 -4.2825277 1.848788e-04
Batch18.02.2014         -0.48135264 0.06644529 -7.2443452 5.617868e-08
SexFemale:Genotypehet   -0.01704878 0.08560398 -0.1991587 8.435284e-01
SexMale:Genotypehet     -0.30625701 0.07813579 -3.9195486 4.975257e-04
```

We can also see in this section the assessment of the "Genotype p-value" 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 $HDL \backsim Sex+Weight+Batch$. Looking at the output the "Genotype p-value" is statistical significant as a 0.0001 value is reported.

The percentage changes – the ratio of the genotype effect for a sex relative to the wildtype signal for that variable for that sex (see section 4.2.7 for more details) are estimated as -1.44% for females and -19.15% for males.

There was an evidence for sexual dimorphism, so according to the decision tree from Figure 7 the classification tag "males only" has been assigned.

The final section of the output "Model Output Summary" provides a summary table showing the estimates for each component of the final model fitted. As a regression model estimates each component of the model, by isolating how that effect influences the dependent variable (i.e. HDL) as though all other parts of the model had been fixed and isolated. From the summary output, we can see that the intercept (the expected reading for HDL) for female reference genotype (aka female wildtype animals of 0 body weight for the first batch) is 0.185. Then we can see that being male is statistical significant and leads to an increase in HDL of 0.152. We can also see the impact of weight 0.0528/g. As batch was significant source of variation, and it is treated as a fixed effect we can see the estimated impact of each batch on the readings. Finally, we can see the impact of being het male mouse where the values were lower by -0.3 units with standard error of 0.078. The lack of significance of the female het mice can be seen in that the estimated effect is -0.02 with an error term 0.09 which is larger than the estimated effect and

therefore this effect cannot be classed as significant.

10.2.3 Assessment of raw data and distribution characteristics

Graph of the knockout data against the baseline controls:

```
> boxplotSexGenotype(test, "HDL", "High Density Lipoprotein (mM)")
```
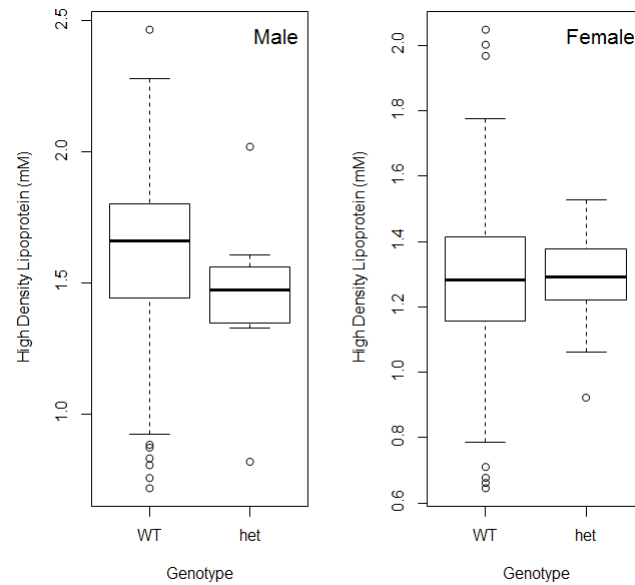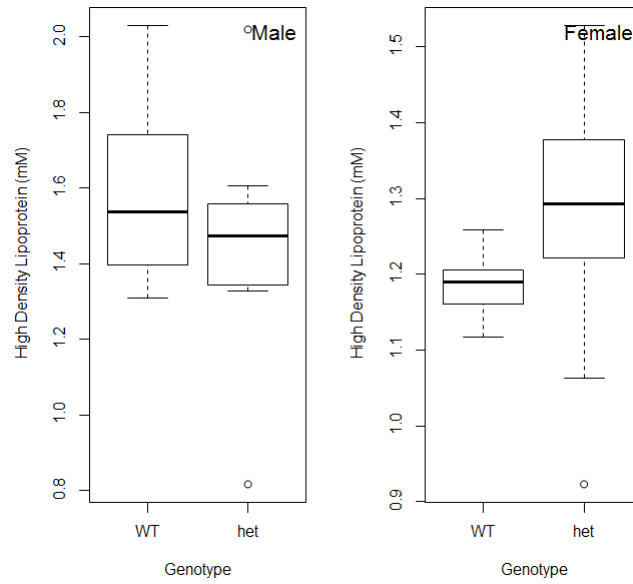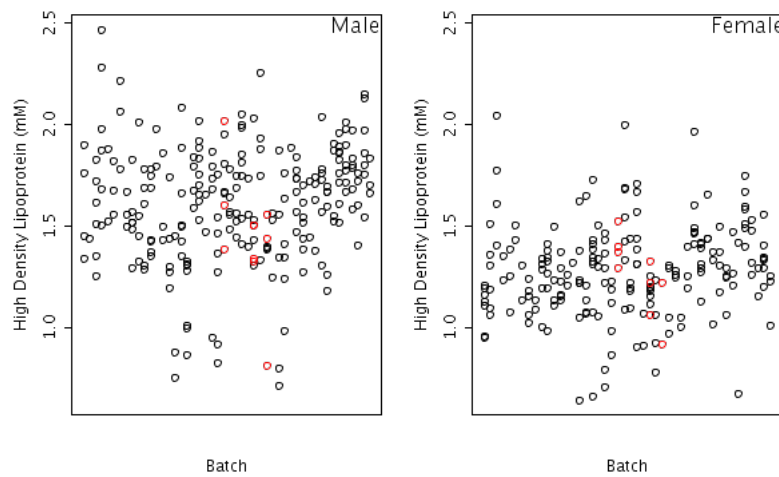

Figure 35: *boxplotSexGenotype* for the *Dbn1* High Density Lipoprotein example showing the baseline and concurrent controls.

Graph of the knockout versus the concurrent controls:

```
> boxplotSexGenotype(testTF_new, "HDL", "High Density Lipoprotein (mM)")
```

109

Figure 36: *boxplotSexGenotype* for the *Dbn1* High Density Lipoprotein example showing only the concurrent controls.

Graph showing batch variation for the baseline and concurrent controls:

```
> scatterplotSexGenotypeBatch(test, "HDL", "High Density Lipoprotein (mM)")
```
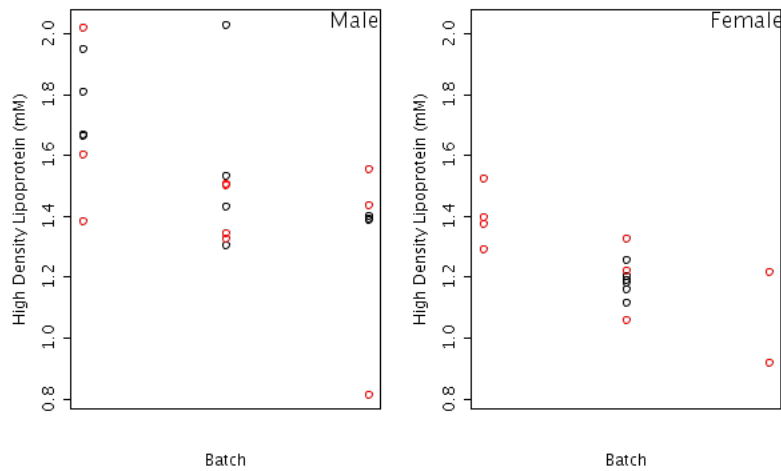

Figure 37: *scatterplotSexGenotypeBatch* for the *Dbn1* High Density Lipoprotein example showing the baseline and concurrent controls. The test genotype *Dbn1* data are highlighted with red colour.

110

Graph showing batch variation for the concurrent controls only:

```
> scatterplotSexGenotypeBatch(testTF_new, "HDL", "High Density Lipoprotein (mM)")
```


Figure 38: *scatterplotSexGenotypeBatch* for the *Dbn1* High Density Lipoprotein example showing the concurrent controls. The test genotype *Dbn1* data are highlighted with red colour.

10.2.4 Assessment of model fit

Three functions are available and they focus on looking at residual behaviour. The mixed model methodology has an additional two (*qqplotRandomEffects* and *qqplotRotatedResiduals*) which are not relevant to this framework as they assess the assumptions around batch when treated as a random effect. 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 wildtype and residuals for the knockout (see section 8.2). 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 wildtype arises from a high throughput program with a running baseline), the statistical test can be overall sensitive to departures from nor-

mality 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 *Dbn1* High Density Lipoprotein example, the residuals on the both the wildtype and heterozygous knockout group are near perfect showing the model is fitting this data well.

