

Package ‘GWAtoolbox’

May 28, 2012

Version 2.1.2

Date 2012-05-25

Title GWAS Quality Control

Author Daniel Taliun <Daniel.Taliun@eurac.edu>, Christian Fuchsberger <cfuchsb@umich.edu>, Cristian Pattaro <Cristian.Pattaro@eurac.edu>

Maintainer Daniel Taliun <Daniel.Taliun@eurac.edu>, Christian Fuchsberger <cfuchsb@umich.edu>

Description Controls for duplicated IDs, distribution of betas, SEs,etc.

Depends R (>= 2.13.1)

Suggests snow

SystemRequirements GNU GSL

License GPL (>= 3)

Repository CRAN

Repository/R-Forge/Project gwatoolbox

Repository/R-Forge/Revision 72

Date/Publication 2012-05-28 07:01:11

R topics documented:

annotate	2
dispersion_check	9
gwasformat	11
gwasqc	17
kusk_check	24
pannotate	25
pgwasformat	34
pgwasqc	40
Index	48

annotate *SNPs annotation with regions (e.g. genes).*

Description

For provided SNPs finds all genes in specified deviation around them (e.g. 0, +/- 50000, +/- 100000, ...).

Usage

```
annotate(script)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be annotated along with basic information from each individual study.
--------	--

Details

Function `annotate()` annotates every marker in input files with regions (e.g. genes) 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 in the input script. The regions with their chromosomal coordinates must be provided in a separate file. It is possible to annotate markers if only their names are available (e.g. rsId) in input 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.

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:

```
PROCESS input_file_1.txt
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'.

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:

```
REGIONS_FILE genes_file_1.txt
```

```
PROCESS input_file_1.txt
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 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.

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:

```
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’.

Specifying Column Names in Input Data Files

In the table below, the complete list of the default column names for the GWAS data file 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

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:

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.

Specifying Column Names in Regions Files

In the table below, the complete list of the 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

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:

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.

Specifying Column Names in Map Files

In the table below, the complete list of the 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

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:

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.

Field Separator in Input Data Files

The field (column) separator can be different for each GWAS data file. `annotate()` automatically detects the original separator field for each input file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

```

PROCESS input_file_1.txt
SEPARATOR COMMA
PROCESS input_file_2.csv
PROCESS input_file_3.txt

```

For the input file ‘input_file_1.txt’ the field separator is determined automatically by the program but, for the input files ‘input_file_2.csv’ and ‘input_file_3.txt’ the separator is manually set to comma by the user.

Field Separator in Regions Files

The field (column) separator can be different for each regions file. `annotate()` automatically detects the original separator field for each regions file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **REGIONS_FILE_SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

```

REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
REGIONS_FILE genes_file_2.csv
REGIONS_FILE_SEPARATOR COMMA
PROCESS input_file_2.csv
REGIONS_FILE genes_file_3.txt
PROCESS input_file_3.txt

```

For the regions file ‘genes_file_1.txt’ the field separator is determined automatically by the program but, for the regions files ‘genes_file_2.csv’ and ‘genes_file_3.txt’ the separator is manually set to comma by the user.

Field Separator in Map Files

The field (column) separator can be different for each map file. `annotate()` automatically detects the original separator field for each map file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **MAP_FILE_SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

```
MAP_FILE map_file_1.txt
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
MAP_FILE map_file_2.csv
MAP_FILE_SEPARATOR COMMA
REGIONS_FILE genes_file_2.csv
PROCESS input_file_2.csv
MAP_FILE map_file_3.txt
PROCESS input_file_3.txt
```

For the map file 'map_file_1.txt' the field separator is determined automatically by the program but, for the map files 'map_file_2.csv' and 'map_file_3.txt' the separator is manually set to comma by the user.

Case Sensitivity

By default the `annotate()` assumes that column names in the all specified files are case insensitive. For example, the column names *CHR*, *Chr*, and *chr* are all perfectly equivalent. This behaviour can be modified for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names, as specified below:

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

Example:

```
CASESENSITIVE 1
MAP_FILE map_file_1.txt
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
CASESENSITIVE 0
MAP_FILE map_file_2.csv
REGIONS_FILE genes_file_2.csv
PROCESS input_file_2.csv
```

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 output 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:

```
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.

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**. The supported arguments are listed below:

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

Example:

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

Output File Name

The output file names are created by adding a prefix to the input file names. The prefix is specified with the command **PREFIX**.

Example:

