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

Christian Fuchsberger   Daniel Taliun   Cristian Pattaro

18-03-2013

Contents

1 Introduction

GWAtoolbox 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 of large files where, for each single nucleotide polymorphism (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 unique name, chromosomal position, genotype (reference and non-reference alleles), frequency of the reference allele, SNP effect size, its standard error, and p-value. Additional information such as minor allele frequency (MAF) and imputation quality are often provided. As a consequence, the typical dimension of GWAS result files is of >2.5 Million rows by >10 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 from one to several files, either because multiple traits are being analyzed or because different analyses on the same trait are needed. Consequently, data analysts working in these 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 individual 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 uploading the GWAS files into the R working space, file management, and data plotting, can take a considerable amount of time, and a systematic QC of hundreds of GWAS files can be unfeasible or may require several weeks.

The *GWAtoolbox* provides 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. Routines for the between-study comparison are also provided to check systematic difference between files. In addition, the package contains annotation and graphical tools to assist the result interpretation.

2 Installation

The *GWAtoolbox* package can be downloaded from http://www.eurac.edu/GWAtoolbox.html. It requires R version 2.15.0 or higher. The installation procedure depends on the host operating system and user privileges. In the following, detailed installation instructions for a wide range of settings are provided.

2.1 Windows

GWAtoolbox for Windows is distributed in compiled binary format. Installation:

1. Download the latest version of the package: *GWAtoolbox_X.Y.Z.zip*.

2. Start the R program.

3a. If you have administrator privileges, you can install the package to the main R library:

   i. Execute the command:
   ```
   install.packages("path/to/GWAtoolbox_X.Y.Z.zip",
   repos=NULL)
   ```
   where `path/to` is the directory where the package was downloaded.

   ii. Load the package in R with the command:
   ```
   library(GWAtoolbox)
   ```

3b. If you do NOT have administrator privileges:

   i. Execute the command:
   ```
   install.packages("path/to/GWAtoolbox_X.Y.Z.zip",
   lib="path/to/install/directory",
   repos=NULL)
   ```
   where `path/to` is the directory where the package was downloaded, and `path/to/install/directory` is the path with your install directory.

   ii. Load the package in R with command:
   ```
   library(GWAtoolbox, lib.loc =
   "path/to/install/directory")
   ```

2.2 Unix

GWAtoolbox for Unix is distributed in source format and, therefore, it needs to be compiled on the user machine. This requires the C/C++ compilers to be available on the system. If the requirement is fulfilled, the package can be installed as follows:

1. Download the latest package version *GWAtoolbox_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/GWAtoolbox_X.Y.Z.tar.gz
   ```
   where `path/to` is the directory where the package was downloaded.

   ii. Start the R program and load the package with the command:
   ```
   library(GWAtoolbox)
   ```

2b. If you do NOT have administrator privileges, follow the following steps:

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

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

where `path/to` is the directory where the package was downloaded, and `path/to/install/directory` is the path with your install directory.

ii. Start the R program and load the package with the command:

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

2.3   Mac OS X

GWAtoolbox for Mac OS X is distributed in compiled binary format. The following steps describe the installation procedure:

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

2a. To install from the Mac OS X shell (*Terminal*):

i. If you have administrator privileges, you can install packages to the main R library:

A. In the Mac OS X shell execute the command:

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

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

B. Start the R program and load the package with the command:

```
library(GWAtoolbox)
```

ii. If you do NOT have administrator privileges:

A. In the Mac OS X shell execute the single line command:

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

where `path/to` is the directory where the package was downloaded, and `path/to/install/directory` is the path with your install directory.

B. Start the R program and load the package with the command:

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

2b. To install from R:

i. Start the R program.

ii. If you have administrator privileges, you can install packages to the main R library:

A. Execute the command:

```
install.packages("path/to/GWAtoolbox_X.Y.Z.tar.gz",
repos=NULL)
```

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

B. Load the package in R with the command:

```
library(GWAtoolbox)
```
iii. If you do NOT have administrator privileges:

A. Execute the command:
```
install.packages("path/to/GWAtoolbox_X.Y.Z.tar.gz",
lib="path/to/install/directory",
repos=NULL)
```
where `path/to` is the directory where the package was downloaded, and `path/to/install/directory` is the path with your install directory.

B. Load the package in R with the command:
```
library(GWAtoolbox, 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 complementary aspects of a good QC analysis:

1. Formal checking: control that all files that are going to be submitted to the meta-analysis fulfill some formatting guidelines, including:

   - consistency of column names with meta-analysis guidelines;
   - presence of the minimal required information;
   - 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 information is present and unequivocal;
   - the number of chromosomes and chromosome coding are as expected;

2. Quality checking: evaluating the quality of data in each single file. This includes:

   - assessing the presence of unexpected values for some of the items required for the meta-analysis (e.g.: negative p-values or negative standard errors);
   - assessing p-value inflation and analyzing p-value distribution;
   - assessing the distribution of the main summary statistics, including the effect estimates, their standard errors, genotype imputation quality, etc.

3. Global checking: identification of any systematic bias that may affect the analysis. This step is aimed to uncover studies that are systematically different from the others. This may happen when, for instance, analysts of one study forgot to log-transform the phenotype or apply the wrong model to the data.

