Introduction to “GWASinspector”

This tutorial introduces the “GWASinspector” package, its general form and how to apply the algorithm on GWAS result files.

Developed by:  Alireza Ani, Peter J. van der Most, Ahmad Vaez, Ilja M. Nolte

Version: 1.4

Date: January 2020
**Introduction**

When evaluating the results of a genome-wide association study (GWAS), it is important to perform a quality control to ensure that the results are valid, complete, correctly formatted, and, in case of meta-analysis, consistent with other studies in the same analysis. This package was developed to facilitate and streamline this process and provide the user with a comprehensive report. The process is divided into different phases, which are described in detail in this tutorial.
## Contents

- **Introduction** .......................................................................................................................... 2
- **The GWASinspector package** .................................................................................................. 4
  - How to install GWASinspector ................................................................................................. 4
  - Package dependency list ........................................................................................................ 4
  - Additional modules ................................................................................................................... 5
  - Auxiliary files .......................................................................................................................... 6
- **Available functions** ................................................................................................................ 8
- **Output file naming conventions** ............................................................................................. 12
- **Setting QC parameters in configuration file** .......................................................................... 15
  - Configuration file sections and properties .............................................................................. 16
  - Selecting input names using the filename key ........................................................................ 18
- **Step-by-step guide to run a QC** .............................................................................................. 19
  - Step 1: Load the package ........................................................................................................ 19
  - Step 2: Check R environment ................................................................................................... 19
  - Step 3: Download the standard allele frequency reference datasets ....................................... 20
  - Step 4: Get the header-translation table .................................................................................. 20
  - Step 5: Get the configuration file .............................................................................................. 20
  - Step 6: Modify the parameters in the configuration file ............................................................ 20
  - Step 7: Run the QC function ..................................................................................................... 21
- **Quality control steps of GWAS result files** ........................................................................... 22
  - Phase 1: Loading the dataset .................................................................................................... 23
  - Phase 2: Data integrity & SNP check ......................................................................................... 24
  - Phase 3: Aligning alleles with reference and comparing the allele frequencies ....................... 26
  - Phase 4: Quality control of the other statistics ......................................................................... 28
  - Phase 5: Generating the report ................................................................................................ 29
  - Phase 6: Saving the cleaned dataset ......................................................................................... 29
  - Phase 7: Comparing multiple GWAS results files ................................................................. 29
- **Interpreting the QC reports** .................................................................................................... 30
  - The overall report ..................................................................................................................... 30
  - The individual file report ......................................................................................................... 33
The GWASInspector package

How to install GWASInspector
The easiest way to install the GWASInspector package is to get it from the Comprehensive R Archive Network (CRAN).

install.packages('GWASinspector')

In this method, the minimally required packages (see below) will also be downloaded and installed automatically.

You can also install GWASInspector using the source or binary package, which is available from our website (GWASinspector.com). In this case, you will have to install any required packages beforehand yourself.

install.packages('/path/to/package', repos = NULL, type = 'source')

Package dependency list
The following packages and modules are used inside GWASInspector. Some system modules are optional. However, it is suggested to install them to gain full functionality. You can check the availability of the optional items after installing GWASInspector by running `GWASinspector::system.check()` function.

<table>
<thead>
<tr>
<th>Package name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ini</td>
<td>✓</td>
<td>Loads the configuration file</td>
</tr>
<tr>
<td>futile.logger</td>
<td>✓</td>
<td>Used for saving log messages</td>
</tr>
<tr>
<td>data.table</td>
<td>✓</td>
<td>Used for storing and manipulating input data</td>
</tr>
<tr>
<td>hash</td>
<td>✓</td>
<td>Used for translating header values</td>
</tr>
<tr>
<td>tools</td>
<td>✓</td>
<td>R base package</td>
</tr>
<tr>
<td>ggplot2</td>
<td>✓</td>
<td>Used for creating plots</td>
</tr>
<tr>
<td>knitr</td>
<td>✓</td>
<td>Used for creating plots</td>
</tr>
<tr>
<td>rmarkdown</td>
<td>✓</td>
<td>Used for writing reports in different formats</td>
</tr>
<tr>
<td>gridExtra</td>
<td>✓</td>
<td>Used for creating plot matrices</td>
</tr>
<tr>
<td>grid</td>
<td>✓</td>
<td>Used for creating plot matrices</td>
</tr>
<tr>
<td>RSQLite</td>
<td>✓</td>
<td>Used for storing reference files as a database</td>
</tr>
<tr>
<td>kableExtra</td>
<td>✓</td>
<td>Used for styling HTML report files</td>
</tr>
<tr>
<td>xlsx</td>
<td>✗</td>
<td>Used for creating excel report</td>
</tr>
<tr>
<td>pandoc</td>
<td>✗</td>
<td>See below</td>
</tr>
<tr>
<td>Java</td>
<td>✗</td>
<td>See below</td>
</tr>
</tbody>
</table>
Additional modules

- Pandoc is a free and open-source document converter that is used for creating an easy-to-view HTML report.
  Note: pandoc is automatically installed with RStudio.