```
REGIONS_FILE genes_file_1.txt
PREFIX annotated_
PROCESS input_file_1.txt
PROCESS input_file_2.csv
PREFIX annot_
PROCESS input_file_3.tab
```

All the output files corresponding to the input files ‘input_file_1.txt’ and ‘input_file_2.csv’ will be prefixed with *annotated_*; the output files corresponding to the input file ‘input_file_3.tab’ will be prefixed with *annot_*.

Author(s)

Daniel Taliun, Christian Fuchsberger, Cristian Pattaro

Examples

```
# name of an input script
script <- "ANNOTATE_script.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

# run annotate() function
annotate(script)
```

dispersion_check *Effect estimates precision by sample size.*

Description

Provides an assessment of the distribution of estimates’ precision vs. the study sample size.

Usage

```
dispersion_check(script, sample_sizes = NULL, plot = TRUE)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be processed along with basic information from each individual study, and instructions for the quality check.
sample_sizes	A vector with all study sample sizes.
plot	Logical value indicating if the diagnostic plot should be produced.

Details

The function uses *CSV* reports generated by `gwasqc()` function. As input it requires the same script as `gwasqc()` function and assumes that all the reports are 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 a data frame with the information extracted from the *CSV* reports and produces the diagnostic plot.

Author(s)

Cristian Pattaro, Daniel Taliun, Christian Fuchsberger

Examples

```
# name of an input script
script <- "GWASQC_script.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

# run gwasqc() function
gwasqc(script)

# run dispersion_check() function
Z <- dispersion_check("GWASQC_script.txt", plot=TRUE)
text(Z$median_n, Z$mean_se, labels=Z$study, pos=c(2,4,2,1,2))
```

gwasformat	<i>Formatting of GWAS data files.</i>
------------	---------------------------------------

Description

Renames and re-orders columns, sets tabulation as field (column) separator, calculates inflation factors and applies genomic control in GWAS data files.

Usage

```
gwasformat(script, logfile)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be processed along with basic information from each individual study.
logfile	Name of a log file with processing output. The output contains calculated inflation factors, total number of markers and number of filtered markers.

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:

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

The formatting is applied first to 'input_file_1.txt' and then to 'input_file_2.csv'.

Field Separator

The field (column) separator can be different for each GWAS data file and during the formatting it is changed to *tabulation*. `gwasformat()` automatically detects the original separator field for each input file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

```

PROCESS input_file_1.txt
SEPARATOR COMMA
PROCESS input_file_2.csv
PROCESS input_file_3.txt

```

For the input file ‘input_file_1.txt’ the field separator is determined automatically by the program but, for the input files ‘input_file_2.csv’ and ‘input_file_3.txt’ the separator is manually set to comma by the user. After the formatting all three files will have tabulation as new field separator.

Renaming Columns

The original column names in the GWAS data files are renamed using the command **RENAME** in the input script. The command is followed by two words: the first one corresponds to the original column name, and the second one corresponds to the new column name. The column names can’t contain tabulation or space characters.

Example:

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 *marker*, which should be renamed. The new column name should be *SNPID* for ‘input_file_1.txt’, and *rsId* for ‘input_file_2.csv’ and ‘input_file_3.txt’. 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

```

Column Names

In the table below, the complete list of the default column names for the GWAS data file 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 <i>Rsq</i> , IMPUTE’s <i>properinfo</i> , ...
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)
AVPOSTPROB	Average posterior probability for imputed marker allele dosage

Given that different names can be provided for each GWAS data file, `gwasformat()` 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 present column name. To redefine the default column names for *coded* and *non-coded* alleles, the command **ALLELE** followed by two present column names is used. If the present column name was renamed to the new column name with the command **RENAME**, then the new column name must be used in the redefinition command.

Example 1:

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 P-value and standard error are *pval* and *SE*, respectively. In the 'input_file_2.csv', the column name for the P-value is the same as in 'input_file_1.txt', but the column name for the standard error is *STDERR*. The correct column redefinition is as follows:

```
PVALUE pval
STDERR SE
PROCESS input_file_1.txt
STDERR STDERR
PROCESS input_file_2.csv
```

There is no need to redefine the **PVALUE** field. Alternatively, if the column *pval* in 'input_file_1.txt' and 'input_file_2.csv' needs to be renamed to *p-value*, then the input script is as follows:

```
RENAME pval p-value
PVALUE p-value
PROCESS input_file_1.txt
STDERR STDERR
PROCESS input_file_2.csv
```

Example 2:

