OpenStats: A Robust and Scalable Software Package for Reproducible Analysis of High-Throughput Phenotypic Data

true

31 March 2020

Contents

OpenStats is a freely available R package that presents statistical methods and detailed analyses to promote the hard process of identification of abnormal phenotypes. The package incorporates several checks and cleaning on the input data prior to the statistical analysis. For continuous data, Linear Mixed Model with an optional model selection routine is implemented, whilst for categorical data, Fisher's Exact Test is implemented. For cases where the linear mixed model fails, Reference Range Plus method has been employed for a quick, simple analysis of the continuous data. User can perform inspections and diagnostics of the final fitted model by the visualisation tools that come with the software. Furthermore, the user can export/report the outputs in the form of either standard R list or JavaScript Object Notation (JSON). OpenStats has been tested and demonstrated with an application of $2.5M+$ analyses from the Internationa Mouse Phenotyping Consortium (IMPC).

The User's Guide with more details about the statistical analysis is available as part of the online documentation from (https://rpubs.com/hamedhm/openstats). Project Github repository including *dev* version of the package is available on (https://git.io/JeOVN).

Building block of the software

OpenStats consists of one input layer and three operational layers: - **(Input layer)** Input data and specifiying model: this includes the input data and an initial model in the form of standard R formula, e.g. $y \sim x + 1$. - **(Operational layer 1)** Dataset preprocessing: this includes checking, cleaning and terminology unification procedures and is completed by the function *OpenStatsList* which creates an *OpenStatsList* object. - **(Operational layer 2)** Data analysis: this is managed by the function *OpenStatsAnalysis* and consists of Linear Mixed Model, Fisher's Exact test and Reference Range plus framework implementations. The results are stored in an *OpenStatsMM/FE/RR* object. - **(Operational layer 3)** Report/Export: the exports/reports are managed by the function *OpenStatsReport*. *OpenStats* reports the outputs in the form of either List or JSON objects.

Data preprocessing

OpenStatsList function performs data processing and creates an *OpenStatsList* object. As input, *OpenStatsList* function requires dataset of phenotypic data that can be presented as data frame. For instance, it can be dataset stored in csv, tsv or txt file. Data is organised with rows and columns for samples and features respectively. Following shows an example of the input data where rows and columns represent mice and features (mouse id, treatment group, gender, age of animals in days):

```
library(OpenStats)
```

```
##
##  >========================================================================<
##  OpenStats is developed by International Mouse Phenotyping Consortium (IMPC)
##  More details           : https://www.mousephenotype.org/
##  Source code and issues : https://git.io/JeOVN
##  Contact us             : hamedhm@ebi.ac.uk
##  >========================================================================<
```

```
###################
# Data preparation
###################
fileCon = system.file("extdata", "test_continuous.csv",
                      package = "OpenStats")
read.csv(fileCon, as.is = TRUE)[60:75, c('external_sample_id'     ,
                                         'biological_sample_group',
                                         'sex'                    ,
                                         'age_in_days')]
```

```
##    external_sample_id biological_sample_group    sex age_in_days
## 60             C10058                 control female          53
## 61             C10059                 control female          53
## 62             C10060                 control female          53
## 63             C10061                 control female          53
## 64             C10062                 control female          53
## 65             C10063                 control   male          53
## 66             C10064                 control   male          53
## 67             C10065                 control   male          53
## 68             C10066                 control   male          53
## 69             C10067                 control   male          53
## 70             C10192            experimental   male          46
## 71             C10193            experimental   male          46
## 72             C10194            experimental   male          46
## 73             C10195            experimental   male          46
## 74             C10197            experimental   male          46
## 75             C10199            experimental   male          46
```

The main preprocessing tasks performed by the *OpenStatsList* function are:

- terminology unification,
- filtering out undesirable records (when the argument *dataset.clean* is set to TRUE),
- imputing missings such as blanks, spaces or user-specified terms with NA,
- and checking whether the dataset can be used for the statistical analysis.

We define "terminology unification" as the terminology used to describe data (variables) that are essential for the analysis. *OpenStats* package uses the following nomenclature for the names of columns: "Genotype", the only mandatory variable, "Sex", "Batch" "LifeStage" and "Weight". In addition, expected (default) Sex, LifeStage values are "Male/Female" and "Early/Late" respectively. However, the user can define the custome

levels by setting *dataset.values.male*, *dataset.values.female*, *dataset.values.early* and *dataset.values.late* in the OpenStatsList function. Missing value is specified by *dataset.values.missingValue* argument and set to *NA*.

The statistical analysis requires exactly two "Genotype" groups for comparison (e.g. wild-type versus knockout). Thus the function *OpenStatsList* requires users to define the reference genotype (mandatory argument *refGenotype* with default value "control") and test genotype (mandatory argument *testGenotype*), defaulted to "experimental"). If the *OpenStatsList* function argument *dataset.clean* is set to TRUE then all records with genotype values others than reference or test genotype are filtered out.

All tasks in OpenStats are accompanied by step-by-step reports, error messages, warnings and/or other useful information about the progress of the function. If messages are not desirable, *OpenStatsList* function's argument *debug* can be set to FALSE meaning there will be no messages.

