GWAStoolbox
An R package for the fast processing of data from Genome-Wide Association Studies

Christian Fuchsberger Daniel Taliun Cristian Pattaro

August 20, 2010

Contents

1 Introduction

GWAStoolbox is an *R* package for processing data originated from Genome-Wide Association Studies (GWAS). GWAS have become increasingly popular in the last years, leading to the discovery of hundreds of common genetic variants affecting the risk of diseases (such as diabetes, hypertension, chronic kidney disease, etc.) or the level of quantitative biological parameters.

Results from GWAS typically consist in large files where, for each single nucleotide polymorphisms (SNP), statistics related to the association between the SNP and the studied trait are stored. The number of SNPs which is currently being analyzed in most GWAS is in excess of 2.5 Million and is expected to increase rapidly. For each individual SNP, the minimal information stored consists of the SNP identification code (SNPID), chromosomal position, genotype (reference and non-reference alleles), frequency of the reference allele, and SNP effect size and its standard error. Additional information such as p-value, minor allele frequency (MAF), and an imputation quality index are often provided. As a consequence, the typical dimension of GWAS result files is of >2.5 Million rows by >9 columns, for a total file size which is often larger than 300 Mbytes.

With the aim of detecting common or less common genetic variants with modest effects, it is now common practice to pool results from individual studies into meta-analysis efforts which not rarely involve dozens of studies. In these consortia initiatives, each individual study contributes several files either because multiple traits are being analyzed or because different analyses on the same trait are needed. Consequently, statisticians working in consortia have to deal with a massive amount of files which need to be quality controlled to avoid problems during the meta-analysis process. As a result of the quality control (QC) process, some files could be found to be corrupted or erroneous so that new data upload is needed from individuals studies. In this way, the loop between the consortium and the individual study analyst originates multiple file checks, until a satisfactory data quality is achieved.

When working with such large datasets in R, simple operations such as the uploading files into the R working space, file management, and data plotting, can take considerable time, and a systematic QC of hundreds of files can be unfeasible or may require several weeks.

With the *GWAStoolbox* we provide a set of instruments to simplify the data handling in the framework of meta-analyses of GWA data. The function *gwasqc()* is capable to process a high number of GWAS data files in a single run, and producing several QC reports and figures. A routine for the between-study comparison is also provided to check systematic difference between files. In addition, the package contains annotation and graphical tools to help the result interpretation.

2 Installation

GWAStoolbox package can be downloaded from `http://www.eurac.edu/en/research/institutes/geneticmedicine/Software/GWAStoolbox.html`. It requires R version 2.9.2 or higher. The installation of package varies depending on your host operating system and user privileges. In this section we provide detailed installation instructions for a wide range of settings.

2.1 Windows

GWAStoolbox for Windows is distributed in compiled binary form. The following steps describe the installation procedure:

1. Download the latest package version *GWAStoolbox_X.Y.Z.zip*.

2. Start the R program.

3a. If you have administrator privileges (you can install packages to the main R library):

   i. Execute the command:

   ```
   install.package("path/to/GWAStoolbox_X.Y.Z.zip", re-
   pos=NULL)
   ```

   where `path/to` is the directory of the downloaded package.

   ii. Now you can load the package in R with the command:

   ```
   library(GWAStoolbox)
   ```

3b. If you do NOT have sufficient privileges to install packages to the main R library directory:

   i. Execute the command:

   ```
   install.package("path/GWAStoolbox_X.Y.Z.zip",
   lib="path/to/install/directory",
   repos=NULL)
   ```

   where `path/to/install/directory` is the path with your install directory.

   ii. Now you can load the package in R with command:

   ```
   library(GWAStoolbox, lib.loc = "path/to/install/directory")
   ```

2.2 Unix

GWAStoolbox for Unix is distributed in source form and, therefore, it is compiled on the user machine. This requires the following tools to be installed:

- C/C++ compilers
- GNU Scientific Library (GSL)* version 1.8

When these requirements are fulfilled, the following steps will guide you through the package installation process:

1. Download the latest package version *GWAStoolbox_X.Y.Z.tar.gz*.

2a. If you have administrator privileges (you can install packages to the main R library):

   i. In the Unix shell execute the command:

   ```
   R CMD INSTALL path/to/GWAStoolbox_X.Y.Z.tar.gz
   ```

*http://www.gnu.org/software/gsl/

where `path/to` is the directory of the downloaded package.

ii. Now you can start the R program and load the package with the command:

```
library(GWAStoolbox)
```

2b. If you do NOT have sufficient privileges to install packages to the main R library directory:

i. In the Unix shell execute the single line command:

```
R CMD INSTALL path/to/GWAStoolbox_X.Y.Z.tar.gz
-l path/to/install/directory
```

where `path/to/install/directory` is the path with your install directory.

ii. Now you can start the R program and load the package with the command:

```
library(GWAStoolbox, lib.loc="path/to/install/directory")
```

3 The Quality Control Workflow

A careful and thorough data QC should be performed before starting any meta-analysis of GWAS data, especially when many studies are involved. In this framework, we identified three objectives of a good QC analysis:

1. formal checking: whether all files that will be entered in the meta-analysis process fulfill the format guidelines. This includes:
   - consistency of column names with meta-analysis guidelines;
   - presence of the minimal required information;
   - the number of chromosomes is as expected;
   - data are in a format that can be analyzed (numeric, character, factor);
   - all SNP identification numbers are unique;
   - alleles are coded in letters/numbers as expected;
   - missing values are coded in a consistent way;
   - the field separator is as expected;
   - strand assessment;
2. quality checking: evaluating the quality of data in each single file. This includes:
   - presence of unexpected values for some of the items required for the meta-analysis (e.g.: negative p-values or standard errors);
   - p-value inflation and p-value distribution;
3. global checking: identification of any systematic biases that can disturb the analysis. It is aimed to uncover studies that are systematically different from the others. This may happen when, for instance, analysts of one study forget to log-transform the phenotype or apply the wrong model to the data.

Formal checks and quality checks of individual studies are performed in *GWAStoolbox* using the *gwasqc()* function. *gwasqc()* was built to include the following features:

1. rapid file processing and reporting;
2. eliminate routine user operations;
3. multi-format reporting which includes *HTML*, *CSV*, and text files.

The complete QC workflow can be summarized in four basic steps (see Figure 1):

1. collect the GWAS data files;
2. write an input script to process of all GWAS files with the *gwasqc()* function;
3. run the QC using *gwasqc()*;
4. analyze the QC results to uncover errors or inconsistencies.

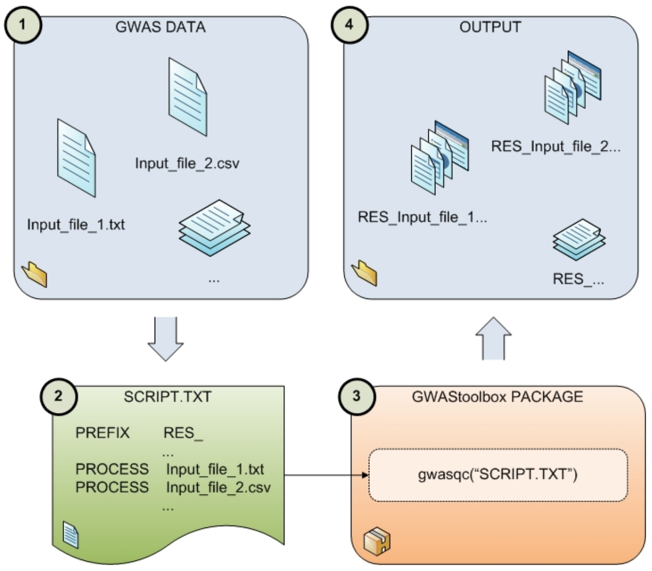


Figure 1: The quality control workflow.

In the next sections we cover each of the four steps and describe the requirements for the input files and the precise content of all output files.

4 GWAS data files

GWAS data are usually stored as delimited text files. The first line of the file is the header row that describes the content of every column. The field separator for the columns can be any among *whitespace*, *tabulation*, *comma*, or *semicolon*.

The field separator must be the same for every row in the file, including the header.