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
```

Columns Ordering

By default the `gwasformat()` doesn't change the original ordering of columns in the input file. This behaviour can be modified for every input file in the input script using the command **ORDER** as specified below:

Argument	Description
OFF	The original ordering of columns is preserved
ON	Columns are re-ordered following the alphabetical ordering
ON column_1 column_2 ... column_n	Columns are re-ordered following the specified order <i>column_1 column_2 ... column_n</i>

Example:

Let's assume to have three input files: 'input_file_1.txt', 'input_file_2.csv' and 'input_file_3.txt'. Each file contains columns *marker*, *chromosome* and *bp* in the order as they are listed. The following input script renames the column *marker* to *SNPID* and switches the ordering mode for every input file:

```

RENAME marker SNPID
MARKER SNPID
CHR chromosome
POSITION bp
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 ordering of columns 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*.

Case Sensitivity

By default the `gwasformat()` 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 modified for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names, as specified below:

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

Example:

```

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

```

Specifying Filters

The `gwasformat()` filters SNPs based on minor allele frequency(MAF) and imputation quality. The default thresholds are listed below:

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

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 1:

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 2:

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

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

Inflation Factor and Genomic Control

By default the `gwasformat` doesn't calculate the inflation factor and doesn't apply the genomic control. This behaviour can be modified for every input file in the input script using the command **GC/GENOMICCONTROL** as specified below:

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

If the inflation factor value is less than 1.0, then the genomic control is not applied.

Example:

```
GC ON
PROCESS input_file_1.txt
GC OFF
PROCESS input_file_2.csv
GC 1.1
PROCESS input_file_3.txt
```

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*.

Output File Name

The output file names are created by adding a prefix to the input file names. The prefix is specified with the command **PREFIX**.

Example:

```
PREFIX res_
PROCESS input_file_1.txt
PROCESS input_file_2.csv
PREFIX result_
PROCESS input_file_3.tab
```

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

The Output Files

`gwasformat` produces one formatted (renamed/re-ordered columns, genomic control correction and etc.) copy of every original GWA data file. The formatting history information, containing calculated inflation factors and number of filtered markers, is saved to the log file under the provided *logfile* name.

Author(s)

Daniel Taliun, Christian Fuchsberger, Cristian Pattaro

Examples

```
# name of an input script
script <- "GWASFORMAT_script.txt"

# name of a logfile
logfile <- "gwasformat_log.txt"
```

```
# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

# run gwasformat() function
gwasformat(script, logfile)
```

gwasqc

Quality Control Of GWA Data

Description

Performs the quality control of data from Genome-Wide Association Studies (GWAS).

Usage

```
gwasqc(script)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be processed along with basic information from each individual study, and instructions for the quality check.
--------	--

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:

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

The QC is applied first to 'input_file_1.txt' and then to 'input_file_2.csv'.

Field Separator

The field (column) separator can be different for each GWAS data file. `gwasqc()` 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**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

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

For the input file 'input_file_1.txt' the field separator is determined automatically by the program but, for the input files 'input_file_2.csv' and 'input_file_3.txt' the separator is manually set to tabulation by the user.

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:

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

The *hyphen* symbol identifies missing values in the input file 'input_file_1.txt' and *NA* identifies missing values in the input file 'input_file_2.txt'.

Column Names

In the table below, the complete list of the default column names for the GWAS data file 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 <i>Rsq</i> , IMPUTE's <i>properinfo</i> , ...
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)
AVPOSTPROB	Average posterior probability for imputed marker allele dosage

Given that different names can be provided for each GWAS data file, `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 the new column name. To redefine the default column names for *coded* and *non-coded* alleles, the command **ALLELE** followed by two new column names is used.

Example 1:

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 effect size and standard error are *beta* and *SE*, respectively. In the 'input_file_2.csv', 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 redefinition is as follows:

```

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

```

There is no need to redefine the **EFFECT** field.

Example 2:

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

```

Case Sensitivity

By default the `gwasqc()` 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 modified for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names, as specified below:

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

- 1 Column names in the input file
are case sensitive

Example:

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

Filter for Implausible Values

Often, there is the necessity to identify implausible values, to exclude unreliable results from the meta-analysis. Implausible values can happen due to data sparseness, errors in the data handling, or other causes.

`gwasqc()` identifies SNPs with suspicious statistics (p-value, standard error, etc.) by applying appropriate threshold values. After the data processing, a detailed report including the number of SNPs with implausible statistics and the nature of the problem is produced. In addition, suspicious SNPs are excluded from the calculation of the summary statistics on data quality.