  ```
  > qqplotGenotype(result)
  ```
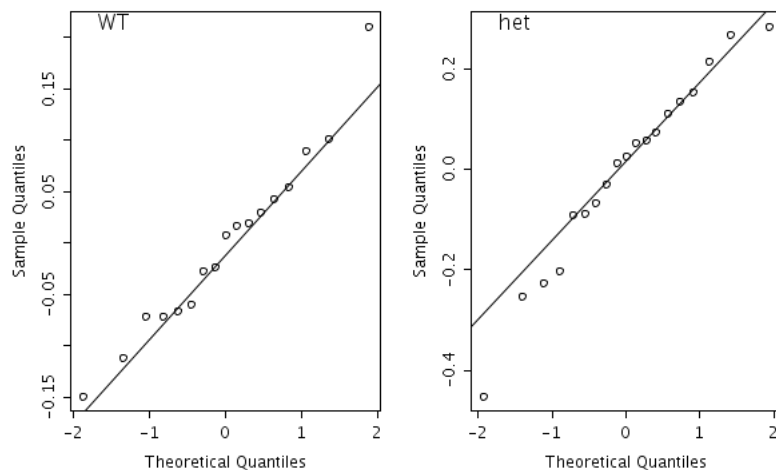

Figure 39: *qqplotGenotype* function for *Dbn1* High Density Lipoprotein example.

- **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 wildtype and knockout line. For the *Dbn1* High Density Lipoprotein example, we can see that the variation in

residual is consistent across all the batches. It is higher in the knock-out dataset which is why the final fitted model did not assume equal variance.
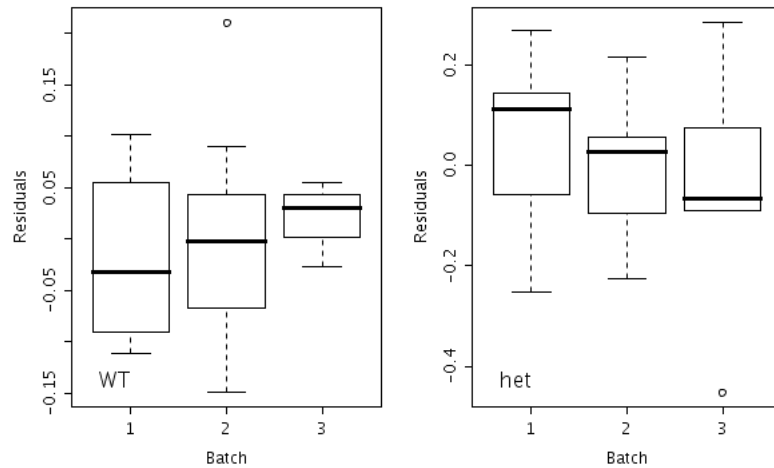
```
> boxplotResidualBatch(result)
```


Figure 40: *boxplotResidualBatch* function for *Dbn1* High Density Lipoprotein 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 *Dbn1* High Density Lipoprotein data, we can see that there spread of the residuals is fairly consistent, suggesting the model is a good fit for all the data points.

```
> plotResidualPredicted(result)
```
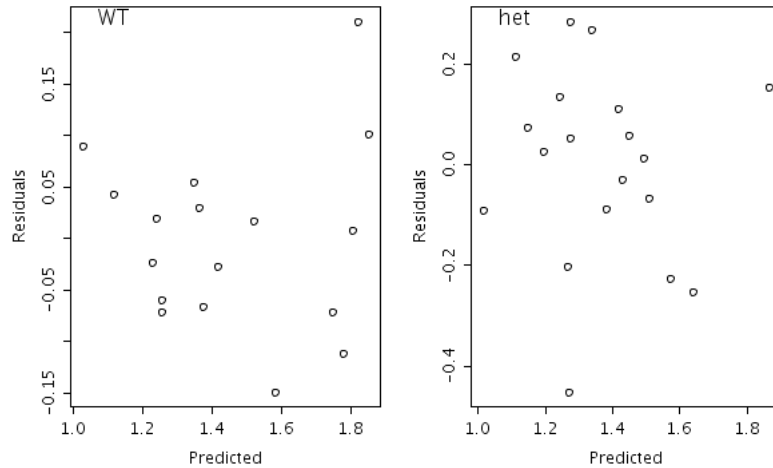
Figure 41: *plotResidualPredicted* function for *Dbn1* High Density Lipoprotein example.

10.3 PhenStat RR framework Usage Example

The following dataset, provided by Wellcome Trust Sanger Institute (WTSI) Mouse Genetics Project (MGP) is Dual-energy X-ray absorptiometry data obtained for a study on gene knockout mice carrying *Akt2<tm1Wcs>* targeted allele which were created by blastocyst injection of targeted ES cells, and bred on the 129S5/SvEvBrd genetic background. Data was collected doing a standardized high throughput phenotyping pipeline following a multi-batch workflow, where regular control animals are collected and knockout animals of the correct age are issued to the pipeline as they arise.

See Table 11.

10.3.1 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.