Formal and quality checks of individual studies are performed in *GWAtoolbox* using the *gwasqc()* function. *gwasqc()* was built to address specific requirements, specifically:

1. it allows rapid file processing and reporting;

2. it eliminates routine user operations;

3. it allows multi-format reporting, including *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.
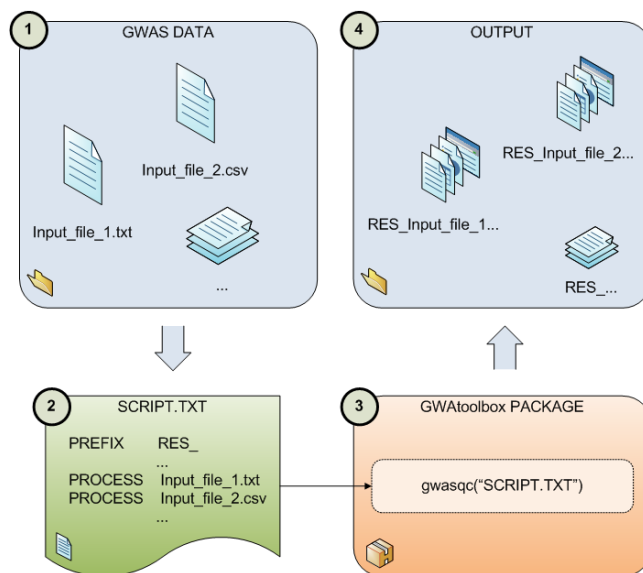

Figure 1: The quality control workflow.

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 between 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 *GWAtoolbox*, 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 estimate

- Standard error of the effect estimate

- P-value

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 to *gwasqc()* through a script in a text file. The format of the script file resembles the METAL input file format*.

In the input script, 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 when more GWAS files are in the same format, file specifications can be entered only once for all files. Example 1 illustrates the content of a hypothetical input script file.

Example 1

---

*http://www.sph.umich.edu/csg/abecasis/metal/

```
# Description of input data columns
MARKER          SNPID
CHR             Chromosome
POSITION        Position
N               n_total
ALLELE          coded_allele noncoded_allele
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 Listing the 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
> ```
>
> QC is applied first to *input_file_1.txt* and then to *input_file_2.csv*. When files reside on different directories, the full path must be specified. ◁

5.2 Describing the Input Data Columns

5.2.1 Field Separator

The field separator may be different for each GWAS data file. *gwasqc()* automatically detects the field separator of each input file *based on the first 10 rows.*

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

The user has the possibility to specify the separator manually for each GWAS file using the command **SEPARATOR**. Table 1 lists all supported separators.

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

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

Example 3 In the following input script:

```
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()*; the separator for the input files *input_file_2.csv* and *input_file_3.txt* is set to tabulation by the user. ◁

If the user manually specifies the wrong field separator, then the file(s) still will be processed with this separator. As a consequence, the mandatory columns will not be detected and the user will see it in the final report.

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 In the following input script:

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

the *hyphen* symbol is set as symbol for missing value for the file *input_file_1.txt*. Afterwards, the coding for missing vaules 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 default column names for the GWAS data files is reported. These names identify uniquely the items in the GWAS data file.

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 estimate
STDERR	Standard error of the effect estimate
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)
AVPOSTPROB	Average posterior probability for imputed marker allele dosage

Table 2: The default column names.

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 the effect estimate and its standard error are *beta* and *SE*, respectively. In *input_file_2.txt*, the column name for the effect estimate 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:

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

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

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, *gwasqc()* assumes that column names of GWAS input files are case insensitive. For example, the 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**. Table 3 lists all possible arguments.

Argument	Description
0	Column names are case insensitive (default)
1	Column names are case sensitive

Table 3: The list of arguments for the CASESENSITIVE command.

Example 7 Consider the following commands:

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

Column names of *input_file_1.txt* are case sensitive and must correspond exactly to the default column names; column names of *input_file_2.csv* are case insensitive. ◁

5.3 Specifying Data Filters

5.3.1 Implausible Values Filters

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, or the p-value are sometimes generated by the software used for the association testing. For example, in the case of a disease outcome with a small number of cases or of a SNP with very small MAF, statistical packages can report inconsistent results 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 data quality summary statistics.

Implausible value filters are used by *gwasqc()* to identify implausible values. Table 4 lists the columns for which the filters can be applied and their 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 value filters.

The default thresholds can be redefined for every column in the input script. The new thresholds 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 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 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 and new $(0, 1)$ thresholds 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. ◁

5.3.2 High Quality Filters

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

The default thresholds can be redefined with the command **HQ_SNP** for every input file in the input script. The command is followed by two values:

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

Table 5: The default high quality imputation filters.

the first one is the threshold for the MAF and the one is the threshold for the imputation quality.

Example 9 If we want to define 'high quality SNPs' those with MAF > 0.03 and 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 Filters

The plotting filters are used to select meaningful data to be displayed in the various summary plots. Each filter allows 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 MAF 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
```

Here, new SNP quality thresholds are set for plotting results from *input_file_1.txt*. For the first level thresholds, we have selected MAF > 0.02 and the imputation quality > 0.3, for the second level threshold, MAF > 0.03 and imputation quality > 0.5. ◁

5.4 Controlling The Output

5.4.1 Output File Names

Output file names are defined based on input file names, with the addition of a specified prefix (all types of output files will share the same prefix). 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
```

All result output files corresponding to the input files *input_file_1.txt* and *input_file_2.csv* will be prefixed with *res_*; result output files corresponding to the input file *input_file_3.tab* will be prefixed with *result_*. ◁

5.4.2 Verbosity Level For Graphical Output

The user can control the number of output pictures with the command **VERBOSITY** (see Table 7 for the available options).

Argument	Description
1	Lowest verbosity level (default).
2	Highest verbosity level.

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

Example 12 The input script contains the following commands:

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

The file *input_file_1.txt* is processed with the highest verbosity level and therefore all figures are generated; *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 information in the input files. Figure 2 shows all the dependencies. If some dependency is not satisfied because of missing data or filter setting, then the corresponding plot is not produced or may be truncated at different levels.