There is a minimum set of columns, that every GWAS data file should contain. In *GWAStoolbox*, the following information is required for every file:

- Marker name
- Chromosome number or name
- Marker position
- Coded and non-coded allele
- Allele frequency for coded allele
- Strand
- Imputation label
- Imputation quality
- Effect size
- Standard error
- P-value

The *gwasqc()* function will take full responsibility for checking if an input file contains all the information and will report about missing data.

More non-mandatory items can be included in the data file as, for example, the study sample size, the SNP call rate for genotyped SNPs, the p-value of the Hardy-Weinberg equilibrium test for genotyped SNPs, etc.

5 The Input Script

gwasqc() can analyze several GWAS data files consecutively. Instructions are provided using a script in a text file. The format of the script file resembles the one of the METAL input files[†].

Within the input script file, the user can list all GWAS file names to be analyzed and specify the format of each single GWAS file, including column names, field separator, etc. In the case that more GWAS files are in the same format, file specifications can be entered only once, before listing the file names. Example 1 illustrates the content of a hypothetical input script file.

Example 1

```
# Description of input data columns
MARKER          SNPID
CHR             Chromosome
POSITION        Position
N               n_total
ALLELE          coded_allele noncoded_allele
```

[†]http://www.sph.umich.edu/csg/abecasis/metal/

```
STRAND        strand
EFFECT        beta
STDERR        se
PVALUE        pval
FREQLABEL     allele_freq_coded_allele
IMPUTED       imputed
IMP_QUALITY   oevar_imp

# High quality filters
HQ_SNP   0.01   0.3

# Plotting filters
MAF      0.01   0.05
IMP      0.3    0.6

# Prefix for output files
PREFIX   res_

# Input file with GWA data
PROCESS  input_file.txt
```

◁

5.1 Specifying Input Data Files

The names of the GWAS data files are specified in the input script with the command **PROCESS**[‡] If multiple files have to be checked, multiple **PROCESS** lines must be specified.

Example 2 The input script contains the following two lines:

```
PROCESS   input_file_1.txt
PROCESS   /dir_1/dir_2/input_file_2.csv
```

Then, QC is applied first to *input_file_1.txt* and then to *input_file_2.csv*. As used in the example, when files reside in different directories, the full path must be specified. ◁

5.2 Describing Input Data Columns

5.2.1 Field Separator

The field separator may be different for each GWAS data file. The *gwasqc()* function automatically detects the separator field for each input file *based on the first 10 rows*. However, the user has the possibility to specify the separator manually for each individual file using the command **SEPARATOR**. Table 1 lists all supported separators.

Example 3 In the following input script:

[‡] *GWAStoolbox* supports single line feed ('\n') character or carriage return character ('\r') followed by line feed character as the line terminators in the input files.

Argument	Separator
COMMA	*comma*
TAB	*tabulation*
WHITESPACE	*whitespace*
SEMICOLON	*semicolon*

Table 1: The list of arguments for the SEPARATOR command.

```
PROCESS      input_file_1.txt
SEPARATOR    TAB
PROCESS      input_file_2.csv
PROCESS      input_file_3.txt
```

the field separator for the input file *input_file_1.txt* is determined automatically by *gwasqc()*, but for the input files *input_file_2.csv* and *input_file_3.txt* the separator is manually set to tabulation. ◁

5.2.2 Missing Values

By default *gwasqc()* assumes that missing values are labeled as *NA*. However, the label for missing value can be specified manually by the user with the command **MISSING**.

Example 4 Let's assume the following input script:

```
MISSING     -
PROCESS     input_file_1.txt
MISSING     NA
PROCESS     input_file_2.csv
```

For the file *input_file_1.txt* the *hyphen* symbol is set as symbol for missing value. Afterwards, it is changed to *NA* and is used to process *input_file_2.csv*. ◁

5.2.3 Column Names

In table 2 the complete list of the default column names for a GWAS data file is reported. These names identify uniquely the items in the GWAS data file.

Given that different names can be provided with the GWAS data files, *gwasqc()* allows to redefine the default values for every input file in the input script. The redefinition command consists of the default column name followed by a new column name. To redefine the default column names for *coded* and *non-coded* alleles, the command **ALLELE** is followed by two new column names.