```
> file <- system.file("extdata", "test_Akt2.csv", package="PhenStat")
> DEXAdata <- read.csv(file)
> test <- PhenList(dataset=DEXAdata,
  testGenotype="Akt2/Akt2",
```

```
    refGenotype="WT",
    dataset.colname.batch="dateOfExperiment",
    dataset.values.female="female",
    dataset.values.male="male",
    dataset.colname.genotype="Genotype",
    dataset.colname.sex="Gender")
> result <- testDataset(test,
depVariable="Lean.Mass",
method="RR")
```

10.3.2 Visualisation of data

The function *categoricalBarplot* has been provided to visualise the categorical data formed from the RR framework 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*.
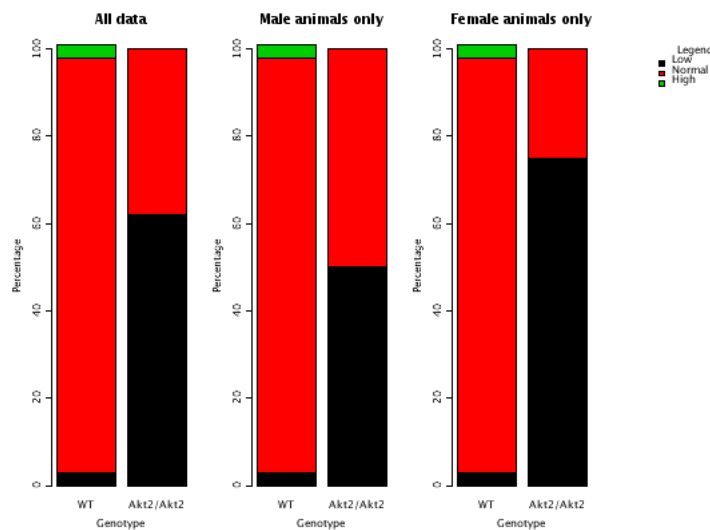
```
> categoricalBarplot(result)
```


Figure 42: *categoricalBarplot* function for *Akt2* Lean Mass example.

10.3.3 Understanding the summaryOutput

```
> summaryOutput(result)

Test for dependent variable:
*** Lean.Mass ***

Method:
*** Reference Ranges Plus framework ***

1) High vs Normal/Low
          All Females only Males only
p-value 1.0000       1.0000     1.0000
ES         3%           3%         3%

2) Low vs Normal/High
          All Females only Males only
p-value 0.0000       0.0000     0.0000
ES        59%          72%        47%


--------------------------------------------------------------------------
Classification Tag
--------------------------------------------------------------------------
With phenotype threshold value 0.01 - significant in males (Low), females (Low)
and in combined dataset (Low)


--------------------------------------------------------------------------
Thresholds
--------------------------------------------------------------------------
Natural variation:          95
Min control points:         60
Normal values 'males only':   18.10925 to 30.0315
Normal values 'females only': 14.38975 to 23.52675


--------------------------------------------------------------------------
Count Matrices
--------------------------------------------------------------------------
All
            WT Akt2/Akt2
Low          30      16
Normal/High 1116     10


All
            WT Akt2/Akt2
High         30       0
Normal/Low  1116     26


Females only
            WT Akt2/Akt2
```

```
Low          15         9
Normal/High 559         3


Males only
          WT Akt2/Akt2
Low          15         7
Normal/High 557         7