The default filter thresholds are listed below:

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)

The user has the option to modify the thresholds to account for specific needs. The new thresholds can be specified after the redefinition of the column name.

Example:

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 re-define 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) 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.

High Quality Filters

SNPs with low imputation quality and with too small minor allele frequency (MAF) could make spuriously small p-values happen. Checking for the presence of cryptic relatedness or hidden population sub-structure through the estimation of the inflation factor *lambda* can be important, but one needs to identify the SNPs that could artificially increase the *lambda* value. `gwasqc()` identifies the 'high quality' SNPs by means of filters on the imputation quality and on the MAF. Summary statistics are calculated on the 'high quality' SNPs only. The default thresholds are listed below:

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

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:

If we want to define 'high quality' SNPs those 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
```

Plotting Filter

The plotting filter is used to select appropriate data for the various summary plots. The filter has two threshold levels and each of them is applied dependently on the plot type and column. The default threshold values are listed below:

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

The default thresholds for the coded allele frequency and imputation quality can be redefined accordingly with the commands **MAF** and **IMP** for each input file.

Example:

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

A new plotting filter is set for the input file 'input_file_1.txt'. There is a first level of filters which selects SNPs with MAF > 0.02 and the imputation quality > 0.3, and a second, higher, level filter which selects SNPs with MAF > 0.03 and imputation quality > 0.5.

Output File Name

For both text and graphic output files, the output file names are created by adding a prefix to the input file names. The prefix is specified with the command **PREFIX**.

Example:

```
PREFIX res_
PROCESS input_file_1.txt
PROCESS input_file_2.csv
PREFIX result_
PROCESS input_file_3.tab
```

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

Verbosity Level

The command **VERBOSITY** allows to control the number of output figures, as described below:

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

Example:

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

Number And Content Of Plots

Number and content of the output plots depend on the setting of the plotting filter and on the available columns in the input file. If some dependency is not satisfied because of missing columns or some filter setting, then some plots could not be created or they could be truncated at different levels than expected. See the tutorial for the list of dependencies.

The boxplots comparing *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 in spite of the full file names, which could be too long and, therefore, clutter the plots.

Example:

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 boxplots will be based on the study sample sizes, which is reported with different names in the two input files. The labels "first" and "second" will be used to identify the two studies in the plots.

The Output Files

`gwasqc()` produces 4 types of files:

1. Figures, including QQ-plots, histograms, and boxplots.
2. One textual report file with `.txt` extension.
3. One comma-separated file with `.csv` extension, that contains all the summary statistics for the high quality imputation data.
4. One *HTML* document, that combines both textual output and figures and allows a very easy and dynamic querying of all the output in a hypertext browser.

Author(s)

Daniel Taliun, Christian Fuchsberger, Cristian Pattaro

Examples

```
# name of an input script
script <- "GWASQC_script.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

# run gwasqc() function
gwasqc(script)
```

kusk_check	<i>Comparison Of Skewness and Kurtosis of Effect between GWAS studies.</i>
------------	--

Description

Compare the distributions of effect estimates between GWA studies.

Usage

```
kusk_check(script, worst = c(50, 75, 90, 99, 100), plot = TRUE)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be processed along with basic information from each individual study, and instructions for the quality check.
worst	A vector consisting of any of integer numbers 50, 75, 95, 99, and 100.
plot	Logical value indicating if the diagnostic plot should be produced.

Details

The *GWAtoolbox* allows automatic comparison of skewness and kurtosis of effect size distribution between GWA studies. The `gwasqc()` function estimates the corresponding skewness and kurtosis values during the quality check and includes them into the *CSV* reports. Then, the auxiliary `kusk_check()` function can be used to export this information to the data frame and to produce the diagnostic plots. As input it requires the same script as `gwasqc()` function and assumes that all the *CSV* reports are in the current working directory. Additionally, an optional list consisting of any of integer numbers 50, 75, 95, 99, and 100 can be specified. The numbers correspond to the percentage of SNPs to be considered as representing the null distribution.

Author(s)

Cristian Pattaro, Daniel Taliun, Christian Fuchsberger

Examples

```
# name of an input script
script <- "GWASQC_script.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)
```

```
# run gwasqc() function
gwasqc(script)

# run kusk_check() function
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], lab = W$study[W$ku50 > 5], cex = 1, pos = 4)
```

pannotate

SNPs annotation with regions (e.g. genes).

Description

For provided SNPs finds all genes in specified deviation (e.g. 0, +/- 50000, +/- 100000, ...). The function is analogous to [annotate](#) and supports parallel processing of multiple GWAS data files. The parallelization is implemented with **snow** package using “SOCK” cluster type.