Example 5 Let's assume to have two input files, *input_file_1.txt* and *input_file_2.txt*. In *input_file_1.txt*, the column names for effect size and standard error are *beta* and *SE*, respectively. In the *input_file_2.txt*, the column name for the effect size is the same as in *input_file_1.txt*, but the column name for the standard error is *STDERR*. The correct column redefinitions are as follows:

Default column name(s)	Description
MARKER	Marker name
CHR	Chromosome number or name
POSITION	Marker position
ALLELE1, ALLELE2	Coded and non-coded alleles
FREQLABEL	Allele frequency for the coded allele
STRAND	Strand
IMPUTED	Label value indicating if the marker was imputed (1) or genotyped (0)
IMP_QUALITY	Imputation quality statistics; this can be different depending on the software used for imputation: MACH's *Rsq*, IMPUTE's *properinfo*, ...
EFFECT	Effect size
STDERR	Standard error
PVALUE	P-value
HWE_PVAL	Hardy-Weinberg equilibrium p-value
CALLRATE	Genotype callrate
N	Sample size
USED_FOR_IMP	Label value indicating if a marker was used for imputation (1) or not (0)

Table 2: The default column names.

```
EFFECT      beta
STDERR      SE
PROCESS     input_file_1.txt
STDERR      STDERR
PROCESS     input_file_2.csv
```

First, we redefine column names for the input file *input_file_1.txt*. We note that the column *beta* doesn't need to be redefined for the input file *input_file_2.csv*. However, for this file we need to redefine the column *STDERR*, returning it to the default column naming. ◁

Example 6 Consider an input file *input_file_1.txt* with the following names for ALLELE1 and ALLELE2: *myRefAllele* and *myNonRefAllele*. The new column definition is applied as follows:

```
ALLELE     myRefAllele myNonRefAllele
PROCESS    input_file_1.txt
```

◁

5.2.4 Case Sensitivity

By default the *gwasqc()* function assumes that column names in the input files are case insensitive. For example, the column names *STDERR*, *StdErr*, and *STDErr* are all perfectly equivalent. This behaviour can be changed for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names. Table 3 lists all possible arguments.

Argument	Description
0	Column names in the input file are case insensitive (default)
1	Column names in the input file are case sensitive

Table 3: The list of arguments for CASESENSITIVE command.

Example 7 Consider the following commands:

```
CASESENSITIVE   1
PROCESS         input_file_1.txt
CASESENSITIVE   0
PROCESS         input_file_2.csv
```

In this case, the column names in the input file *input_file_1.txt* are case sensitive and must correspond exactly to the default column names, while the column names in the input file *input_file_2.csv* are case insensitive. $\triangleleft$

5.3 Specifying Data Filters

5.3.1 Implausible Values Filter

Often, there is the necessity to identify implausible values for the statistics that will be included in the meta-analysis. Implausible values for the effect estimate, for its standard error, and for the p-value are sometimes generated by the software used for the association testing. In case of small numbers, which is typical of a disease outcome with a small number of cases or of a SNP with very small minor allele frequency, statistical packages can report inconsistent results. This is due to statistical algorithms that fail to converge because of data sparseness. Other types of inconsistencies can originate from errors in the file management.

In these situations, it is important to identify the SNPs with inconsistent values, so that they can be removed before starting the meta-analysis. *gwasqc()* can identify these values by using appropriate threshold values. The number of SNPs affected by this kind of problems is reported. In addition, these SNPs are excluded from the calculation of the summary statistics on data quality.
The implausible values filter is used in the *gwasqc()* function to identify implausible data values. Table 4 lists the columns for which the filter is applied and the default thresholds.

Default column name	Default thresholds
STDERR	$[0, 100000]$
IMP_QUALITY	$(0, 1.5)$
PVALUE	$(0, 1)$
FREQLABEL	$(0, 1)$
HWE_PVAL	$(0, 1)$
CALLRATE	$(0, 1)$

Table 4: The default implausible values filter.