The chunk of code below demonstrates an example of using *OpenStatsList* when the user sets out-messages to TRUE/FALSE:

```
  ########################################
  # Default behaviour with messages
  ########################################
  library(OpenStats)
  fileCon = system.file("extdata", "test_continuous.csv",
                        package = "OpenStats")
  test_Cont = OpenStatsList(
    dataset = read.csv(fileCon),
    testGenotype = 'experimental',
    refGenotype  = 'control',
    dataset.colname.genotype  = 'biological_sample_group',
    dataset.colname.batch     = 'date_of_experiment',
    dataset.colname.lifestage = NULL,
    dataset.colname.weight    = 'weight',
    dataset.colname.sex       = 'sex'
  )
```

```
## 2020-03-31 11:19:57. Input data of the dimensions, rows = 410, columns = 75

## 2020-03-31 11:19:57. Checking the input data in progress ...

## 2020-03-31 11:19:57. Checking the specified missing values [x2] (` `, ``) ...

## 2020-03-31 11:19:57.     1/2. Checking (` `) ...

## 2020-03-31 11:19:57.     2/2. Checking (``) ...

## 2020-03-31 11:19:57. Checking whether variable `biological_sample_group` exists in the data ...
## 2020-03-31 11:19:57.     Result = TRUE

## 2020-03-31 11:19:57.      Levels (Total levels = 2, missings = 0%):
## 2020-03-31 11:19:57.       1. control
## 2020-03-31 11:19:57.       2. experimental

## 2020-03-31 11:19:57. Checking whether variable `sex` exists in the data ...
## 2020-03-31 11:19:57.     Result = TRUE

## 2020-03-31 11:19:57.      Levels (Total levels = 2, missings = 0%):
## 2020-03-31 11:19:57.       1. female
## 2020-03-31 11:19:57.       2. male

## 2020-03-31 11:19:57. Checking whether variable `date_of_experiment` exists in the data ...
## 2020-03-31 11:19:57.     Result = TRUE

## 2020-03-31 11:19:57.      Levels (Total levels = 43, missings = 0%):
```

```
## 2020-03-31 11:19:57.       1. 2012-07-23T00:00:00Z
## 2020-03-31 11:19:57.       2. 2012-07-30T00:00:00Z
## 2020-03-31 11:19:57.       3. 2012-08-06T00:00:00Z
## 2020-03-31 11:19:57.       4. 2012-08-13T00:00:00Z
## 2020-03-31 11:19:57.       5. 2012-08-20T00:00:00Z
## 2020-03-31 11:19:57.       6. 2012-11-26T00:00:00Z
## 2020-03-31 11:19:57.       7. 2012-12-24T00:00:00Z
## 2020-03-31 11:19:57.       8. 2013-01-02T00:00:00Z
## 2020-03-31 11:19:57.       9. 2013-01-15T00:00:00Z
## 2020-03-31 11:19:57.       10. 201 ...

## 2020-03-31 11:19:57. Checking whether variable `weight` exists in the data ...
## 2020-03-31 11:19:57.     Result = TRUE

## 2020-03-31 11:19:57.      Summary:
## 2020-03-31 11:19:57.       mean      = 20.0036585365854
## 2020-03-31 11:19:57.       sd        = 2.63972182732584
## 2020-03-31 11:19:57.       Missings  = 0%

## 2020-03-31 11:19:57. Variable `biological_sample_group` renamed to `Genotype`

## 2020-03-31 11:19:57. Variable `sex` renamed to `Sex`

## 2020-03-31 11:19:57. Variable `date_of_experiment` renamed to `Batch`

## 2020-03-31 11:19:57. Variable `weight` renamed to `Weight`

## 2020-03-31 11:19:57. Total samples in Genotype:Sex interactions:
## 2020-03-31 11:19:57.      Level(frequency):
## 2020-03-31 11:19:57.       1. control.Female(196)
## 2020-03-31 11:19:57.       2. experimental.Female(5)
## 2020-03-31 11:19:57.       3. control.Male(201)
## 2020-03-31 11:19:57.       4. experimental.Male(8)

## 2020-03-31 11:19:57. Total `Weight` data points for Genotype:Sex interactions:
## 2020-03-31 11:19:57.      Level(frequency):
## 2020-03-31 11:19:57.       1. control.Female(196),
## 2020-03-31 11:19:57.       2. experimental.Female(5),
## 2020-03-31 11:19:57.       3. control.Male(201),
## 2020-03-31 11:19:57.       4. experimental.Male(8)

## 2020-03-31 11:19:57. Successfully performed checks in 0.18 second(s).
```

```r
  ######################################
  # Default behaviour without messages
  ######################################
  fileCon = system.file("extdata", "test_continuous.csv",
                        package = "OpenStats")
  test_Cont = OpenStatsList(
    dataset = read.csv(fileCon),
    testGenotype = 'experimental',
    refGenotype  = 'control',
    dataset.colname.genotype  = 'biological_sample_group',
    dataset.colname.batch     = 'date_of_experiment',
    dataset.colname.lifestage = NULL,
    dataset.colname.weight    = 'weight',
    dataset.colname.sex       = 'sex',
    debug                     = FALSE
```

```
)
# No output printed
```

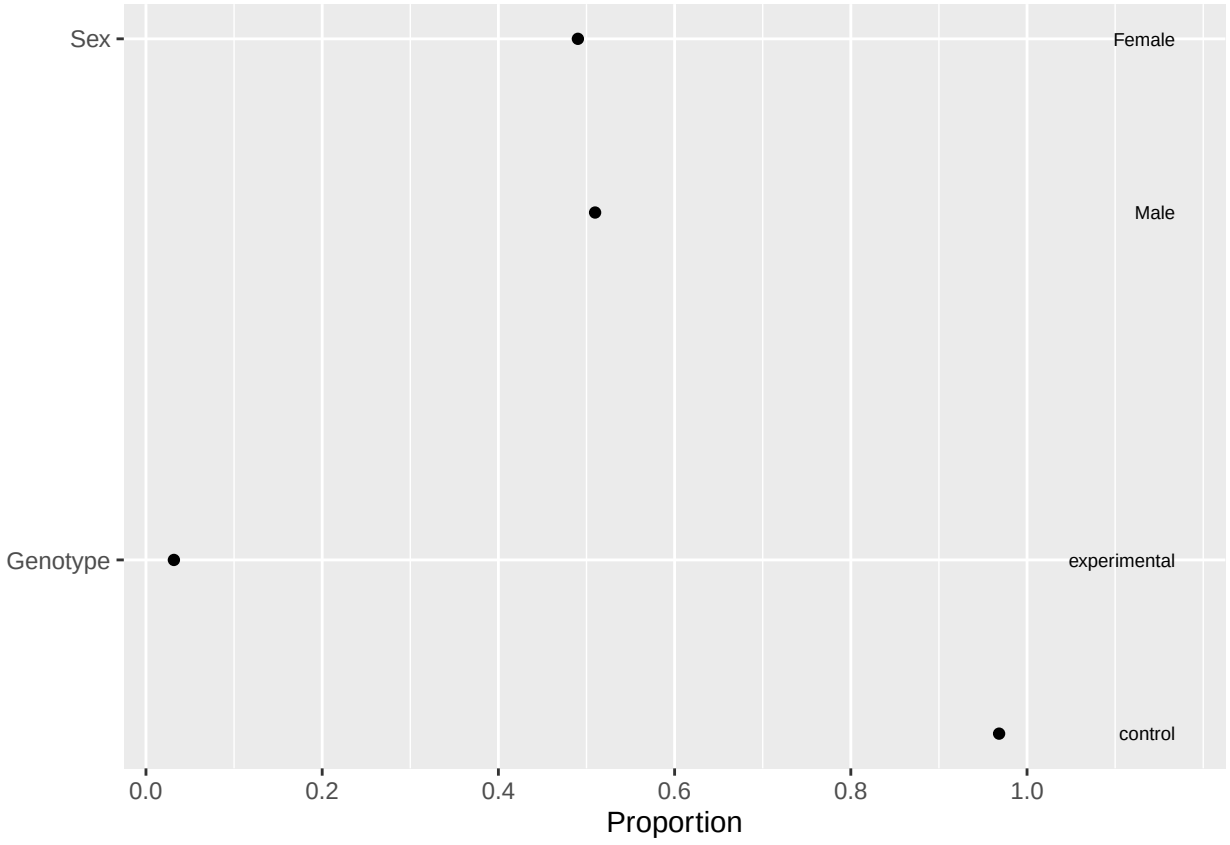
OpenStatsList Object

The output of the *OpenStatsList* function is the *OpenStatsList* object that contains a cleaned dataset as well as a copy of the original dataset. *OpenStats* allows **plot** and **summary** of the OpenStatList object. Below is an example of the OpenStatsList function accompanied by the plot and summary:

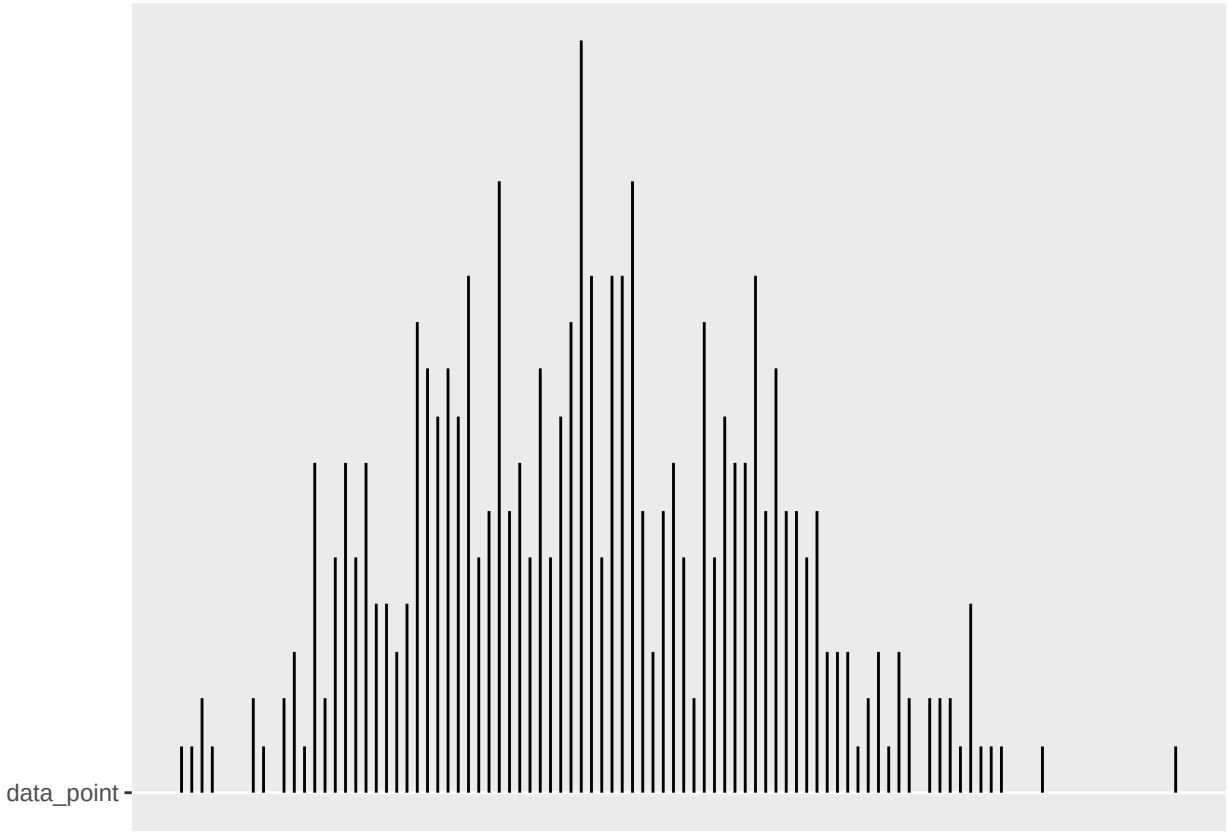
```
library(OpenStats)
df = read.csv(system.file("extdata", "test_continuous.csv",
                          package = "OpenStats"))
OpenStatsList   = OpenStatsList(
  dataset      = df,
  testGenotype = 'experimental',
  refGenotype  = 'control',
  dataset.colname.batch     = 'date_of_experiment',
  dataset.colname.genotype  = 'biological_sample_group',
  dataset.colname.sex       = 'sex',
  dataset.colname.weight    = 'weight',
  debug                     = FALSE
)
plot   (OpenStatsList, vars = c('Sex', 'Genotype', 'data_point'))
```

```
## 2020-03-31 11:19:58. Working on the plot ...
```

```
## $Categorical
```



```
##
## $Continuous
```




```
summary(OpenStatsList,
        style     = 'grid',
        varnumbers= FALSE, # See more options ?summarytools::dfSummary
        graph.col = FALSE, # Do not show the graph column
        valid.col = FALSE,
        vars      = c('Sex', 'Genotype', 'data_point'))
```

```
## Loading required namespace: summarytools
```

```
## Registered S3 method overwritten by 'pryr':
##   method      from
##   print.bytes Rcpp
```

```
## 2020-03-31 11:20:00. Working on the summary table ...
```

```
## Data Frame Summary
##
## Dimensions: 410 x 3
## Duplicates: 0
##
## +-------------+----------------------+--------------------+---------+
## | Variable    | Stats / Values       | Freqs (% of Valid) | Missing |
## +=============+======================+====================+=========+
## | Sex         | 1. Female            | 201 (49.0%)        | 0       |
## | [factor]    | 2. Male              | 209 (51.0%)        | (0%)    |
```

```
## +-------------+-----------------------+---------------------+---------+
## | Genotype    | 1. control            | 397 (96.8%)         | 0       |
## | [factor]    | 2. experimental       |  13 ( 3.2%)         | (0%)    |
## +-------------+-----------------------+---------------------+---------+
## | data_point  | Mean (sd) : 11 (1.7)  | 408 distinct values | 0       |
## | [numeric]   | min < med < max:      |                     | (0%)    |
## |             | 7 < 10.9 < 16.7       |                     |         |
## |             | IQR (CV) : 2.6 (0.2)  |                     |         |
## +-------------+-----------------------+---------------------+---------+
```

OpenStatsList object stores many characteristics of the data, for instance, reference genotype, test genotype, original column names, factor levels etc.

Data Analysis

OpenStats package contains three statistical frameworks for the phenodeviants identification:

- Simple Linear/Linear Mixed Models framework that assumes baseline values of the dependent variable are normally distributed but batch (defined as the date of experiment in the IMPC) is the between-group source of variation.
- Reference Range Plus framework identifies the normal variation form a group called *Reference variable* (wild-type animals in the IMPC), classifies dependent variable as low, normal or high and compare proportions. This framework recommended for the sufficient number of controls (more than 60 records) to correctly identify normal variation.
- Fishers Exact Test is a standard framework for categorical data which compares data proportions and calculates the percentage change in classification.

OpenStats's function *OpenStatsAnalysis* works as a hub for the different statistical analysis methods. It checks the dependent variable, the data, missings, not proper terms in the model (such as terms that do not exist in the input data) and runs the selected statistical analysis framework and returns modelling/testing results. All analysis frameworks output a statistical significance measure, effect size measure, model diagnostics, and graphical visualisations.