Usage

```
pannotate(script, processes)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be annotated along with basic information from each individual study.
processes	An integer greater than 1, which indicates the number of parallel processes. All processes are created on a <i>localhost</i> and communicate through sockets.

Details

Function `pannotate()` annotates every marker in input files with regions (e.g. genes) 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 in the input script. The regions with their chromosomal coordinates must be provided in a separate file. It is possible to annotate markers if only their names are available (e.g. rsId) in input 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.

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:

```
PROCESS input_file_1.txt
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’.

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:

```
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
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 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.

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:

```
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’.

Specifying Column Names in Input Data Files

In the table below, the complete list of the default column names for the GWAS data file 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

Given that different names can be provided for each GWAS data file, `pannotate()` 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:

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.

Specifying Column Names in Regions Files

In the table below, the complete list of the 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

Given that different names can be provided for each regions file, `pannotate()` 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:

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.

Specifying Column Names in Map Files

In the table below, the complete list of the 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

Given that different names can be provided for each map file, `pannotate()` 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:

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.

Field Separator in Input Data Files

The field (column) separator can be different for each GWAS data file. `pannotate()` automatically detects the original separator field for each input file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

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

For the input file 'input_file_1.txt' the field separator is determined automatically by the program but, for the input files 'input_file_2.csv' and 'input_file_3.txt' the separator is manually set to comma by the user.

Field Separator in Regions Files

The field (column) separator can be different for each regions file. `pannotate()` automatically detects the original separator field for each regions file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **REGIONS_FILE_SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

```
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
REGIONS_FILE genes_file_2.csv
REGIONS_FILE_SEPARATOR COMMA
PROCESS input_file_2.csv
REGIONS_FILE genes_file_3.txt
PROCESS input_file_3.txt
```

For the regions file 'genes_file_1.txt' the field separator is determined automatically by the program but, for the regions files 'genes_file_2.csv' and 'genes_file_3.txt' the separator is manually set to comma by the user.

Field Separator in Map Files

The field (column) separator can be different for each map file. `pannotate()` automatically detects the original separator field for each map file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **MAP_FILE_SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

```
MAP_FILE map_file_1.txt
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
MAP_FILE map_file_2.csv
MAP_FILE_SEPARATOR COMMA
REGIONS_FILE genes_file_2.csv
PROCESS input_file_2.csv
MAP_FILE map_file_3.txt
PROCESS input_file_3.txt
```

For the map file 'map_file_1.txt' the field separator is determined automatically by the program but, for the map files 'map_file_2.csv' and 'map_file_3.txt' the separator is manually set to comma by the user.

Case Sensitivity

By default the `pannotate()` assumes that column names in the all specified files are case insensitive. For example, the column names *CHR*, *Chr*, and *chr* are all perfectly equivalent. This behaviour can be modified for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names, as specified below:

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

Example:

```
CASESENSITIVE 1
```

```

MAP_FILE map_file_1.txt
REGIONS_FILE genes_file_1.txt
PROCESS input_file_1.txt
CASESENSITIVE 0
MAP_FILE map_file_2.csv
REGIONS_FILE genes_file_2.csv
PROCESS input_file_2.csv

```

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. `pannotate()` 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 output 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:

```

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.

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, `pannotate()` 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**. The supported arguments are listed below:

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

Example:

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

Output File Name

The output file names are created by adding a prefix to the input file names. The prefix is specified with the command **PREFIX**.

Example:

```
REGIONS_FILE genes_file_1.txt
PREFIX annotated_
PROCESS input_file_1.txt
PROCESS input_file_2.csv
PREFIX annot_
PROCESS input_file_3.tab
```

All the output files corresponding to the input files 'input_file_1.txt' and 'input_file_2.csv' will be prefixed with *annotated_*; the output files corresponding to the input file 'input_file_3.tab' will be prefixed with *annot_*.

Author(s)

Daniel Taliun, Christian Fuchsberger, Cristian Pattaro

Examples

```
# name of an input script
script <- "PANNOTATE_script.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

# run pannotate() function with 2 parallel processes
pannotate(script, 2)
```

 pgwasformat

Formatting of GWAS result files.

Description

Formats headers, orders columns, calculates inflation factors and applies genomic control in GWAS result files. The function is analogous to `gwasformat` and supports parallel processing of multiple GWAS data files. The parallelization is implemented with `snow` package using “SOCK” cluster type.

Usage