The default thresholds can be redefined for every column in the input script. The new threshold values for a column can be specified after the redefinition of the column name (see Section 5.2.3).

Example 8 Let's assume that the input file *input_file_1.txt* has a standard error column called *STDERR* and that the corresponding column in the input file *input_file_2.csv* is called *SE*. In addition, the imputation quality column is defined as *oevar_imp* in both files. The following script shows how the user can redefine the column names while applying different plausibility filters:

```
STDERR        STDERR 0 80000
IMP_QUALITY   oevar_imp 0 1
PROCESS       input_file_1.txt
STDERR        SE 0 100000
PROCESS       input_file_2.csv
```

The file *input_file_1.txt* has new $[0, 80000]$ thresholds for the standard error column and new $(0, 1)$ threshold for the imputation quality. For the file *input_file_2.csv* the thresholds of $[0, 100000]$ will be applied to the standard error column, while for the imputation quality column the same filters as for the *input_file_1.txt* will be applied. $\triangleleft$

5.3.2 High Quality Filters

In many cases, the analysis is restricted to SNPs with high imputation quality and with not too small minor allele frequency. We call these SNPs '*high quality SNPs*', that is SNPs for which results should be quite robust. In the special case, when estimating the inflation factor, *lambda*, to check the presence of cryptic relatedness or hidden population sub-structures, it can be important to remove SNPs that could artificially increase the value of *lambda*. Summary statistics are calculated after excluding SNPs with low quality (*CSV* report files). Table 5 lists the default thresholds for the allele frequency and for the imputation quality.

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

Table 5: The default high quality imputation filters.

The default values can be redefined using the command **HQ_SNP** for every input file in the input script. The command is followed by two values: the first one corresponds to the threshold for the minor allele frequency, and the second one corresponds to the threshold for the imputation quality.

Example 9 If we want to define 'high quality SNPs' those with minor allele frequency > 0.03 and with imputation quality > 0.4, we would add the following lines to the input script:

```
HQ_SNP      0.03 0.4
PROCESS     input_file_1.txt
```

◁

5.3.3 Plotting Filter

The plotting filter is used to select appropriate data for the QQ-plots, boxplots and histograms. The filter has two threshold levels: each of them is applied dependently on the plot type and column. Figure 2 (see Section 5.4.3) shows what data and filters are used when producing plots. Table 6 lists the default threshold values.

Default column name	Default 1st level thresholds	Default 2nd level thresholds
FREQLABEL	> 0.01	> 0.05
IMP_QUALITY	> 0.3	> 0.6

Table 6: The default plotting filter.

The default threshold values for the coded allele frequency and imputation quality can be redefined accordingly with the commands **MAF** and **IMP** for the every input file in the input script.

Example 10 Assume the input script contains the following commands:

```
MAF        0.02 0.03
IMP        0.3 0.5
PROCESS    input_file_1.txt
```

In this example new plotting filter thresholds are set for the input file *input_file_1.txt*. For the first level threshold the coded allele frequency > 0.02 and the imputation quality > 0.3, while for the second level threshold the coded allele frequency > 0.03 and imputation quality > 0.5. ◁

5.4 Specifying Output Files

5.4.1 Output File Name

The output file names are constructed from the input file names by adding the specified prefix. This is done both for the textual output files and image files. The prefix can be specified once for all input files, or for every single input file or groups of input files explicitly using the command **PREFIX**.

Example 11 Consider the following input script:

```
PREFIX      res_
PROCESS     input_file_1.txt
PROCESS     input_file_2.csv

PREFIX      result_
PROCESS     input_file_3.tab
```

In this example, all the result output files corresponding to the input files *input_file_1.txt* and *input_file_2.csv* will be prefixed with *res_*, while the result output files corresponding to the input file *input_file_3.tab* will be prefixed with *result_*. ◁

5.4.2 Verbosity Level

The *GWAStoolbox* package provides the possibility to control the number of generated output figures using command **VERBOSITY** (see Table 7 for the available options).

Argument	Description
1	The default and the lowest verbosity level.
2	The highest verbosity level.

Table 7: The list of arguments for the VERBOSITY command.