Here we explain the main bits of the *OpenStatsAnalysis* function:

- **OpenStatsListObject**: defines the dataset stored in an *OpenStatsList* object.
- **model**: defines the fixed effect model for example, $Response \sim Genotype + Sex$
- **method**: defines which statistical analysis framework to use.

The possible values for the *method* arguments are "MM" which stands for mixed model framework, "FE" to perform Fisher's exact test model and "RR" for Reference Range Plus framework. The semantic naming in the input arguments of the OpenStatsAnalysis function allows natural distinction of the input arguments For example, $MM\_$, $RR\_$ and $FE\_$ prefix represent the arguments that can be set in the corresponging frameworks. Having said that,

- MM_fixed, MM_random, MM_weight refer to the fixed effect terms, random effect term and between group variation
- FE_formula refers to the model that need to be analysed by Fisher's exacts test(the default $category \sim Genotype + Sex + LifeStage$ in the IMPC)
- RR_formula, RRrefLevel, RR_prop refer to the Reference Range plus model (default $data\_point \sim Genotype + Sex + LifeStage$ in the IMPC). Note that the first term on the right hand side of the model (here *Genotype*) is the *Reference Variable* and the reference level is defined by "RRrefLevel" (default is set "control" in the IMPC). Finally the natural variation of the reference level to define the so called "NORMAL" category is determined by *RR_prop* (IMPC 0.95 that is mutants outside the .025 quantile from right/left tails of the distribution are labeles as high/low respectively).

The *OpenStatsAnalysis* function performs basic checks to ensure that the data and model match, the model is feasible for the type of the data and reports step-by-step progress of the function. Some of the checks are

listed below:

- Mixed Model (MM) frameworks:
  - *MM_checks*: A vector of four 1/0 or TRUE/FALSE values such as c(TRUE, TRUE, TRUE, TRUE)[default]. Performing pre-checks on the input model for some known scenarios. The first element of the vector activates checks on the model terms (in MM_fixed) to have existed in data. The second term removes any single level -factor- from the model (in MM_fixed). The third term removes the single value (such as a column of constants/no variation) from the -continuous- terms in the model (in MM_fixed). The Fourth element checks the interaction term to make sure all interactions have some data attached. Caution is needed for this check as it may take longer than usual if the formula in MM_fixed contains many factors.
  - Note that OpenStatsAnalysis function always removes duplicated columns in the dataset prior to applying the linear mixed model.
  - Regardless of the check settings, the OpenStatsAnalysis function always checks for the existence of the "MM_random" terms (provided "MM_random" is set) in the input data
- Reference Range Plus (RR) and Fisher's exact test (FE) framework's:
  - *FERR_FullComparisions* Only applies to the "RR" or "FE" frameworks. A vector of two logical flags, default c(TRUE, FALSE). Setting the first value to TRUE, then all combinations of the effects (all levels of factors in the input model - for example Male_LifeStage, Male_Genotype, Male_Mutant, Male_control, Female_control, Female_Mutant, Female_LifeStage and so on) will be tested. Otherwise only *main effects* (no sub-levels - for example Sex_LifeStage [not for instance Male_LifeStage]) will be tested. Setting the second element of the vector to TRUE (default FALSE) will force the Fisher's Exact test to do all comparisons between different levels of the RESPONSE variable. For example, if the response has three levels such as 1) positive, 2) negative and 3) neutral then the comparison will be between 1&2, 1&3, 2&3 and 1&2&3 (obviously the latter is the full table).
- All frameworks
  - OpenStatsAnalysis allows confidence intervals for all estimates in three frameworks. One can set the confidence level by setting MMFERR_conf.level to a value in $(0, 1)$ interval (default 0.95).

All frameworks are equipped with the step-by-step report of the progress of the function. Warnings, errors and messages are reported to the user. In the situation where the function encounters a critical failure, then the output object contains a slot called *messages* that reports back the cause of the failure.

Examples

In this section, we show some examples of the functionalities in *OpenStats* for the continuous and categorical data. Each section contains the code and different possible scenarios.

Linear mixed model framework

The linear mixed model framework applies to continuous data. In this example, data is extracted from the sample data that accompany the software. Here, "Genotype" is the effect of interest. The response is stored in the variable "data_point" and genotype (Genotype) and body weight (Weight) are covariates. The model selection is left to the default, stepwise, and between-group covariance structure are assumes proportional to the genotype levels (different variation for controls than mutants):

```
library(nlme)
library(OpenStats)
  ##################
  # Data preparation
  ##################
  ##################
  # Continuous data - Creating OpenStatsList object
  ##################
  fileCon = system.file("extdata", "test_continuous.csv",
```

```
                        package = "OpenStats")
test_Cont = OpenStatsList(
    dataset = read.csv(fileCon),
    testGenotype = 'experimental',
    refGenotype  = 'control',
    dataset.colname.genotype  = 'biological_sample_group',
    dataset.colname.batch     = 'date_of_experiment',
    dataset.colname.lifestage = NULL,
    dataset.colname.weight    = 'weight',
    dataset.colname.sex       = 'sex',
    debug                     = FALSE
)
##################
# LinearMixed model (MM) framework
##################
MM_result = OpenStatsAnalysis(
    OpenStatsList = test_Cont,
    method   = 'MM',
    MM_fixed = data_point ~ Genotype  + Weight
)
```

```
## 2020-03-31 11:20:00. OpenStats loaded.

## 2020-03-31 11:20:00. Checking the input model for including functions ...

## 2020-03-31 11:20:00. Linear Mixed Model (MM framework) in progress ...

## 2020-03-31 11:20:00. Removing possible leading/trailing whitespace from the variables in the formula

## 2020-03-31 11:20:00. Checking duplications in the data model:
## 2020-03-31 11:20:00.      Genotype, data_point, Weight

## 2020-03-31 11:20:00.     No duplicate found.

## 2020-03-31 11:20:00. Checking for the feasibility of terms and interactions ...
## 2020-03-31 11:20:00.      Formula: data_point ~ Genotype + Weight

## 2020-03-31 11:20:00.     1 of 1. Checking for the feasibility of terms and interactions ...

## 2020-03-31 11:20:00.          Checking Genotype ...

## 2020-03-31 11:20:00.     Checked model: data_point ~ Genotype + Weight

## 2020-03-31 11:20:00. Check missings in progress ...

## 2020-03-31 11:20:00.      Missings in variable `data_point`: 0%

## 2020-03-31 11:20:00.      Missings in variable `Genotype`: 0%

## 2020-03-31 11:20:00.      Missings in variable `Weight`: 0%

## 2020-03-31 11:20:00. Checking the random effect term ...
## 2020-03-31 11:20:00.     Formula: ~1 | Batch

## 2020-03-31 11:20:00. Lme: Fitting the full model ...

## 2020-03-31 11:20:00.     Applied model: lme

## 2020-03-31 11:20:00.     The full model successfully applied.

## 2020-03-31 11:20:00.     Computing the confidence intervals at the level of 0.95 ...

## 2020-03-31 11:20:00.      CI for `all` term(s) successfully estimated
```

```
## 2020-03-31 11:20:00. The specified "lower" model:
## 2020-03-31 11:20:00.     ~Genotype + 1
## 2020-03-31 11:20:00. The model optimisation is in progress ...
## 2020-03-31 11:20:00.     The direction of the optimisation (backward, forward, both): both
## 2020-03-31 11:20:00.     Optimising the model ...
## 2020-03-31 11:20:01.     Optimised model: data_point ~ Genotype + Weight
## 2020-03-31 11:20:01.     Computing the confidence intervals at the level of 0.95 ...
## 2020-03-31 11:20:01.      CI for `all` term(s) successfully estimated
## 2020-03-31 11:20:01. Testing varHom ...
## 2020-03-31 11:20:01.     Computing the confidence intervals at the level of 0.95 ...
## 2020-03-31 11:20:01.      CI for `all` term(s) successfully estimated
## 2020-03-31 11:20:01.     VarHom checked out ...
## 2020-03-31 11:20:01. Testing Batch ...
## 2020-03-31 11:20:01.     Computing the confidence intervals at the level of 0.95 ...
## 2020-03-31 11:20:01.      CI for `all` term(s) successfully estimated
## 2020-03-31 11:20:01. Estimating effect sizes ...
## 2020-03-31 11:20:01.     Total effect sizes estimated: 2
## 2020-03-31 11:20:01. Quality tests in progress ...
## 2020-03-31 11:20:01. MM framework executed in 1 second(s).
```