Females only
          WT Akt2/Akt2
High         15         0
Normal/Low 559         12


Males only
          WT Akt2/Akt2
High         15         0
Normal/Low 557         14
```

The first two lines of *summaryOutput* indicate the statistical framework used and the dependent variable studied.

The next section "Model Output" reports the summary table of statistical assessment for two classifications: high and low. Three datasets are assessed: All, Females only and Males only. Two measures are provided for each dataset considered:

1. A test of statistical significance (p-value) assessed using a Fisher Exact Test.

2. A measure of biological significance, the maximum effect size change (ES)

In the case of Reference Range framework classification tag means not sexual dimorphism classification, but rather the overall estimation of the signals significance across the different datasets.

The signal is significant in all three tested datasets and the direction of the signal is the same – Low, so classification tag "significant in males (Low), females (Low) and in combined dataset (Low)" is assigned (see Table 6).

The section titled 'Thresholds' indicates how the reference range methodology has been configured.

- "Natural variation" indicates how much of the data was used to define normal. This defaults to 95%.

- "Min control points" indicates how many data points were required as a minimum to build a reference range.

117

- The thresholds for defining normal are reported for the male and female for this variable. Data is analysed as a male only, a female only and all. In the all classification, the male and female classifications are combined but please note the classification of normal, low or high are run on a sex specific basis. In the situation where only one sex is present the output is returned in the all category.

The final section of the output provides tables showing the counts calculated for each group and possible level.

The following is reported for dependent variable lean mass for the *Akt2* dataset and we can see that for all datasets, there is a statistically significant change of the values for knockout animals towards low classification with a large effect size.

10.3.4 Understanding the maximum effect size reported

The effect size reported is the maximum percentage change seen in the low or high classification. 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 wildtype. Then across the low and high levels, the maximum percentage change is selected after ignoring the direction of the change.

```
> for (i in seq_along(analysisResults(result))) {
    val <- analysisResults(result)[[i]]
    if (analysedSubset(val)=="males"){
        print(getPercentageMatrix(val))
    }
 }
                 WT Akt2/Akt2
Low          2.622378       50
Normal/High 97.377622       50
                 WT Akt2/Akt2
High         2.622378        0
Normal/Low  97.377622      100
```

Thus in the *Akt2* example, the maximum effect size would be 47% for the male data as the increase in classification of low ($|3 - 50| = 47\%$) was larger than the change in the high classification ($|3 - 0| = 3\%$).

118

10.4 PhenStat FE framework Usage Example

The following dataset, provided by Wellcome Trust Sanger Institute (WTSI) Mouse Genetics Project (MGP) 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 standardized high throughput phenotyping pipeline following a multi-batch workflow, where regular control animals are collected and knockout animals of the correct age are then 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 wildtype population.

Genotype	Sex	Number Animals	Number Batches
Aff3/Aff3	Female	7	4
	Male	6	4
wildtype	Female	446	70
	Male	451	70

Table 13: Number of animals and number of batches in the *Aff3* dataset

10.4.1 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.