Example 12 Assume the input script contains the following commands:

```
VERBOSITY       2
PROCESS         input_file_1.txt
VERBOSITY       1
PROCESS         input_file_2.csv
```

In this example the input file *input_file_1.txt* is processed with the highest verbosity level and therefore all figures are produced, while the input file *input_file_2.csv* is processed with the lowest verbosity level and less output figures are generated. ◁

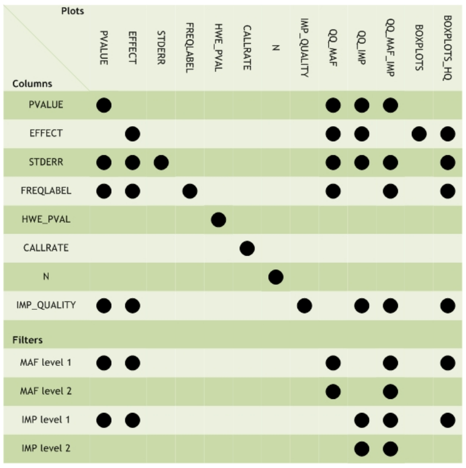
5.4.3 Number And Content Of Plots

Number and content of the output plots depend on the setting of the plotting filters (see Section 5.3.3) and on the available columns in the input file. Figure 2 shows the dependencies. If some dependency is not satisfied because of missing column or filter setting, then the corresponding plot is not produced or may be truncated at different levels.

Furthermore, the boxplots comparing the *EFFECT* distributions across studies allow the specification of a **BOXPLOTWIDTH** that can be based on one of the other available information (typically the sample size). As an argument, **BOXPLOTWIDTH** requires one of the default column names. If **BOXPLOTWIDTH** is not specified all boxplots have the same width.

It is also possible to specify labels for every input file, to be used in the plots instead of the full file names, which could be too long and, therefore, clutter the plots.

Example 13 Let *n_total* be the column name which identifies the sample size in the input file *input_file_1.txt*, and *samplesize* the corresponding name in *input_file_2.csv*. Then, consider the following input script:



	Plots												
Columns	PVALUE	EFFECT	STDERR	FREQLABEL	HWE_PVAL	CALLRATE	N	IMP_QUALITY	QQ_MAF	QQ_IMP	QQ_MAF_IMP	BOXPLOTS	BOXPLOTS_HQ
PVALUE	●								●	●	●		
EFFECT		●							●	●		●	●
STDERR	●	●	●						●	●	●		●
FREQLABEL	●	●		●					●		●		●
HWE_PVAL					●								
CALLRATE						●							
N							●						
IMP_QUALITY	●	●						●		●	●		●
Filters													
MAF level 1	●	●							●		●		●
MAF level 2									●		●		
IMP level 1	●	●								●	●		●
IMP level 2										●	●		

Figure 2: The dependency of plots on columns and filters.

```
N               n_total
PROCESS         input_file_1.txt first
N               samplesize
PROCESS         /dir_1/dir_2/input_file_2.csv second
BOXPLOTWIDTH    N
```

In this example, the width of the first boxplot for the input file *input_file_1.txt* depends on the *n_total* column, while the width of the second boxplot for the input file *input_file_2.csv* depends on the *samplesize* column. The labels "first" and "second" will be used to label the two studies in the plots. ◁

6 The Output Files

The *GWAStoolbox* package produces four types of files:

1. Figures with QQ-plots, histograms and boxplots (see Figure 2 for all employed input columns and filters).
2. Textual report file with *.txt* extension.
3. Comma separated file with *.csv* extension, that contains statistics for the high quality imputation data (see Section 5.3.2).
4. The *HTML* document, that combines both textual output and figures.

7 Example

This is an embedded R code example. All input files of this example are located in the subdirectory *doc* of the installed *GWAStoolbox* package.