Sub-model estimation

OpenStats llows stimateing submodels from an input model. This is called Split model effects in the outputs and it is mainly useful for reporting sex/age-specific etc. effects. This is performed by creating submodels of a full model. For instance, for the input fixed effect (*MM_fixed*) model $response \sim Genotype + Sex + Weight$ a possible submodel is $response \sim Sex + Sex : Genotype + Weight$ that can be used to estimate sex-specific effects for genotype. This model is then estimated under the configuration of the optimal model. One can turn off Split model effects by setting the fourth element of *MM_optimise* to FALSE.

An alternative to the analytically estimating the sub-models is to break the input data into splits and run the model on the subset of the data. This can be performed by passing the output of OpenStatsAnalysis function, *OpenStatsMM*, to the function, *OpenStatsComplementarySplit*. This function allows the OpenStatsMM object as input and a set of variable names that split happens on. The example below shows the split on "Sex":

```
library(nlme)
library(OpenStats)
  ##################
  # Data preparation
  ##################
  ##################
  # Continuous data - Creating OpenStatsList object
  ##################
  fileCon = system.file("extdata", "test_continuous.csv",
                        package = "OpenStats")
  test_Cont = OpenStatsList(
      dataset = read.csv(fileCon),
      testGenotype = 'experimental',
```

```
        refGenotype  = 'control',
        dataset.colname.genotype  = 'biological_sample_group',
        dataset.colname.batch     = 'date_of_experiment',
        dataset.colname.lifestage = NULL,
        dataset.colname.weight    = 'weight',
        dataset.colname.sex       = 'sex',
        debug                     = FALSE
    )
    ##################
    # LinearMixed model (MM) framework
    ##################
    MM_result = OpenStatsAnalysis(
        OpenStatsList = test_Cont,
        method        = 'MM',
        MM_fixed      = data_point ~ Genotype  + Weight,
        debug         = FALSE
    )
    # SplitEffect estimation with respect to the Sex levels
    Spliteffect = OpenStatsComplementarySplit(object    = MM_result,
                                              variables = 'Sex')
```

```
## 2020-03-31 11:20:03. Split effects in progress ...

## 2020-03-31 11:20:03. Variable(s) to split:
## 2020-03-31 11:20:03.     Sex

## 2020-03-31 11:20:03.     Splitting in progress ...

## 2020-03-31 11:20:03.     Spliting on Sex ...

## 2020-03-31 11:20:03. Processing the levels: Female

## 2020-03-31 11:20:04. [Successful]

## 2020-03-31 11:20:04. Processing the levels: Male
## 2020-03-31 11:20:04. [Successful]
```