```
> file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> dataset_cat <- read.csv(file)
> test <- PhenList(dataset=dataset_cat,
    testGenotype="Aff3/Aff3",
    refGenotype="+/+",
    dataset.colname.batch="Assay.Date")
> result <- testDataset(test, depVariable="Thoracic.Processes", method="FE")
```

10.4.2 Visualisation of data

The *categoricalBarplot* function is used 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*.
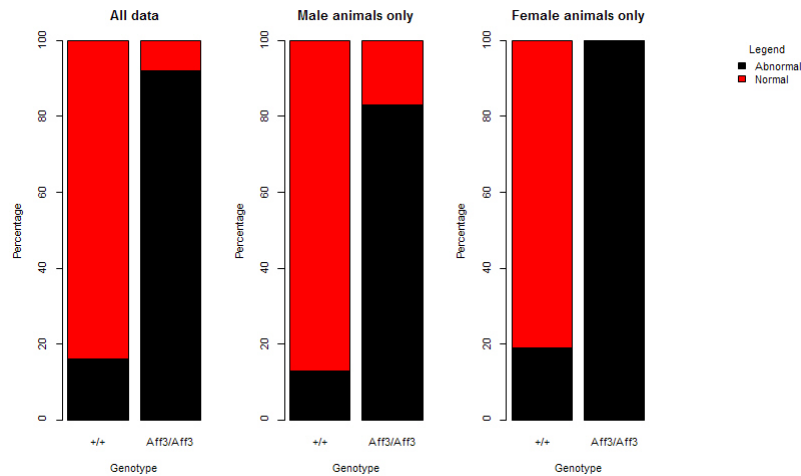
```
> categoricalBarplot(result)
```


Figure 43: *categoricalBarplot* function for *Aff3* example.

10.4.3 Understanding the summary output

The first two lines of *summaryOutput* indicate the dependent variable studied and the statistical framework used. The next section, titled "Model output", reports the summary of statistical assessment. Three datasets are assessed: All, Females only and Males only. Two measures are provided for each dataset considered:

1. A test of statistical significance (p-value) assessed using a Fisher Exact Test.

2. A measure of biological significance, the maximum effect size (ES)

In the case of Fisher Exact Test classification tag means not sexual dimorphism classification, but rather the overall estimation of the signals significance across the different datasets.

The signal is significant in all three tested datasets, so classification tag "significant in males, females and in combined dataset" is assigned (see Table 6).

The final section provides tables showing the counts calculated for each tested dataset.

```
> summaryOutput(result)

Test for dependent variable:
*** Thoracic.Processes ***

Method:
*** Fisher Exact Test framework ***


-------------------------------------------------------------------------
Model Output ('*' highlights results with p-values less than threshold 0.01)
-------------------------------------------------------------------------
All
* p-value:     0.0000
* Effect size: 76%

Males only
* p-value:     0.0003
* Effect size: 70%

Females only
* p-value:     0.0000
* Effect size: 81%


-------------------------------------------------------------------------
Classification Tag
-------------------------------------------------------------------------
With phenotype threshold value 0.01 - significant in males, females and in combined dataset


-------------------------------------------------------------------------
Count Matrices
-------------------------------------------------------------------------
All
         +/+ Aff3/Aff3
Abnormal 142        12
Normal   753         1

Males only
         +/+ Aff3/Aff3
```

```
Abnormal  59        5
Normal    390       1


Females only
          +/+ Aff3/Aff3
Abnormal  83        7
Normal    363       0
```

10.4.4 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 wildtype 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 wildtype. 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.

Trait level	wildtype		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 14: Number of animals and number of batches in the *Aff3* dataset

10.4.5 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 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.
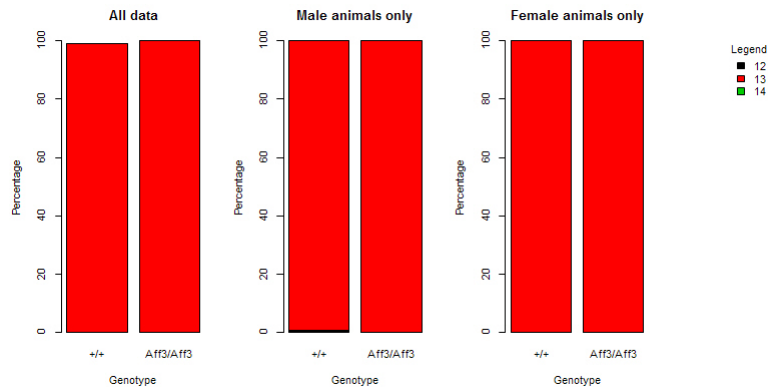
Figure 44: *categoricalBarplot* function for *Aff3* example with number of ribs as dependent variable.

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")
Warning:
Weight column is not present in dataset.
Equation 'withWeight' can't be used in such a case and has been replaced to 'withoutWeight'.

Error:
Insufficient variability in dependent variable 'Number.Of.Ribs.Left' for MM/TF framework.
Fisher Exact Test can be better way to do the analysis.
```

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:

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

Test for dependent variable:
*** Number.Of.Ribs.Left ***

Method:
*** Fisher Exact Test framework ***


-----------------------------------------------------------------------
Model Output ('*' highlights results with p-values less than threshold 0.01)
-----------------------------------------------------------------------
```