**Installation:** Please follow the instructions on how to install pandoc on your system. You can also use the 'installr' package on Windows operating systems. First, install the 'installr' package and then run the following command.

```r
install.packages('installr')
installr::install.pandoc()
```

On most Linux distributions you can simply run the below command for installation. Please note that the user should have root privileges:

```bash
sudo apt-get install pandoc
```

- Java is required for 'xlsx' package to work properly.

- xlsx package is used for generating the report file in Excel format. You can install it with the following command. Java Environment is required and should be installed first.

```r
install.packages('xlsx')
```
Auxiliary files

The header translation table

This file is used to translate the dataset’s column names (the header) into the standard names used by GWASinspector. A sample file is provided as part of the package, which can be modified according to your requirements. The file contains a two-column table, with the left column containing the standard column names used by GWASinspector, and the right column the alternatives ones from the GWAS results files. Both the standard and alternative columns must be fully capitalized and separated by a tab character.

You can obtain a sample file that includes most common variable/header names with the following command:

```r
get.headerTranslation('/path/to/referenceFolder')
```

💡 Note: Duplicated entries and empty lines in the header translation table are NOT acceptable and will cause an error.

Example:

<table>
<thead>
<tr>
<th>Correct file</th>
<th>Wrong file because of a duplicated entry and an empty line</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFFECT_ALL A1</td>
<td>EFFECT_ALL A1</td>
</tr>
<tr>
<td>EFFECT_ALL ALLELE1</td>
<td>EFFECT_ALL ALLELE1</td>
</tr>
<tr>
<td>EFFECT_ALL COD_ALL</td>
<td>EFFECT_ALL COD_ALL</td>
</tr>
<tr>
<td>EFFECT_ALL CODED_ALL</td>
<td>EFFECT_ALL CODED_ALL</td>
</tr>
<tr>
<td>EFFECT_ALL CODED_ALLELE</td>
<td>EFFECT_ALL CODED_ALLELE</td>
</tr>
<tr>
<td>EFFECT_ALL EFFECT_ALLELE</td>
<td>EFFECT_ALL EFFECT_ALLELE</td>
</tr>
<tr>
<td>EFFECT_ALL EFF_ALL</td>
<td>EFFECT_ALL EFF_ALL</td>
</tr>
<tr>
<td>OTHER_ALL A2</td>
<td>EFFECT_ALL EFF_ALL</td>
</tr>
<tr>
<td>OTHER_ALL NONEFF_ALL</td>
<td>OTHER_ALL A2</td>
</tr>
<tr>
<td>OTHER_ALL ALLELE2</td>
<td>OTHER_ALL NONEFF_ALL</td>
</tr>
<tr>
<td>OTHER_ALL NON_COD_ALL</td>
<td>OTHER_ALL ALLELE1</td>
</tr>
<tr>
<td>OTHER_ALL NON_CODED_ALL</td>
<td>OTHER_ALL NON_COD_ALL</td>
</tr>
<tr>
<td>OTHER_ALL NON_EFFECT_ALLELE</td>
<td>OTHER_ALL NON_COD_ALL</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The standard allele reference file

The standard allele reference file is used to check the alleles in the datasets and to ensure that they are all in the same configuration (same strand, same coded alleles) in the post-QC data. This file can be in different formats, including .txt, .csv, .dat, .rds and .rdata files, and should contain the following columns:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hID</td>
<td>The harmonized ID of a variant. The structure of this item is [Chromosome:Position:VT], where VT (variant type) can be either 1 for SNPs or 2 for INDELs.</td>
</tr>
<tr>
<td>ID</td>
<td>rsID of the variant (optional)</td>
</tr>
<tr>
<td>REF</td>
<td>Reference allele</td>
</tr>
<tr>
<td>ALT</td>
<td>Alternate allele</td>
</tr>
<tr>
<td>AF#</td>
<td>Allele frequency of the alternate allele</td>
</tr>
</tbody>
</table>

# Some reference panels have multiple population level AF values. In this case, the population string should be included in column name (EUR for European population, AFR for African population, etc.).

The alternative allele reference file

The optional alternative allele reference serves as a back-up to the standard one. Variants that aren’t found in the standard reference will be compared to the alternative reference. If they aren’t in the alternative reference, they will be added to it. A check against an alternative reference is particularly useful before combining multiple GWAS results files in a meta-analysis. Mind that the order of the input files is important, because for the first occurrence of a variant that is not found in the standard reference, the data are considered the correct data for this variant (an assumption that might not be correct).

The alternative reference has seven required columns: hID, ID, REF, ALT, AF, SOURCE, and DATE_ADDED. In both reference files, the alleles must be aligned to the positive strand.

📝 Note: The alternative reference data set may be updated by the QC function (unlike the standard reference data).