Reference range plus framework

Reference range plus framework applies to continuous data. In this example, data is extracted from the sample data that accompany the software. Here, "Genotype" is the effect of interest. The response is stored in the variable "data_point" and genotype (Genotype) and sex (Sex) are covariate.

```
  library(nlme)
  library(OpenStats)
    ##################
    # Data preparation
    ##################
    ##################
    # Continuous data - Creating OpenStatsList object
    ##################
    fileCon = system.file("extdata", "test_continuous.csv",
                          package = "OpenStats")
    test_Cont = OpenStatsList(
        dataset = read.csv(fileCon),
        testGenotype = 'experimental',
        refGenotype  = 'control',
        dataset.colname.genotype  = 'biological_sample_group',
```

```
        dataset.colname.batch     = 'date_of_experiment',
        dataset.colname.lifestage = NULL,
        dataset.colname.weight    = 'weight',
        dataset.colname.sex       = 'sex',
        debug                     = FALSE
    )
  ##################
    # Reference range framework
    ##################
    RR_result = OpenStatsAnalysis(
        OpenStatsList = test_Cont,
        method        = 'RR',
        RR_formula    = data_point ~ Genotype + Sex
    )
```

```
## 2020-03-31 11:20:05. OpenStats loaded.

## 2020-03-31 11:20:05. Checking the input model for including functions ...

## 2020-03-31 11:20:05. Reference Range Plus (RR framework) in progress ...

## 2020-03-31 11:20:05. Optimisation level:

## 2020-03-31 11:20:05.     Estimation of all factor combination effects = TRUE

## 2020-03-31 11:20:05.     Estimation of inter level factors for the response = FALSE

## 2020-03-31 11:20:05. The input formula: data_point ~ Genotype + Sex

## 2020-03-31 11:20:05. The reformatted formula for the algorithm: ~data_point + Genotype + Sex

## 2020-03-31 11:20:05. The probability of the middle area in the distribution: 0.95

## 2020-03-31 11:20:05.      Tails probability: 0.025
## 2020-03-31 11:20:05.      Formula to calculate the tail probabilities: 1-(1-x)/2, (1-x)/2 where x = (

## 2020-03-31 11:20:05. Discritizing the continuous data into discrete levels. The quantile = 0.975

## 2020-03-31 11:20:05. Stp 1. Low versus Normal/High

## 2020-03-31 11:20:05. Removing possible leading/trailing whitespace from the variables in the formula

## 2020-03-31 11:20:05. Preparing the reference ranges ...

## 2020-03-31 11:20:05. Preparing the data for the variable: Genotype

## 2020-03-31 11:20:05. Reference level is set to `control`

## 2020-03-31 11:20:05.     Initial quantiles for cutting the data
## 2020-03-31 11:20:05.              Probs: 0, 0.025, 1
## 2020-03-31 11:20:05.              N.reference: 397
## 2020-03-31 11:20:05.              Quantiles: 6.04, 8.08, 17.73

## 2020-03-31 11:20:05.      Detected percentiles in the data (8 decimals): Low = 0.02518892, NormalHigh

## 2020-03-31 11:20:05.     Spliting on Sex ...

## 2020-03-31 11:20:05. Preparing the data for the combined effect: Female

## 2020-03-31 11:20:05. Reference level is set to `control`

## 2020-03-31 11:20:05.     Initial quantiles for cutting the data
## 2020-03-31 11:20:05.              Probs: 0, 0.025, 1
## 2020-03-31 11:20:05.              N.reference: 196
```

```
## 2020-03-31 11:20:05.              Quantiles: 7.579, 9.267, 17.73

## 2020-03-31 11:20:05.      Detected percentiles in the data (8 decimals): Low = 0.0255102, NormalHigh

## 2020-03-31 11:20:05. Preparing the data for the combined effect: Male

## 2020-03-31 11:20:05. Reference level is set to `control`

## 2020-03-31 11:20:05.     Initial quantiles for cutting the data
## 2020-03-31 11:20:05.              Probs: 0, 0.025, 1
## 2020-03-31 11:20:05.              N.reference: 201
## 2020-03-31 11:20:05.              Quantiles: 6.04, 7.673, 15.645

## 2020-03-31 11:20:05.      Detected percentiles in the data (8 decimals): Low = 0.02985075, NormalHigh

## 2020-03-31 11:20:05. Stp 2. Low/Normal versus High

## 2020-03-31 11:20:05. Removing possible leading/trailing whitespace from the variables in the formula

## 2020-03-31 11:20:05. Preparing the reference ranges ...

## 2020-03-31 11:20:05. Preparing the data for the variable: Genotype

## 2020-03-31 11:20:05. Reference level is set to `control`

## 2020-03-31 11:20:05.     Initial quantiles for cutting the data
## 2020-03-31 11:20:05.              Probs: 0, 0.975, 1
## 2020-03-31 11:20:05.              N.reference: 397
## 2020-03-31 11:20:05.              Quantiles: 6.04, 14.5, 17.73

## 2020-03-31 11:20:05.      Detected percentiles in the data (8 decimals): LowNormal = 0.97481108, High

## 2020-03-31 11:20:05.     Spliting on Sex ...

## 2020-03-31 11:20:05. Preparing the data for the combined effect: Female

## 2020-03-31 11:20:05. Reference level is set to `control`

## 2020-03-31 11:20:05.     Initial quantiles for cutting the data
## 2020-03-31 11:20:05.              Probs: 0, 0.975, 1
## 2020-03-31 11:20:05.              N.reference: 196
## 2020-03-31 11:20:05.              Quantiles: 7.579, 14.744, 17.73

## 2020-03-31 11:20:05.      Detected percentiles in the data (8 decimals): LowNormal = 0.9744898, High

## 2020-03-31 11:20:05. Preparing the data for the combined effect: Male

## 2020-03-31 11:20:05. Reference level is set to `control`

## 2020-03-31 11:20:05.     Initial quantiles for cutting the data
## 2020-03-31 11:20:05.              Probs: 0, 0.975, 1
## 2020-03-31 11:20:05.              N.reference: 201
## 2020-03-31 11:20:05.              Quantiles: 6.04, 13.527, 15.645

## 2020-03-31 11:20:05.      Detected percentiles in the data (8 decimals): LowNormal = 0.97014925, High

## 2020-03-31 11:20:05. Fisher exact test with 1500 iteration(s) in progress ...

## 2020-03-31 11:20:05. Analysing Low vs NormalHigh ...

## 2020-03-31 11:20:05. Analysing LowNormal vs High ...

## 2020-03-31 11:20:05. RR framework executed in 0.51 second(s).
```

Fisher's exact test framework

Fisher's Exact test framework applies to categorical data. In this example, data is extracted from the sample data that accompany the software. Here, Genotype is the effect of interest. The response is stored in the variable *category* and Genotype and Sex are the covariates.

```
library(nlme)
library(OpenStats)
##################
  # Categorical data - Creating OpenStatsList object
  ##################
  fileCat   = system.file("extdata", "test_categorical.csv",
                          package = "OpenStats")
  test_Cat = OpenStatsList(
      dataset      = read.csv(fileCat, na.strings = '-'),
      testGenotype = 'Aff3/Aff3',
      refGenotype  = '+/+',
      dataset.colname.genotype  = 'Genotype',
      dataset.colname.batch     = 'Assay.Date',
      dataset.colname.lifestage = NULL,
      dataset.colname.weight    = 'Weight',
      dataset.colname.sex       = 'Sex',
      debug                     = FALSE
  )
##################
  # Fisher's exact test framework
  ##################
  FE_result = OpenStatsAnalysis(
      OpenStatsList = test_Cat,
      method     = "FE",
      FE_formula =  Thoracic.Processes ~ Genotype + Sex
  )
```

```
## 2020-03-31 11:20:05. OpenStats loaded.
## 2020-03-31 11:20:05. Checking the input model for including functions ...
## 2020-03-31 11:20:05. Fisher Exact Test (FE framework) in progress ...
## 2020-03-31 11:20:05. Optimisation level:
## 2020-03-31 11:20:05.     Estimation of all factor combination effects = TRUE
## 2020-03-31 11:20:05.     Estimation of inter level factors for the response = FALSE
## 2020-03-31 11:20:05. The input formula: Thoracic.Processes ~ Genotype + Sex
## 2020-03-31 11:20:05. The reformatted formula for the algorithm: ~Thoracic.Processes + Genotype + Sex
## 2020-03-31 11:20:05.     Top framework: FE
## 2020-03-31 11:20:05. Fisher exact test with 1500 iteration(s) in progress ...
## 2020-03-31 11:20:05. Check missings in progress ...
## 2020-03-31 11:20:05.      Missings in variable `Thoracic.Processes`: 0.32%
## 2020-03-31 11:20:05.      Missings in variable `Genotype`: 0%
## 2020-03-31 11:20:05.      Missings in variable `Sex`: 0%
## 2020-03-31 11:20:05. Removing possible leading/trailing whitespace from the variables in the formula
```

```
## 2020-03-31 11:20:05. Step 1. Testing "Thoracic.Processes"
## 2020-03-31 11:20:05. The data (variable(s) = Thoracic.Processes) contain 2 missing(s) ...
## 2020-03-31 11:20:05.     Missing data removed
## 2020-03-31 11:20:05.     Testing for the main effect: Genotype
## 2020-03-31 11:20:05.     Testing for the main effect: Sex
## 2020-03-31 11:20:05. Combined effects in progress ...
## 2020-03-31 11:20:05.     Splitting in progress ...
## 2020-03-31 11:20:05.     Spliting on Genotype ...
## 2020-03-31 11:20:05.     Spliting on Sex ...
## 2020-03-31 11:20:05.     Shrinking in progress ...
## 2020-03-31 11:20:06.     Dichotomising the final tables ...
## 2020-03-31 11:20:06.     Finalising the tables ....
## 2020-03-31 11:20:06. Testing for the combined effects ...
## 2020-03-31 11:20:06. Step 2. Testing "Genotype"
## 2020-03-31 11:20:06.     Testing for the main effect: Sex
## 2020-03-31 11:20:06. Total tested categories = 2: Thoracic.Processes, Genotype
## 2020-03-31 11:20:06.     Total tests =  7
## 2020-03-31 11:20:06. FE framework  executed in 0.07 second(s).
```