Consider the two GWAS data files: *gwa_data_example_1.tbl* and *gwa_data_example_2.csv*. The first file contains 16 columns separated with tabulation:

```
> t <- read.table("gwa_data_example_1.tbl", header = T, nrow = 1,
+     sep = "\t")
> colnames(t)
```

```
 [1] "SNPID"         "chr"           "position"      "coded_all"
 [5] "noncoded_all"  "strand_genome" "beta"          "SE"
 [9] "pval"          "AF_coded_all"  "callrate"      "n_total"
[13] "oevar_imp"     "imputed"       "used_for_imp"  "HWE_pval"
```

At the same time, the second file also contains 16 columns, however separated with comma:

```
> t <- read.table("gwa_data_example_2.csv", header = T, nrow = 1,
+     sep = ",")
> colnames(t)
```

```
 [1] "SNPID"         "chr"           "position"      "coded_all"
 [5] "noncoded_all"  "strand_genome" "beta"          "StdErr"
 [9] "p"             "AF_coded_all"  "callrate"      "n_total"
[13] "oevar_imp"     "imputed"       "used_for_imp"  "HWE_pval"
```

In order to perform the quality control check of these two files with *GWAStoolbox* package, we prepare a simple input script *GWAS_script.txt*. Below are listed commands, which were inlcuded in the script:

```
> cat(readLines("GWASQC_script.txt"), sep = "\n")
```

```
# Column names
ALLELE                coded_all noncoded_all
CALLRATE         callrate
CHR                chr
EFFECT                beta
FREQLABEL         AF_coded_all
HWE_PVAL         HWE_pval
IMPUTED         imputed
IMP_QUALITY         oevar_imp
MARKER                SNPID
N                 n_total
POSITION         position
PVALUE                pval
STRAND                strand_genome
STDERR                SE
USED_FOR_IMP         used_for_imp

# Plotting filters for the coded allele frequency and imputation quality
MAF 0.01    0.05
IMP 0.3     0.5

# Prefix for output files
PREFIX        res_

# Column N controls the width of boxplots
```

```
BOXPLOTWIDTH        N

# Input file and its short name for plotting
PROCESS gwa_data_example_1.tbl first


PVALUE              p
STDERR              StdErr

PROCESS gwa_data_example_2.csv second
```

When the input script was prepared, we load the *GWAStoolbox* library and call the *gwasqc()* function as follows:

```
> library(GWAStoolbox)
> gwasqc("GWASQC_script.txt")
```

As a result, the following output files were generated:

```
boxplots.html
boxplots_EFFECT.png
boxplots_EFFECT_HQ.png
main.html
menu.html
res_gwa_data_example_1.tbl.csv
res_gwa_data_example_1.tbl.html
res_gwa_data_example_1.tbl.txt
res_gwa_data_example_1.tbl_AF_coded_all.png
res_gwa_data_example_1.tbl_beta.png
res_gwa_data_example_1.tbl_n_total.png
res_gwa_data_example_1.tbl_oevar_imp.png
res_gwa_data_example_1.tbl_pval.png
res_gwa_data_example_1.tbl_qqplot_IMP.png
res_gwa_data_example_1.tbl_qqplot_MAF.png
res_gwa_data_example_1.tbl_qqplot_MAF_IMP.png
res_gwa_data_example_1.tbl_SE.png
res_gwa_data_example_2.csv
res_gwa_data_example_2.html
res_gwa_data_example_2.txt
res_gwa_data_example_2_AF_coded_all.png
res_gwa_data_example_2_beta.png
res_gwa_data_example_2_n_total.png
res_gwa_data_example_2_oevar_imp.png
res_gwa_data_example_2_p.png
res_gwa_data_example_2_qqplot_IMP.png
res_gwa_data_example_2_qqplot_MAF.png
res_gwa_data_example_2_qqplot_MAF_IMP.png
res_gwa_data_example_2_StdErr.png
```

8 Additional Tools

Coming soon.

References

[1] Cristen J. Willer , Yun Li , and Gonçalo R. Abecasis. (2010) **METAL: fast and efficient meta-analysis of genomewide association scans**. Bioinformatics 26: 2190-2191.

[2] Paul I.W. de Bakker , Manuel A.R. Ferreira , Xiaoming Jia , Benjamin M. Neale , Soumya Raychaudhuri , and Benjamin F. Voight (2008) **Practical aspects of imputation-driven meta-analysis of genome-wide association studies**. Hum. Mol. Genet. 17: R122-R128.

Index