Columns \ Plots	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 dependencies of graphical outputs on columns and filters.

When multiple files are processed, boxplots from the distributions of the effect estimates are displayed in a single graph for across-study comparison. It is possible to specify the width of each box 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 define labels for each 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*. Consider the following input script:
>
> ```
> N              n_total
> PROCESS        input_file_1.txt first
> N              samplesize
> PROCESS        /dir_1/dir_2/input_file_2.csv second
> BOXPLOTWIDTH   N
> ```
>
> 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 typical output of the *gwasqc()* function consists of the following files:

1. **Graphical files** (*PNG* file extension) include **histograms** and **boxplots** of the distribution of the main statistics from each GWAS file: effect estimates, imputation quality index, sample size, p-value, allele frequency;

QQ-plots of the p-value distribution are also provided to investigate the presence of study-design bias. See Figure 2 for an exhaustive list of available plots.

2. **Textual report** (*TXT* file extension) contains information on the GWAS file-format quality and statistics summarizing the distribution of all data present in the GWAS files (effect estimates, p-values, etc.). The statistics provided in this report can be compared with the graphical output described above.

3. **Comma separated report** (*CSV* file extension) contains summary statistics for the high quality SNPs, as they have been defined by the user when setting the parameter **HQ_SNP** in the QC script (see Section 5.3.2). The tabular format of this file is intended to be useful for users who wants to perform additional analyses and compare results from different GWAS files without having to manage the large original GWAS files. Figure 3 shows an example of such file. For each of the variables listed by

	N	EFFECT	STDERR	PVALUE	MAF
N	2543887	2542617	2542422	2542617	2543887
N_HQ	2448544	2448544	2448544	2448544	2448544
N_NAs	0	1270	1270	1270	0
Mean	516	-0.00137	0.295076	0.500823	0.230416
StdDev	0	0.333379	0.158119	0.28873	0.143312
Min	516	-8.76038	0.17353	7.34E-08	0.010001
Max	516	6.12165	2.53341	1	0.5
Median	516	-0.00149	0.233707	0.502629	0.217008
Skewness	nan	-0.06933	2.886758	-0.00935	0.214034
Kurtosis	nan	9.109873	11.14145	-1.20085	-1.18454

	IMPUTED	IMP_QUALITY	CALLRATE	HWE_PVAL	STD_EFFECT_0.5
N	2543887	2543887	2543887	NA	NA
N_HQ	2448544	2448544	2448544	NA	NA
N_NAs	0	0	0	NA	NA
Mean	1	0.958868	1	NA	NA
StdDev	0	0.097734	0	NA	NA
Min	1	0.300006	1	NA	NA
Max	1	1.0467	1	NA	NA
Median	1	0.994577	1	NA	NA
Skewness	nan	-3.90226	nan	NA	-0.00587
Kurtosis	nan	16.76683	nan	NA	3.297394

	STD_EFFECT_0.75	STD_EFFECT_0.95	STD_EFFECT_0.99	STD_EFFECT_1
N	NA	NA	NA	NA
N_HQ	NA	NA	NA	NA
N_NAs	NA	NA	NA	NA
Mean	NA	NA	NA	NA
StdDev	NA	NA	NA	NA
Min	NA	NA	NA	NA
Max	NA	NA	NA	NA
Median	NA	NA	NA	NA
Skewness	-0.01036	0.022903	0.040024	-0.07186
Kurtosis	3.822136	4.712606	5.861049	9.066722

Figure 3: The comma separated report file.

columns, summary statistics are reported by row. Specifically:

- **N** – number of SNPs with available information (i.e. non-missing values).

- **N_HQ** – number of high quality SNPs with information available (i.e. non-missing values).

- **N_NAs** – number of SNPs with missing values for the specific field of interest.

- **Mean**, **StdDev**, **Min**, **Max**, **Median**, **Skewness**, and **Kurtosis** are referred to the distribution of the specific field.

Columns include:

- **N** – study sample size.
- **EFFECT**, **STDERR**, and **PVALUE** – summaries of the SNP-phenotype associations.
- **MAF**, **IMPUTED**, **IMP_QUALITY**, **CALLRATE**, and **HWE_PVAL** – summaries of the genotype distribution and quality.
- **STD_EFFECT_0.5**, **STD_EFFECT_0.75**, **STD_EFFECT_0.95**, **STD_EFFECT_0.99**, **STD_EFFECT_1** – these columns are only of interest for the reported skewness and kurtosis of the standardized effect estimates (beta/SE). The numbers 0.5, 0.75, 0.95, 0.99, and 1 are referred to the percentage of SNPs with the highest p-values chosen to study the effect size distribution (see Section 9.1 for additional details).

4. **An HTML report** (*main.html* file) combines both textual and graphical output, allowing the user to easily surfing across the results and across the studies included in the QC analysis. In operating systems allowing graphical interfaces, the *HTML* document should be the first file to be opened to investigate the results.