OpenStatsAnalysis output object

OpenStatsAnalysis output consists of three elements namely, *input*, *output* and *extra*. The *input* object encapsulate the input parameters to the function, *output* hold the analysis results and the *extra* keeps some extra processes on the data/model. Below is an example output from the Reference Rage plus framework:

```
library(nlme)
library(OpenStats)
  #################
  # Data preparation
  #################
  #################
  # Continuous data - Creating OpenStatsList object
  #################
  fileCon = system.file("extdata", "test_continuous.csv",
                        package = "OpenStats")
  test_Cont = OpenStatsList(
      dataset = read.csv(fileCon),
      testGenotype = 'experimental',
      refGenotype  = 'control',
      dataset.colname.genotype  = 'biological_sample_group',
      dataset.colname.batch     = 'date_of_experiment',
      dataset.colname.lifestage = NULL,
      dataset.colname.weight    = 'weight',
      dataset.colname.sex       = 'sex',
      debug                     = FALSE
  )
#################
```

```
    # Reference range framework
    #################
    RR_result = OpenStatsAnalysis(
        OpenStatsList = test_Cont,
        method        = 'RR',
        RR_formula    = data_point ~ Genotype + Sex,
        debug         = FALSE
    )
    lapply(RR_result,names)
```

```
## $output
## [1] "SplitModels"
##
## $input
##  [1] "OpenStatsList"     "data"              "depVariable"
##  [4] "rep"               "method"            "formula"
##  [7] "prop"              "ci_level"          "refLevel"
## [10] "full_comparisions"
##
## $extra
## [1] "Cleanedformula"    "TransformedRRprop"
```

```
    #lapply(RR_result$output,names)
```

Summary and export

OpenStats package stores the results of statistical analyses in the *OpenStatsMM/RR/FE* object. The standard *summary* function can be applied to print out a summary table. The function *OpenStatsReport* can be used to create a table of detailed summary in the form of either list or JSON. The following is an example of the summary output of the MM framework. The same function applied to the FE and RR output objects.

```
summary(MM_result)
```

```
## 2020-03-31 11:20:06. Working on the summary table ...
```

```
##
##
## ============================  ================================================================================
## Statistic                     Value
## ============================  ================================================================================
## Applied framework             Linear Mixed Model framework, LME, including Weight
## Final model                   data_point ~ Genotype + Weight
## ............................  ................................
## Tested Gene                   experimental
## Reference Gene                control
## ............................  ................................
## Sexual dimorphism detected?   FALSE, Genotype-Sex interaction is not part of the input (it is not pa
## ............................  ................................
## Genotype contribution overall 0.343064998063529
## Genotype contribution Females -
## Genotype contribution Males   -
## ............................  ................................
## LifeStage contribution        -
## Genotype contribution Early   -
## Genotype contribution Late    -
```

```
## ..........................   ..........................
## Sex contribution               -
## Body weight contribution       0
## ============================  ================================================================================
```

OpenStatsReport function was developed for large scale application where automatic implementation is require. Following is the JSON output of the function from an *OpenStatsMM* object (cut to the first 1500 charachters):

```
strtrim(
    OpenStatsReport(
        object = MM_result,
        JSON = TRUE,
        RemoveNullKeys = TRUE,
        pretty = TRUE
    ),
    1500)
```


```
## {
##   "Applied method": "Linear Mixed Model framework, LME, including Weight",
##   "Dependent variable": "data_point",
##   "Batch included": true,
##   "Residual variances homogeneity": false,
##   "Genotype contribution": {
##     "Overall": 0.343064998063529,
##     "Sexual dimorphism detected": {
##       "Criteria": false,
##       "Note": "Genotype-Sex interaction is not part of the input (it is not part of the final) model
##     }
##   },
##   "Genotype estimate": {
##     "Value": -0.502512854643372,
##     "Confidence": {
##       "Genotypeexperimental lower": -1.54340602996328,
##       "Genotypeexperimental upper": 0.538380320676539
##     },
##     "Level": 0.95
##   },
##   "Genotype standard error": 0.529316719407687,
##   "Genotype p-value": 0.343064998063529,
##   "Genotype percentage change": {
##     "control Genotype": -4.66480574285704,
##     "experimental Genotype": 8.36890228977311
##   },
##   "Genotype effect size": {
##     "Value": 0.572742111514943,
##     "Variable": "Genotype",
##     "Model": "data_point ~ Genotype",
##     "Type": "Mean differences",
##     "Percentage change": {
##       "control Genotype": -4.66480574285704,
##       "experimental Genotype": 8.36890228977311
##     }
##   },
##   "Weight estimate": {
##     "Value": -0.390270185772556,
```