```
All
  p-value:     1.0000
  Effect size: 0%


Males only
  p-value:     1.0000
  Effect size: 0%


Females only
  p-value:     1.0000
  Effect size: 0%


----------------------------------------------------------------------
Classification Tag
----------------------------------------------------------------------
Not significant


----------------------------------------------------------------------
Count Matrices
----------------------------------------------------------------------
All
   +/+ Aff3/Aff3
12   1         0
13 893        13
14   1         0


Males only
   +/+ Aff3/Aff3
12   1         0
13 448         6
14   0         0


Females only
   +/+ Aff3/Aff3
12   0         0
13 445         7
14   1         0
```

10.5   PhenStat LR framework Usage Example

The following dataset, provided by Wellcome Trust Sanger Institute (WTSI) Mouse Genetics Project (MGP) 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 standardized high throughput phenotyping pipeline following a multi-batch workflow, where

regular control animals are collected and knockout animals of the correct age are then 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 wildtype population.

10.5.1 Loading the data and initial steps of analysis

Initial steps focus on loading the data, using the PhenStat tools to generate the PhenList object, recoding the variables to the (0: normal, 1: abnormal) that is required for this framework and then generating the result object. We can then explore the data and fitted results using the visualisation and output functions.

```
> file <- system.file("extdata", "test_categorical.csv", package="PhenStat")
> dataset_cat <- read.csv(file)
> test <- PhenList(dataset=dataset_cat, testGenotype="Aff3/Aff3",
  refGenotype="+/+",
  dataset.colname.batch="Assay.Date")
> test <- LRDataset(test, depVariable="Thoracic.Processes",
abnormalValues="Abnormal")
> result <- testDataset(test, depVariable="Thoracic.Processes", method="LR")
```

10.5.2 Visualisation of data

The *categoricalBarplot* function is used 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.
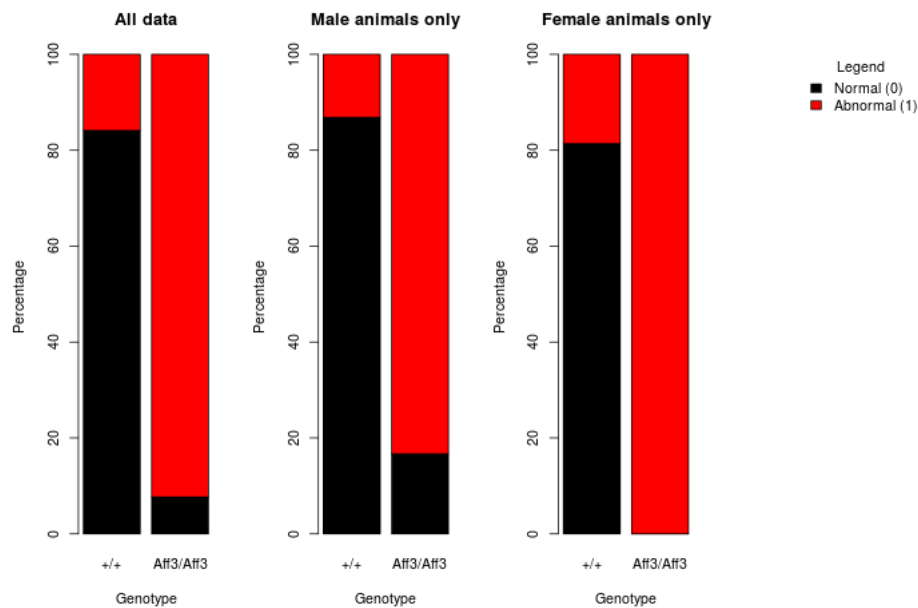
```
> categoricalBarplot(result)
```

Figure 45: *categoricalBarplot* function for Akt2 Thoracic Processes example after recoding to 0 and 1 for Logistic Regression analysis.

10.5.3 Understanding the summary output

```
> summaryOutput(result)


Test for dependent variable:
*** Thoracic.Processes ***


Method:
*** Logistic Regression ***


--------------------------------------------------------------------------
Model Output
--------------------------------------------------------------------------
Final fitted model: Thoracic.Processes ~ Genotype + Sex
Was batch significant? FALSE
Genotype p-value: 1.943939e-09
Genotype effect: 3.7983 +/- 0.9034
Was there evidence of sexual dimorphism? no (p-value 5.441219e-01)