7 Example

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

Consider the following five GWAS data files: *gwa_data_example_1.txt*, *gwa_data_example_2.tbl*, *gwa_data_example_3.csv*, *gwa_data_example_4.txt* and *gwa_data_example_5.csv*. The first file contains 16 whitespace-separated columns:

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

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

The second file contains 16 tab-separated columns:

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

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

Analogously, we can preview the headers of the other three files. To perform the QC of these files we prepare a simple input script: *GWAS_script.txt*. Below are listed the 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 MAF and imputation quality
MAF 0.01    0.05
IMP 0.3     0.5


# Prefix for output files
PREFIX          gwasqc_


# Column N controls the width of boxplots
BOXPLOTWIDTH        N


# Input file and its short name for labels
PROCESS gwa_data_example_1.txt Study1


PVALUE              p
STDERR              StdErr


PROCESS gwa_data_example_2.tbl Study2
PROCESS gwa_data_example_3.csv Study3


PVALUE              pvalue


PROCESS gwa_data_example_4.txt Study4
PROCESS gwa_data_example_5.csv Study5
```

When the input script is ready, load the *GWAtoolbox* library and call the *gwasqc()* function as follows:

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

8 Parallel Processing

gwasqc() processes GWAS data files sequentially, one after another. It makes possible to handle very large files on desktop machines with relatively small amounts of available main memory. *pgwasqc()* function is analogous to *gwasqc()* and processes GWAS data files in parallel. The processing of several files is distributed among available CPUs at an additional cost of main memory. The computational time of *pgwasqc()* highly depends on the configuration of file storage and synchronization of I/O operations.

9 Between-study comparisons

9.1 Comparing skewness and kurtosis of effect size distribution

Association of genetic markers with continuous or binary phenotypes is generally assessed by the use of linear models, where an effect estimate and its standard error are used to summarize the evidence of the association. The effect estimate is usually represented by the beta coefficient of the linear regression model. For binary outcome, this correspond to the log(odds ratio) obtained from logistic regression models. Let's define with θ our parameter of interest, and with $SE(\theta)$ its standard error. For large sample sizes, under the null hypothesis of no association, the distribution of $\theta/SE(\theta) \sim N(0,1)$.

In checking the quality of GWA results, we are interested in assessing potential errors arisen during the analytical process or during the file management process. These errors could origin $\theta/SE(\theta)$ distributions that are systematically biased towards positive (or negative) values, or that are over/under-dispersed. The *kurtosis* and the *skewness* indices are the natural candidates to perform this kind of assessment. Convenient graphical display based on these two indices enables the contemporary plot and comparison of all the studies involved in the meta-analysis, with consequent identification of studies that are systematically different from each other.

Under the forms proposed by Cramer [3], the kurtosis and the skewness indices can be defined as $ku = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^2} - 3$ and $sk = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{\frac{3}{2}}}$. The skewness index assesses the symmetry of a distribution around its central value, and the kurtosis index assesses the dispersion of the distribution around its central value. If $\theta/SE(\theta) \sim N(0,1)$, then for large sample sizes, $sk_\theta = sk\big(\theta/SE(\theta)\big) \to 0$ and $ku_\theta = ku\big(\theta/SE(\theta)\big) \to 0$ (Fisher [4]; Joanes and Gill [5]).

In a GWAS setting, we can assume that the 50% of SNPs with largest p-value are not associated with the phenotype of interest and so, they can be used to represent the null situation. Notice that the 50% of SNPs with worst p-values correspond to the concept of the genomic control inflation factor, which is estimated based on the median chi-square distribution from the p-values.

In real world applications, distribution of $\theta/SE(\theta)$ for 50% worst SNPs is not normally distributed, because SNPs are not independent each other. This situation is more pronounced in presence of genotype imputation, which causes an excess of effect estimetes that are close to 0. For this reason, it is not realistic

to expect that $ku_\theta \to 0$: the distribution is leptokuric, and so $ku_\theta > 0$. However, as long as the studies involved in the meta-analysis used similar imputation reference platforms, ku_θ should be similar for all studies. For what concerns the skewness, there is no good reason why sk_θ of the 50% worst SNPs shouldn't approximate 0.

Then, for given K studies, we can estimate $ku_{\theta,k}$ and $sk_{\theta,k}$ for $k = 1, \ldots, K$, and we can plot the two vectors $sk_{\theta,[1,\ldots,K]}$ vs. $ku_{\theta,[1,\ldots,K]}$ in a Cartesian diagram, with every point representing a different study. We expect all studies to cluster around the same point at $sk = 0$, with similar kurtosis values. Studies that show strong departures from the main cluster could be submitted to detailed investigation in order to detect the reason of such discrepancy. In general, departures along the sk-axis are more serious than departures on the ku-axis, because the first ones will introduce systematic bias in the meta-analysis with one or a few studies that are systematically different from the others, with effect estimates that are more often in one direction.

This diagnostic plot is shown in Figure 4 where the set of SNPs with 50% largest p-values is compared with other sets of SNPs with the largest 75%, 95%, and 100% (i.e. all SNPs are considered) p-values, respectively. Highlighted are points (studies) that are largely difference from the ones in the cluster. In the last scenario, all SNPs are included and the bias is given by the SNPs that are truly associated with the phenotype. Outlier studies, that have been identified in the 50% scatterplot, are colored in red also in the other situations for comparison.
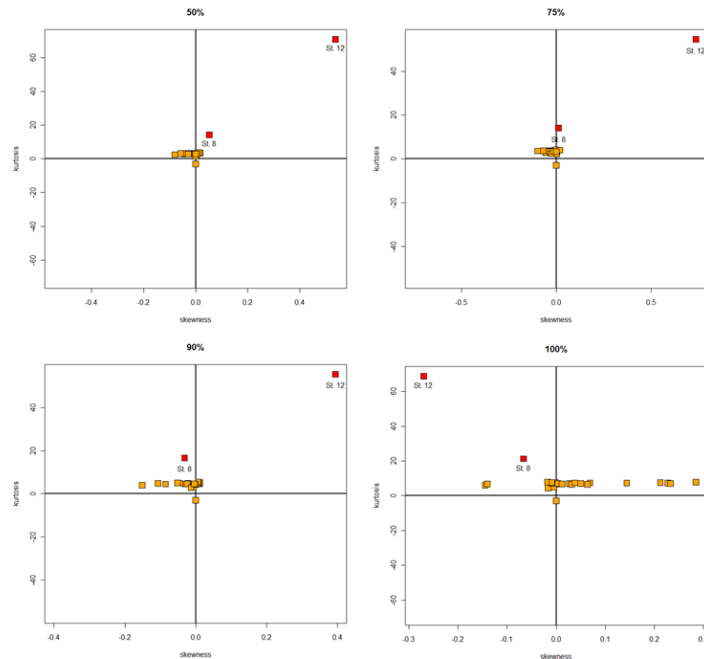

Figure 4: The skewness and kurtosis plot.

The *GWAtoolbox* allows automatic comparison of skewness and kurtosis of

effect size distribution between GWA studies. The *gwasqc()* function estimates the skewness and kurtosis statistics during the QC workflow and includes them into the *CSV* reports. Then, the auxiliary *kusk_check()* function can be used to export this information to a *R* data frame and to produce diagnostic plots. As input, it requires the same script used for *gwasqc()* and assumes that all the *CSV* reports are located in the current working directory. An optional list consisting of any of integer number among 50, 75, 95, 99, and 100, can be specified: numbers correspond to the percentage of SNPs to be considered as representing the null distribution.

Example 14 We report the commands to obtain the scatterplot shown in Figure 4, when the 50% SNPs with largest p-values is considered.