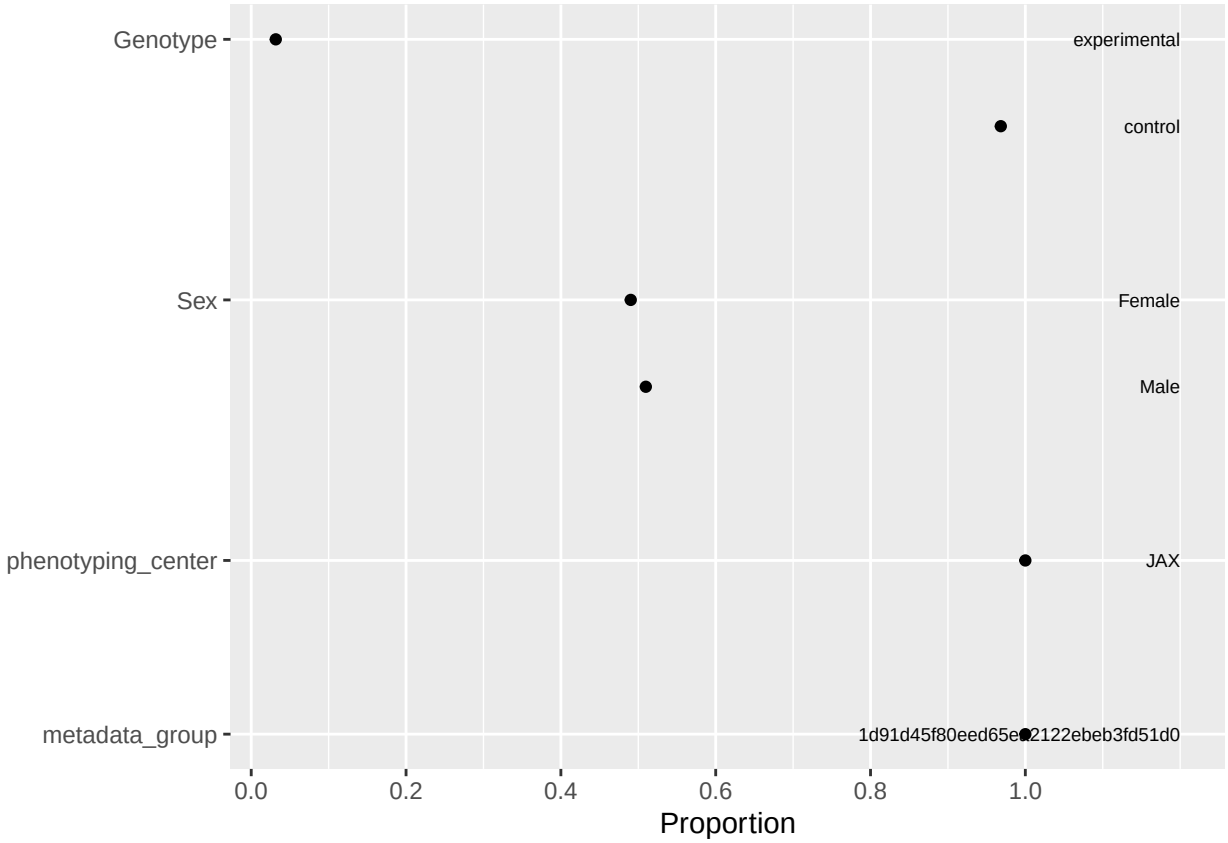
```
##     "Confidence": {
##       "Weight lower": -0.432146253204354,
##       "Weight upper": -0.348394118340757
##     },
##     "Level": 0.95
##   },
##   "Weight standard error": 0.0212948870837609,
##   "Weight p-value": 0,
##   "Weight effect size": {
##     "Value": -0.598034137811873,
##     "Variable": "Weight",
##     "Model
```

Graphics

Graphics in *OpenStats* are as easy as calling the **plot()** function on a OpenStatsList or the OpenStatsMM/FE/RR object. Calling the plot function on the OpenStatsList object is shown below:

```
library(OpenStats)
  ###################
    file = system.file("extdata", "test_continuous.csv",
                       package = "OpenStats")
  ###################
  # OpenStatsList object
  ###################
  OpenStatsList   = OpenStatsList(
    dataset   = read.csv(file),
    testGenotype = 'experimental',
    refGenotype = 'control',
    dataset.colname.batch = 'date_of_experiment',
    dataset.colname.genotype = 'biological_sample_group',
    dataset.colname.sex = 'sex',
    dataset.colname.weight = 'weight',
    debug = FALSE
  )
    plot   (OpenStatsList)
```

```
## 2020-03-31 11:20:07. Working on the plot ...
```




```
summary(
  OpenStatsList,
  style     = 'grid',
  varnumbers = FALSE,      # See more options ?summarytools::dfSummary
  graph.col = FALSE,       # Do not show the graph column
  valid.col = FALSE
)
```

```
## 2020-03-31 11:20:08. Working on the summary table ...
```

```
## Data Frame Summary
##
## Dimensions: 410 x 6
## Duplicates: 320
##
## +---------------------+--------------------------------+--------------------+---------+
## | Variable            | Stats / Values                 | Freqs (% of Valid) | Missing |
## +=====================+================================+====================+=========+
## | Genotype            | 1. control                     | 397 (96.8%)        | 0       |
## | [factor]            | 2. experimental                |  13 ( 3.2%)        | (0%)    |
## +---------------------+--------------------------------+--------------------+---------+
## | Sex                 | 1. Female                      | 201 (49.0%)        | 0       |
## | [factor]            | 2. Male                        | 209 (51.0%)        | (0%)    |
## +---------------------+--------------------------------+--------------------+---------+
## | Batch               | 1. 2012-07-23T00:00:00Z        |  10 ( 2.4%)        | 0       |
## | [factor]            | 2. 2012-07-30T00:00:00Z        |  10 ( 2.4%)        | (0%)    |
## |                     | 3. 2012-08-06T00:00:00Z        |  10 ( 2.4%)        |         |
```

```
## |                      | 4. 2012-08-13T00:00:00Z       |  10 ( 2.4%)         |          |
## |                      | 5. 2012-08-20T00:00:00Z       |   9 ( 2.2%)         |          |
## |                      | 6. 2012-11-26T00:00:00Z       |  10 ( 2.4%)         |          |
## |                      | 7. 2012-12-24T00:00:00Z       |  10 ( 2.4%)         |          |
## |                      | 8. 2013-01-02T00:00:00Z       |  10 ( 2.4%)         |          |
## |                      | 9. 2013-01-15T00:00:00Z       |   9 ( 2.2%)         |          |
## |                      | 10. 2013-01-21T00:00:00Z      |   4 ( 1.0%)         |          |
## |                      | [ 33 others ]                 | 318 (77.6%)         |          |
## +----------------------+-------------------------------+---------------------+----------+
## | age_in_weeks         | Mean (sd) : 7.5 (0.6)         | 6 :   8 ( 1.9%)     | 0        |
## | [integer]            | min < med < max:              | 7 : 216 (52.7%)     | (0%)     |
## |                      | 6 < 7 < 9                     | 8 : 174 (42.4%)     |          |
## |                      | IQR (CV) : 1 (0.1)            | 9 :  12 ( 2.9%)     |          |
## +----------------------+-------------------------------+---------------------+----------+
## | phenotyping_center   | 1. JAX                        | 410 (100.0%)        | 0        |
## | [factor]             |                               |                     | (0%)     |
## +----------------------+-------------------------------+---------------------+----------+
## | metadata_group       | 1. 1d91d45f80eed65ea2122ebeb  | 410 (100.0%)        | 0        |
## | [factor]             |                               |                     | (0%)     |
## +----------------------+-------------------------------+---------------------+----------+
```

There are also graphics for the OpenStatsMM/FE/RR. Below shows an example for the OpenStatsMM output:

```
plot(MM_result, col = 2)
```

```
## 2020-03-31 11:20:08. Working on the plot ...
```