```
pgwasformat(script, logfile, processes)
```

Arguments

<code>script</code>	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be processed along with basic information from each individual study, and instructions for the quality check.
<code>logfile</code>	Name of a log file with processing output. The output contains calculated inflation factors, total number of markers and number of filtered markers.
<code>processes</code>	An integer greater than 1, which indicates the number of parallel processes. All processes are created on a <i>localhost</i> and communicate through sockets.

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:

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

The formatting is applied first to ‘input_file_1.txt’ and then to ‘input_file_2.csv’.

Field Separator

The field (column) separator can be different for each GWAS data file and during the formatting it is changed to *tabulation*. `pgwasformat()` automatically detects the original separator field for each input file *based on the first 10 rows*. However, the user has the possibility to specify the original separator manually for each individual file using the command **SEPARATOR**. The supported arguments and related separators are listed below:

Argument	Separator
----------	-----------

```

COMMA      comma
TAB        tabulation
WHITESPACE whitespace
SEMICOLON  semicolon

```

Example:

```

PROCESS input_file_1.txt
SEPARATOR COMMA
PROCESS input_file_2.csv
PROCESS input_file_3.txt

```

For the input file 'input_file_1.txt' the field separator is determined automatically by the program but, for the input files 'input_file_2.csv' and 'input_file_3.txt' the separator is manually set to comma by the user. After the formatting all three files will have tabulation as new field separator.

Renaming Columns

The original column names in the GWAS data files are renamed using the command **RENAME** in the input script. The command is followed by two words: the first one corresponds to the original column name, and the second one corresponds to the new column name. The column names can't contain tabulation or space characters.

Example:

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 *marker*, which should be renamed. The new column name should be *SNPID* for 'input_file_1.txt', and *rsId* for 'input_file_2.csv' and 'input_file_2.txt'. 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

```

Column Names

In the table below, the complete list of the default column names for the GWAS data file 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

IMP_QUALITY	was imputed (1) or genotyped (0) Imputation quality statistics; this can be different depending on the software used for imputation: MACH's <i>Rsq</i> , IMPUTE's <i>properinfo</i> , ...
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)
AVPOSTPROB	Average posterior probability for imputed marker allele dosage

Given that different names can be provided for each GWAS data file, `pgwasformat()` 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 present column name. To redefine the default column names for *coded* and *non-coded* alleles, the command **ALLELE** followed by two present column names is used. If the present column name was renamed to the new column name with the command *RENAME*, then the new column name must be used in the redefinition command.

Example 1:

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 P-value and standard error are *pval* and *SE*, respectively. In the 'input_file_2.csv', the column name for the P-value is the same as in 'input_file_1.txt', but the column name for the standard error is *STDERR*. The correct column redefinition is as follows:

```
PVALUE pval
STDERR SE
PROCESS input_file_1.txt
STDERR STDERR
PROCESS input_file_2.csv
```

There is no need to redefine the **PVALUE** field. Alternatively, if the column *pval* in 'input_file_1.txt' and 'input_file_2.csv' needs to be renamed to *p-value*, then the input script is as follows:

```
RENAME pval p-value
PVALUE p-value
PROCESS input_file_1.txt
STDERR STDERR
PROCESS input_file_2.csv
```

Example 2:

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
```

Columns Ordering

By default the `pgwasformat()` doesn't change the original ordering of columns in the input file. This behaviour can be modified for every input file in the input script using the command **ORDER** as specified below:

Argument	Description
OFF	The original ordering of columns is preserved
ON	Columns are re-ordered following the alphabetical ordering
ON column_1 column_2 ... column_n	Columns are re-ordered following the specified order <i>column_1 column_2 ... column_n</i>

Example:

Let's assume to have three input files: 'input_file_1.txt', 'input_file_2.csv' and 'input_file_3.txt'. Each file contains columns *marker*, *chromosome* and *bp* in the order as they are listed. The following input script renames the column *marker* to *SNPID* and switches the ordering mode for every input file:

```

RENAME marker SNPID
MARKER SNPID
CHR chromosome
POSITION bp
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 ordering of columns 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*.

Case Sensitivity

By default the `pgwasformat()` 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 modified for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names, as specified below:

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

Example:

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

Specifying Filters

The `pgwasformat()` filters SNPs based on minor allele frequency(MAF) and imputation quality. The default thresholds are listed below:

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

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 1:

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 2:

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

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

Inflation Factor and Genomic Control

By default the `gwasformat` doesn't calculate the inflation factor and doesn't apply the genomic control. This behaviour can be modified for every input file in the input script using the command **GC/GENOMICCONTROL** as specified below:

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

are corrected and saved to the new columns *PVALUE_gc* and *STDERR_gc*, accordingly.