```
> W <- kusk_check("GWASQC_script.txt", worst = c(50), plot = TRUE)
> points(W$sk50[W$ku50 > 5], W$ku50[W$ku50 > 5], pch = 22, bg = 2, cex = 2)
> text(W$sk50[W$ku50 > 5], W$ku50[W$ku50 > 5], labels = W$study[W$ku50 > 5], cex = 1,
◁
```

Currently, no automatic method to identify outlier studies is implemented and that the user needs to define his/her own criteria for the outlier identificaton.

9.2 Precision of the effect estimates by sample size

A different graphical test, that allows the comparison of studies against each other, is based on the assessment of the distribution of estimates' precision vs. study sample size. In general, the average $SE(\theta)$ is expected to be inversely proportional to the study sample size. The auxiliary *dispersion_check()* function plots a scatterplot of the $mean(SE(\theta))$ vs. the median sample size of all studies, as depicted in the example Figure 5. Over-dispersion is defined as the presence of larger SEs than expected given the study sample size and under-dispersion is meant to be the opposite phenomenon. For example, a study with unmodeled relatedness or population stratification may present SEs that are smaller than another study of similar sample size where these issues were accounted for properly.

The *dispersion_check()* function uses the *CSV* reports generated by *gwasqc()*. As input it requires the same script as *gwasqc()* and assumes that all the *CSV* reports files are located in the current working directory. If the study sample size is missing from GWAS files, then the function has an optional parameter allowing to specify a vector with all study sample sizes. The function returns an *R* data frame with the information extracted from the *CSV* reports and produces the diagnostic plot.

Example 15

```
> Z <- dispersion_check("GWASQC_script.txt", plot=TRUE)
> Z

   study    mean_se median_n
1 Study1 0.01188491    1201
2 Study2 0.03206312     201
3 Study3 0.01270057    1000
```
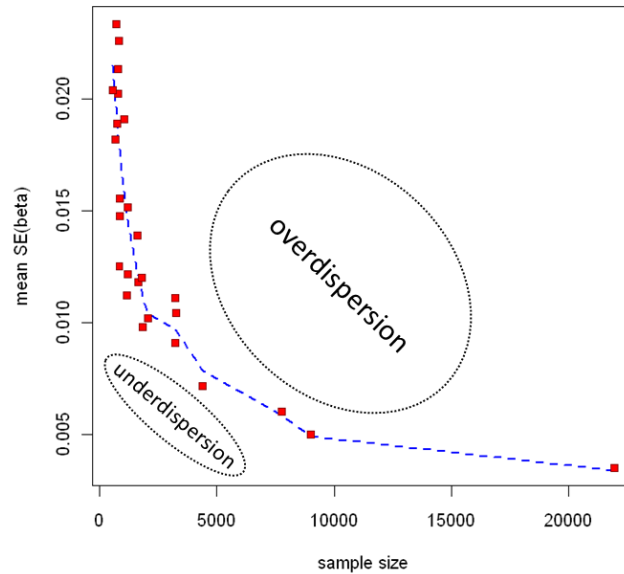
Figure 5: Schematic representation of the dispersion plot and its interpretation.

```
4 Study4 0.02030638      437
5 Study5 0.01532013      721

> text(Z$median_n, Z$mean_se, labels=Z$study, pos=c(2,4,2,1,2))

> Z <- dispersion_check("GWASQC_script.txt", sample_sizes=c(1200, 200, 1000, 500, 700,

> Z

    study    mean_se median_n
1 Study1 0.01188491     1200
2 Study2 0.03206312      200
3 Study3 0.01270057     1000
4 Study4 0.02030638      500
5 Study5 0.01532013      700

> text(Z$median_n, Z$mean_se, labels=Z$study, pos=c(2,4,2,1,2))

◁
```

10 Additional Tools

10.1 GWAS Data Files Formatting

The format of GWAS data files may differ between studies. Inconsistencies in column names, column separators and column ordering are the most common. For the automated data analysis it is more convenient to have a uniform format

across all the GWAS files of interest. The *gwasformat()* function and its parallel version *pgwasformat()* allow to quickly transform GWAS data files into a uniform format. In addition to the possiblity to rename and re-order columns, these functions calculate effective sample size, inflation factor and apply genomic control of P-values and standard errors. The column separator is replaced with tabulation in all processed files.

10.1.1 Input Script

The formatting instructions are provided to *gwasformat()* through the input script that is identical to the script required by *gwasqc()*. Additional commands are supported allowing to rename and re-order columns, enable/disable genomic control. Example 16 illustrates the content of a hypothetical input script file.

Example 16