--------------------------------------------------------------------------
Classification Tag
--------------------------------------------------------------------------
With phenotype threshold value 0.01 - both sexes equally
```

```
--------------------------------------------------------------------------
Model Output Summary
--------------------------------------------------------------------------

                     Value Std.Error   ci.lower     ci.upper      p-value
(Intercept)      -1.4646494 0.1209814 -1.7077333 -1.23321580 0.000000e+00
GenotypeAff3/Aff3 3.7983232 0.9033813  2.3619588  6.02332990 1.943939e-09
SexMale          -0.4251768 0.1840563 -0.7891747 -0.06661705 2.002838e-02
```

The first two lines of *summaryOutput* function results indicate the statistical framework used and the dependent variable studied.

The next section, titled "Model output", of the output clarifies the final fitted model details and global findings. As the "LR" framework is an optimisation process exploring the data to fit the best model to the data, the final fitted model details will vary. The "Final fitted model" tells us that the final model includes both genotype and sex as sex was found to be a significant source of variation. The validation test to assess whether batch was significant tells us that batch wasn't significant. This is a positive outcome as the Biased Reduction Logistic Regression package underpinning this analysis cannot include random effects and hence we cannot manage significant batch effects.

The "Was there evidence of sexual dimorphism?" reports the outcome of the statistical assessment of 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 = 0.54) hence the final model assessed a genotype effect rather than the genotypes for each sex separately.

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 $Thoracic.Processes \sim Sex$. Looking at the output the "Genotype p-value" is highly statistical significant as a $1.9e - 9$ value is reported. The genotype effect is estimated at 3.79 ± 0.9. Logistic regression estimates log odds and to relate this back to the experiment it is recommended that you convert the estimate into an odds ratio by calculating the exponential of the value (i.e. E^β or $\exp(^\beta)$). Where an odds ratio is an indicator of the change in odds resulting from a unit change in the predictor. In this example that gives us an odds ratio of 44.2 which tells us that being a knockout mouse leads to a 44.2 higher odds of being abnormal.

The section titled "Classification Tag" indicate how the genotype effect has

been classed. In this case as the model found no evidence of sexual dimorphism the effect was found to effect "both sexes equally".

Finally in the section "Model Output Summary", the estimates from the final model are reported.

10.6 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 cluster's node and stores the results in particular directory is created. This function is based on the method *getStat* of the *PhenList* object.

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

  # checks if variable is continuous again by using getStat table
  isContinuous <- datasetStat$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 dataset to analyse should also be available for every cluster node.

```
# cluster preparation
# set the folder
setwd("/yourWorkingDirectory")

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

file <- system.file("extdata", "test_Akt2.csv", package="PhenStat")
DEXAdata <- read.csv(file)
test <- PhenList(dataset=DEXAdata,
                 testGenotype="Akt2/Akt2",
                 refGenotype="WT",
                 dataset.colname.batch="dateOfExperiment",
                 dataset.values.female="female",
                 dataset.values.male="male",
                 dataset.colname.genotype="Genotype",
                 dataset.colname.sex="Gender")

# define tasks
tasks <- c(1:length(getVariables(test)))

# 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("/yourWorkingDirectory"))

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

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

# load package and source function for each node
clusterEvalQ(cluster, library(PhenStat))
clusterEvalQ(cluster, source("/pathToTheSource/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 ",
```

```
getVariables(test)[x], " ----------"); try(PhenStatCluster(test,x)); })

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

The output is avaialble in the specified directory: "/yourWorkingDirectory". 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.

Box, George E. P.; Cox, D. R. (1964). "An analysis of transformations". Journal of the Royal Statistical Society, Series B 26 (2): 211–252.

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.

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

Pinkert,C.A. (2002) Transgenic Animal Technology. A laboratory Handbook (2002), *Academic Press Inc.*, USA

Solberg,H.E. (1983) The theory of reference values Part 5. Statistical treatment of collected reference values. Determination of reference limits, *Journal of clinical chemistry and clinical biochemistry*, **21** (11), 749-760

Heinze,G. (2006) A comparative investigation of methods for logistic regression with separated or nearly separated data, *Statistics in Medicine*, **25**, 4216-4226