If the inflation factor value is less than 1.0, then the genomic control is not applied.

Example:

```
GC ON
PROCESS input_file_1.txt
GC OFF
PROCESS input_file_2.csv
GC 1.1
PROCESS input_file_3.txt
```

Effective Sample Size

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

Output File Name

The output file names are created by adding a prefix to the input file names. The prefix is specified with the command **PREFIX**.

Example:

```
PREFIX res_
PROCESS input_file_1.txt
PROCESS input_file_2.csv
PREFIX result_
PROCESS input_file_3.tab
```

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

The Output Files

`gwasformat` produces one formatted (renamed/re-ordered columns, genomic control correction and etc.) copy of every original GWA data file. The formatting history information, containing calculated inflation factors and number of filtered markers, is saved to the log file under the provided *logfile* name.

Author(s)

Daniel Taliun, Christian Fuchsberger, Cristian Pattaro

Examples

```
## Not run:
```

```

# all input and output files are located in the subdirectory "doc" of the installed GWAtoolbox package
# change the workspace
currentWd <- getwd()
newWd <- paste(system.file(package="GWAtoolbox"), "doc", sep="/")
setwd(newWd)

# name of an input script
script <- "PGWASFORMAT_script.txt"

# name of a logfile
logfile <- "pgwasformat_log.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

\dontshow{options(device.ask.default = FALSE)}

# run pgwasformat() function with 2 parallel processes
pgwasformat(script, logfile, 2)

# restore previous workspace
setwd(currentWd)

## End(Not run)

```

pgwasqc

Quality Control Of GWA Data

Description

Performs the quality control of data from Genome-Wide Association Studies (GWAS). The function is analogous to [gwasqc](#) and supports parallel processing of multiple GWAS data files. The parallelization is implemented with **snow** package using “SOCK” cluster type.

Usage

```
pgwasqc(script, processes)
```

Arguments

script	Name of a textual input file with processing instructions. The file should contain the names and locations of all GWAS data files to be processed along with basic information from each individual study, and instructions for the quality check.
processes	An integer greater than 1, which indicates the number of parallel processes. All processes are created on a <i>localhost</i> and communicate through sockets.

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:

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

The QC is applied first to 'input_file_1.txt' and then to 'input_file_2.csv'.

Field Separator

The field (column) separator can be different for each GWAS data file. `pgwasqc()` 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**. The supported arguments and related separators are listed below:

Argument	Separator
COMMA	<i>comma</i>
TAB	<i>tabulation</i>
WHITESPACE	<i>whitespace</i>
SEMICOLON	<i>semicolon</i>

Example:

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

For the input file 'input_file_1.txt' the field separator is determined automatically by the program but, for the input files 'input_file_2.csv' and 'input_file_3.txt' the separator is manually set to tabulation by the user.

Missing Values

By default, `pgwasqc()` 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:

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

The *hyphen* symbol identifies missing values in the input file 'input_file_1.txt' and *NA* identifies missing values in the input file 'input_file_2.txt'.

Column Names

In the table below, the complete list of the default column names for the GWAS data file 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 <i>Rsq</i> , IMPUTE's <i>properinfo</i> , ...
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)
AVPOSTPROB	Average posterior probability for imputed marker allele dosage

Given that different names can be provided for each GWAS data file, `pgwasqc()` 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. To redefine the default column names for *coded* and *non-coded* alleles, the command **ALLELE** followed by two new column names is used.

Example 1:

Let's assume to have two input files, 'input_file_1.txt' and 'input_file_2.txt'. In the '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 redefinition is as follows:

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

There is no need to redefine the **EFFECT** field.

Example 2:

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
```

Case Sensitivity

By default the `pgwasqc()` 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 modified for every input file in the input script using the command **CASESENSITIVE**, that controls case sensitivity for the column names, as specified below:

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

Example:

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

Filter for Implausible Values

Often, there is the necessity to identify implausible values, to exclude unreliable results from the meta-analysis. Implausible values can happen due to data sparseness, errors in the data handling, or other causes.

`pgwasqc()` identifies SNPs with suspicious statistics (p-value, standard error, etc.) by applying appropriate threshold values. After the data processing, a detailed report including the number of SNPs with implausible statistics and the nature of the problem is produced. In addition, suspicious SNPs are excluded from the calculation of the summary statistics on data quality.

The default filter thresholds are listed below:

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)

The user has the option to modify the thresholds to account for specific needs. The new thresholds can be specified after the redefinition of the column name.

Example:

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 re-define 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) 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.

High Quality Filters

SNPs with low imputation quality and with too small minor allele frequency (MAF) could make spuriously small p-values happen. Checking for the presence of cryptic relatedness or hidden population sub-structure through the estimation of the inflation factor *lambda* can be important, but one needs to identify the SNPs that could artificially increase the *lambda* value. `pgwasqc()` identifies the 'high quality' SNPs by means of filters on the imputation quality and on the MAF. Summary statistics are calculated on the 'high quality' SNPs only. The default thresholds are listed below:

Default column name	Default thresholds
FREQLABEL	> 0.01
IMP_QUALITY	> 0.3

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:

If we want to define 'high quality' SNPs those 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
```

Plotting Filter

The plotting filter is used to select appropriate data for the various summary plots. The filter has two threshold levels and each of them is applied dependently on the plot type and column. The default threshold values are listed below:

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

The default thresholds for the coded allele frequency and imputation quality can be redefined accordingly with the commands **MAF** and **IMP** for each input file.

Example:

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

A new plotting filter is set for the input file 'input_file_1.txt'. There is a first level of filters which selects SNPs with $MAF > 0.02$ and the imputation quality > 0.3 , and a second, higher, level filter which selects SNPs with $MAF > 0.03$ and imputation quality > 0.5 .

Output File Name

For both text and graphic output files, the output file names are created by adding a prefix to the input file names. The prefix is specified with the command **PREFIX**.

Example:

```
PREFIX res_
PROCESS input_file_1.txt
PROCESS input_file_2.csv
PREFIX result_
PROCESS input_file_3.tab
```

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

Verbosity Level

The command **VERBOSITY** allows to control the number of output figures, as described below:

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

Example:

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

Number And Content Of Plots

Number and content of the output plots depend on the setting of the plotting filter and on the available columns in the input file. If some dependency is not satisfied because of missing columns

or some filter setting, then some plots could not be created or they could be truncated at different levels than expected. See the tutorial for the list of dependencies.

The boxplots comparing *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 in spite of the full file names, which could be too long and, therefore, clutter the plots.

Example:

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 boxplots will be based on the study sample sizes, which is reported with different names in the two input files. The labels "first" and "second" will be used to identify the two studies in the plots.

The Output Files

pgwasqc() produces 4 types of files:

1. Figures, including QQ-plots, histograms, and boxplots.
2. One textual report file with *.txt* extension.
3. One comma-separated file with *.csv* extension, that contains all the summary statistics for the high quality imputation data.
4. One *HTML* document, that combines both textual output and figures and allows a very easy and dynamic querying of all the output in a hypertext browser.

Author(s)

Daniel Taliun, Christian Fuchsberger, Cristian Pattaro

Examples

```
## Not run:
# all input and output files are located in the subdirectory "doc" of the installed GWAtoolbox package
# change the workspace
currentWd <- getwd()
newWd <- paste(system.file(package="GWAtoolbox"), "doc", sep="/")
setwd(newWd)

# name of an input script
```

```
script <- "PGWASQC_script.txt"

# load GWAtoolbox library
library(GWAtoolbox)

# show contents of the input script
file.show(script, title=script)

# run pgwasqc() function with 2 parallel processes
pgwasqc(script, 2)

# restore previous workspace
setwd(currentWd)

## End(Not run)
```

Index

*Topic **misc**

- annotate, 2
- dispersion_check, 9
- gwasformat, 11
- gwasqc, 17
- kusk_check, 24
- pannotate, 25
- pgwasformat, 34
- pgwasqc, 40

*Topic **package**

- annotate, 2
- dispersion_check, 9
- gwasformat, 11
- gwasqc, 17
- kusk_check, 24
- pannotate, 25
- pgwasformat, 34
- pgwasqc, 40

*Topic **plot**

- gwasformat, 11
- gwasqc, 17
- pgwasformat, 34
- pgwasqc, 40

*Topic **utilities**

- annotate, 2
- dispersion_check, 9
- gwasformat, 11
- gwasqc, 17
- kusk_check, 24
- pannotate, 25
- pgwasformat, 34
- pgwasqc, 40

annotate, 2, 25

dispersion_check, 9

gwasformat, 11, 34

gwasqc, 17, 40

kusk_check, 24

pannotate, 25

pgwasformat, 34

pgwasqc, 40