```
# Rename columns
RENAME  SNPID     rsId
RENAME  chr       chrom
RENAME  position  bp
RENAME  SE        StdErr
RENAME  pval      p-value
RENAME  p         p-value
RENAME  pvalue    p-value

# Description (meaning) of the columns
ALLELE        coded_all noncoded_all
CALLRATE      callrate
CHR           chrom
EFFECT        beta
FREQLABEL     AF_coded_all
HWE_PVAL      HWE_pval
IMPUTED       imputed
IMP_QUALITY   oevar_imp
MARKER        rsId
N             n_total
POSITION      bp
PVALUE        p-value
STRAND        strand_genome
STDERR        StdErr
USED_FOR_IMP  used_for_imp
AVPOSTPROB    avpostprob

# Enable computation of inflation factor and genomic control
GC ON

# Set filters on minor allele frequency and imputation quality
HQ_SNP 0.05 0.4

# Enable column re-ordering and specify the order explicitly
ORDER ON chrom bp rsId strand_genome coded_all noncoded_all
```

```
# Prefix for output files
PREFIX pgwasformat_

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

◁

10.1.2 Renaming Columns

The new column names are specified in the input script with the command
RENAME. The command is followed by two words: the first one corresponds
to the column name in the original GWAS file, and the second one corresponds
to the new column name in the resulting formatted GWAS file. The column
names must not contain tabulation or whitespace characters. If column was
renamed, then only the new column name must be used in the rest of the input
script commands.

> **Example 17** Let's assume to have three input files: *input_file_1.txt*,
> *input_file_2.csv* and *input_file_3.txt*. The files have column named
> *marker*. In file *input_file_1.txt* the column should be renamed to *SNPID*.
> While in files *input_file_2.csv* and *input_file_3.txt* it should be renamed
> to *rsId*. The correct column renaming is as follows:
>
> ```
> RENAME   marker SNPID
> PROCESS  input_file_1.txt
> RENAME   marker rsId
> PROCESS  input_file_2.csv
> PROCESS  input_file_3.txt
> ```
>
> ◁

10.1.3 Ordering Columns

By default, *gwasformat()* doesn't change column ordering followed in the original
GWAS file. This behaviour can be modified for every input file in the input
script using the command **ORDER**. Table 8 lists all possible arguments.

Argument	Description
OFF	The original column ordering is preserved in the resulting formatted GWAS file (default)
ON	Columns are re-ordered following the alphabetical ordering
ON column_1 column_2 ...	Columns are re-ordered following the specified order: *column_1 column_2 ... * .

Table 8: The list of arguments for the ORDER command.

Example 18 Let's assume to have three input files: *input_file_1.txt*, *input_file_2.csv* and *input_file_3.txt*. Each file contains the following columns in the order as they are listed: *marker*, *chromosome*, *bp*. Below are provided commands to rename the column *marker* to *SNPID* and to switch the ordering mode for every input file:

```
RENAME   marker SNPID
ORDER    ON chromosome bp SNPID
PROCESS  input_file_1.txt
ORDER    OFF
PROCESS  input_file_2.csv
ORDER    ON
PROCESS  input_file_3.txt
```

For the input file *input_file_1.txt* the columns are re-ordered to: *chromosome*, *bp*, *SNPID*. For the input file *input_file_2.csv* the original column ordering is preserved: *SNPID*, *chromosome*, *bp*. For the input file *input_file_3.txt* the columns are re-ordered following the alphabetical ordering: *bp*, *chromosome*, *SNPID*. ◁

10.1.4 Filtering

gwasformat() filters SNPs based on minor allele frequency (MAF) and imputation quality. Table 9 lists the default thresholds. The default threshold values

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

Table 9: The default SNP filters.

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 19 If we want to filter SNPs with MAF > 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
```

◁

Example 20 If we want to disable filtering, we would change the input script as follows:

```
HQ_SNP   0 0
PROCESS  input_file_1.txt
```

◁

10.1.5 Inflation Factor and Genomic Control

By default, *gwasformat()* doesn't calculate the inflation factor and doesn't apply genomic control of P-values and standard errors. This behaviour can be modified for every input file in the input script using the command **GC/GENOMICCONTROL**. Table 10 lists all possible arguments. The in-

Argument	Description
OFF	The inflation factor is not calculated and genomic control is not applied (default)
ON	The inflation factor is calculated. Genomic control of *PVALUE* and *STDERR* columns is applied, corrected value are saved to the new columns *PVALUE_gc* and *STDERR_gc*, accordingly. Has no effect if *PVALUE* column is not present.
numeric value	The inflation factor is assumed to be equal to the specified *numeric value*. Values in *PVALUE* and *STDERR* columns are corrected and saved to the new columns *PVALUE_gc* and *STDERR_gc*, accordingly.

Table 10: The list of arguments for the GC/GENOMICCONTROL command.

flation factor is calculated using only filtered SNPs. If calculated or explicitly specified inflation factor is less than 1.0, then genomic control of P-values and standard errors is not applied, new columns *PVALUE_gc* and *STDERR_gc* are filled with corresponding original values.

> **Example 21**
>
> ```
> RENAME pval pvalue
> RENAME se   stderr
> PVALUE pvalue
> STDERR stderr
> GC     ON
> PROCESS input_file_1.txt
> GC     OFF
> PROCESS input_file_2.csv
> GC     1.1
> PROCESS input_file_3.txt
> ```
>
> ◁

10.1.6 Effective Sample Size

By default, the *gwasformat()* computes the effective sample size based on *IMP_QUALITY* and *N* columns. The computed values are saved to the new column *N_effective*.

10.2 GWAS Data Files Annotation

Often there is a need to annotate markers in GWAS data files with regions (e.g. genes). The *annotate()* function and its parallel version *pannotate()* allow to

quickly annotate every marker in GWAS data files with regions that contain it or fall in a specified windows around it (e.g. +/-50kb, +/-100kb and etc). The arbitrary number of windows of various sizes can be specified. The regions with their chromosomal coordinates must be provided in a separate file. Therefore, it is responsibility of an analyst to prepare the list of regions of interest with chromosomal positions on required human genome build version. It is possible to annotate markers if only their names are available in GWAS data files, or if there is a need to change chromosomal positions (e.g. if different version of human genome build should be used). In this case, their chromosomal positions must be provided in a separate map file.

10.2.1 Input Script

The annotation instructions are provided to *annotate()* through the input script that is identical to the script required by *gwasqc()*. Additional commands are supported allowing to specify files with regions for annotation and map files with chromosomal coordinates for markers, if needed. Example 22 illustrates the content of a hypothetical input script file.

Example 22

```
# Description (meaning) of the columns in input files
# with GWAS data to be annotated.
MARKER     SNPID
CHR     chr
POSITION    position

# File with regions.
REGIONS_FILE genes_HGNC_ensembl_rel54_may2009_b36.txt

# Description (meaning) of the columns in file with regions.
REGION_NAME     Gene
REGION_CHR     Chromosome
REGION_START    Start
REGION_END     End

# Window sizes around markers.
REGIONS_DEVIATION   0 10000 25000

# Preserve original columns in the output file and append
# columns with annotated regions to the end.
REGIONS_APPEND   ON

# Specify prefix for output files with annotation results.
PREFIX   annotated_

# Input file with GWAS data to annotate.
PROCESS   gwa_data_example_1.txt

# Next file with GWAS data will be annotated using marker
# chromosomal positions in the external map file.
```

```
MAP_FILE    map_b37.txt

# Description (meaning) of the columns in map file.
MAP_MARKER     snpid
MAP_CHR        chr_b37
MAP_POSITION   pos_b37

# Change the regions file.
REGIONS_FILE   genes_HGNC_ensembl_rel67_may2012_b37.txt

# Change window sizes around markers.
REGIONS_DEVIATION   0 25000 50000

# Output only columns with annotated regions, marker
# names and their chromosomal positions.
REGIONS_APPEND   OFF

# Another input file with GWAS data to annotate.
PROCESS   gwa_data_example_2.tbl
```

◁

10.2.2 Specifying The Input Data Files

The names of the GWAS data files are specified in the input script with the command **PROCESS** (one line per file). A different directory path can be specified for each file.

Example 23

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

The annotation is applied first to *input_file_1.txt* and then to *input_file_2.csv*. ◁

10.2.3 Specifying The Regions Files

The names of the regions (e.g. with genes) files are specified in the input script with the command **REGIONS_FILE**. In the same script different regions files can be specified for different GWAS data files. Also different directory path can be specified for each regions file.

Example 24

```
REGIONS_FILE genes_file_1.txt\cr
PROCESS input_file_1.txt\cr
REGIONS_FILE /dir_1/dir_2/genes_file_2.csv\cr
PROCESS input_file_2.csv\cr
PROCESS input_file_3.txt
```

The annotation is applied first to *input_file_1.txt* using regions from *genes_file_1.txt* file. Then, files *input_file_2.csv* and *input_file_3.txt* are annotated with regions in *genes_file_2.csv* file. ◁

10.2.4 Specifying The Map Files

The names of the map files are specified in the input script with the command **MAP_FILE**. In the same script different map files can be specified for different GWAS data files. Also different directory path can be specified for each map files.

Example 25

```
MAP_FILE map_file_1.txt
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
MAP_FILE /dir_1/dir_2/map_file_2.csv
REGIONS_FILE /dir_1/dir_2/genes_file_2.csv
PROCESS input_file_2.csv
PROCESS input_file_3.txt
```

The annotation is applied first to *input_file_1.txt* using marker genomic positions in *map_file_1.txt* file and regions in *genes_file_1.txt* file. Then, files *input_file_2.csv* and *input_file_3.txt* are annotated with regions in *genes_file_2.csv* file using marker genomic positions in *map_file_2.csv*. ◁

10.2.5 Specifying Column Names in Input Data Files

In table 11 the complete list of default column names that must be present in the GWAS data files for annotation is reported. These names identify uniquely the items in the GWAS data file for annotation purposes.

Default column name(s)	Description
MARKER	Marker name
CHR	Chromosome number or name
POSITION	Marker position

Table 11: The default column names for annotation.

Given that different names can be provided for each GWAS data file, *annotate()* 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 the new column name. When the map file is specified using command **MAP_FILE**, then *CHR* and *POSITION* columns in the GWAS data file are not required.

Example 26 Let's assume to have two input files, *input_file_1.txt* and *input_file_2.csv*. In the *input_file_1.txt*, the column names for marker name, chromosome name and position are *SNPID*, *CHR* and *POS*, respectively. In the *input_file_2.csv*, the column names for marker name is the same as in *input_file_1.txt*, but the column names for the chromosome and position are *chromosome* and *position*, respectively. The correct column redefinition is as follows:

```
MARKER SNPID
POSITION POS
```

```
PROCESS input_file_1.txt
CHR chromosome
POSITION position
PROCESS input_file_2.csv
```

There are no need to define the *CHR* field for the *input_file_1.txt*, since it matches the default name. ◁

10.2.6 Specifying Column Names in Regions Files

In table 12 the complete list of default column names for the regions file is reported. These names identify uniquely the items in the regions file.

Default column name(s)	Description
REGION_NAME	Region name (e.g. gene name)
REGION_CHR	Chromosome number or name
REGION_START	Region (e.g. gene) start position
REGION_END	Region (e.g.) end position

Table 12: The default column names in regions file.

Given that different names can be provided for each regions file, *annotate()* allows to redefine the default values for every regions file in the input script. The redefinition command consists of the default column name followed by the present column name.

Example 27 Let's assume to have two map files, *region_file_1.txt* and *region_file_2.csv*. In the *region_file_1.txt*, the column names for the region name, chromosome, start and end position are *name*, *chr*, *REGION_START* and *REGION_END*, respectively. In the *region_file_2.csv*, the column name for the region name and chromosome are the same as in *regions_file_1.txt*, but the column names for the region start and end positions are *start* and *end*, respectively. The correct column redefinition is as follows:

```
REGIONS_FILE genes_file_1.txt
REGION_NAME name
REGION_CHR chr
PROCESS input_file_1.txt
REGIONS_FILE genes_file_2.csv
REGION_START  start
REGION_END end
PROCESS input_file_2.csv
```

There is no need to define the *REGION_START* and *REGION_END* fields for *genes_file_1.txt* regions file. Also there is no need to redefine *REGION_NAME* and *REGION_CHR* fields for the *genes_file_2.csv* map file. ◁

10.2.7 Specifying Column Names in Map Files

In table 13 the complete list of default column names for the map file is reported. These names identify uniquely the items in the map file.

Default column name(s)	Description
MAP_MARKER	Marker name
MAP_CHR	Chromosome number or name
MAP_POSITION	Marker position

Table 13: The default column names in map file.

Given that different names can be provided for each map file, *annotate()* allows to redefine the default values for every map file in the input script. The redefinition command consists of the default column name followed by the present column name.

Example 28 Let's assume to have two map files, *map_file_1.txt* and *map_file_2.csv*. In the *map_file_1.txt*, the column names for marker name, chromosome and position are *name*, *MAP_CHR* and *pos*, respectively. In the *map_file_2.csv*, the column name for the marker name and chromosome are the same as in *map_file_1.txt*, but the column name for the marker position is *map_pos*. The correct column redefinition is as follows:

```
MAP_FILE map_file_1.txt
MAP_MARKER name
MAP_POSITION pos
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
MAP_FILE map_file_2.csv
MAP_POSITION map_pos
REGIONS_FILE genes_file_2.csv
PROCESS input_file_2.csv
```

There is no need to define the *MAP_CHR* field for both map files. Also there is no need to redefine *MAP_MARKER* for the *genes_file_2.csv* map file. ◁

10.2.8 Specifying Window Size For Annotation

Every marker in the GWAS data file is annotated with the regions (e.g. genes) that fall in a particular window around it. *annotate()* allows to specify multiple window sizes using command **REGIONS_DEVIATION**. Command **REGIONS_DEVIATION** is followed by an arbitrary number of positive integers that specify window sizes around markers in base pairs. Each specified window size results in a new output column where all regions overlapping with this window are reported. The ouput columns are ordered by window size starting with the smallest. Therefore, every new output column represents bigger window size and lists only those regions that were not reported previously. If **REGIONS_DEVIATION** is not specified, then the default window sizes are

0, 100000 and 250000 (i.e. 0, +/-100kb and +/- 250kb around marker). If 0 is specified, then only regions that include the marker are reported.

Example 29

```
REGIONS_FILE genes_file_1.txt
REGIONS_DEVIATION 0 50000 100000
PROCESS input_file_1.txt
REGIONS_DEVIATION 0 100000 250000 500000
PROCESS input_file_2.csv
```

Every marker in *input_file_1.txt* will be annotated with regions that contains it or are within +/-50kb and +/-100kb windows around it. While every marker in *input_file_2.csv* will be annotated with regions that contains it or are within +/-100kb, +/-250kb and +/-500kb windows around it. ◁

10.2.9 Specifying Output Format

Often GWAS data file contains many columns that are not required in the output files with annotation results. By default, in addition to columns with annotated regions, *annotate()* outputs only columns with marker name, chromosome name and position. This behaviour can be modified for every input file in the input script using the command **REGIONS_APPEND**. Table 14 lists all possible arguments.

Argument	Description
OFF	Only the original columns with marker name, chromosome name and position are preserved. Columns with annotated regions are appended to the end. (default)
ON	All the original columns are preserved and columns with annotated regions are appended to the end.

Table 14: The list of arguments for the REGIONS_APPEND command.

Example 30

```
REGIONS_FILE genes_file_1.txt
REGIONS_APPEND ON
PROCESS input_file_1.txt
REGIONS_APPEND OFF
PROCESS input_file_2.csv
```

◁

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.

[3] H. Cramer (1946) **Mathematical Methods of Statistics**. Princeton: Princeton University Press.

[4] R.A. Fisher (1930) **The moments of the distribution for normal samples of measures of departure from normality**. Proc. R. Soc. Series A 130:16-28.

[5] D.N. Joanes, C.A. Gill (1998) **Comparing Measures of Sample Skewness and Kurtosis**. J. Royal Stat. Soc. Series D (The Statistician) 47(1):183-189.

Index