The effect-size reference file

The optional effect-size reference file is used for comparing the effect-size of variants from the GWAS result files with another reference data effect-size, e.g. from earlier GWASs for the same phenotype. This is done to check whether the provided data are consistent with previous results for of the same phenotype or not. This file has four required columns: hID, EFF_ALL, NON_EFF_ALL, EFFECT

The correlation between the effect-size of variants in the two files is calculated and a scatter plot is also generated and saved in the output folder.
Available functions

Note: you can run `?function.name` in R to get an overview of any function in our package.

get.config()

Using this command, you can save a sample configuration file on your computer. Items in the file should be edited according to your needs and are described in the next section.

Arguments:

`dir.path` Directory to save the “config.ini” file.

Example:

```r
get.config(dir.path = '/path/to/folder')
```

get.headerTranslation()

Using this command, you can save the default column-header translation table file. Items in the file should be edited according to your needs. Refer to this section for more information about this file.

Arguments:

`dir.path` Directory to save the “alt_headers.txt” file.

Example:

```r
get.headerTranslation(dir.path = '/path/to/referenceFolder')
```

system.check()

Checks if required and optional packages are installed on the system. Although the optional packages do not contribute to the QC itself, having them available will allow for Excel and HTML formatted log files, which are easier to read and interpret.

Example:

```r
system.check()
```
**setup.inspector()**

This function imports a QC-configuration file into R, and then generates an object of the Inspector class. This object can then be passed to `run.inspector` function to perform the QC.

`setup.inspector()` requires a configuration (ini) file (see above). The function will also warn of any problems encountered when reading and validating this file.

**Arguments:**

- `config.file` Path to the configuration file on your computer.
- `validate` This option is only used for testing and debugging purposes and indicates whether to validate the configuration file. The default value is TRUE and it is strongly recommended to not turn it off for real jobs.

**Example:**

```r
job <- setup.inspector(config.file = '/home/user/config.ini')
print(job)
## or simply
job
```

**run.inspector()**

This is the main function of the package, which runs the QC algorithm on GWAS result files. It requires an object of class “Inspector”, which is created by `setup.inspector` function. Progress of the algorithm is printed to the R terminal and also saved in a log file. Printing of all messages inside the terminal can be suppressed by the `suppressMessages()` function without any effect on the log file.

**Arguments:**

- `inspector` An object of class “inspector”, which is created by the `setup.inspector` command.
- `test.run` If set to true, a test run is performed on the first 1000 lines of the input file. No plots will be generated.

**Example:**

```r
job <- setup.inspector("/home/user/config.ini") ## create the object
job ## check the parameters
job <- run.inspector(job) ## run the algorithm
```
result.inspector()
This function displays a brief report about the results of running the inspector algorithm on a list of GWAS result files. The full report including plots, cleaned files and summary statistics are saved in the output folder during the algorithm run.

Arguments:

- inspector: An object of class "inspector" which has already been inspected.

Example:
```
job <- setup.inspector("/home/user/config.ini")  ## create the object
job ## check the parameters
job <- run.inspector(job)  ## run the algorithm
result.inspector(job)  ## check the results
```

demo.inspector()

This function runs the algorithm on a fabricated GWAS result file. Users only need to set the path to an existing output folder, for saving the report files. The input file and reference dataset are embedded in the package.

Example:
```
demo.inspector('/home/user')
```

sqlite.db.check()

Summary of a SQLite database, reporting the number of tables, number of data rows for each table and the first row. This function will check the allele frequency reference database file specified in an object of class “Inspector” [allele_ref_std] parameter of the configuration file. This file should be in SQLite format with “sqlite” extension.

Arguments:

- inspector: An object of class "Inspector".

Example:
```
job <- setup.inspector("/home/user/config.ini")  ## create the object
def <- sqlite.db.check(job)
```
**man.plot()**

A function to generate Manhattan plots of a dataset.

Arguments:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>object</td>
<td></td>
<td>Data frame or data table containing the below columns</td>
</tr>
<tr>
<td>Chr</td>
<td>string</td>
<td></td>
<td>Name of chromosome column</td>
</tr>
<tr>
<td>Pvalue</td>
<td>string</td>
<td></td>
<td>Name of the p-value column</td>
</tr>
<tr>
<td>Position</td>
<td>string</td>
<td></td>
<td>Name of the position column</td>
</tr>
<tr>
<td>Beta</td>
<td>string</td>
<td>NULL</td>
<td>(optional) name of the effect-size column</td>
</tr>
<tr>
<td>std.error</td>
<td>string</td>
<td>NULL</td>
<td>(optional) name of the standard error column</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td></td>
<td>Full name and path of file to be saved (file extension should be 'png'). [e.g. “c:/users/researcher/study/man_plot.png”]</td>
</tr>
<tr>
<td>plot.title</td>
<td>string</td>
<td>'Manhattan Plot'</td>
<td>Title of the plot</td>
</tr>
<tr>
<td>plot.subtitle</td>
<td>string</td>
<td>''</td>
<td>Subtitle of the plot</td>
</tr>
<tr>
<td>sig.threshold.log</td>
<td>numeric</td>
<td>8</td>
<td>The -log10 transformed significance threshold, used for plotting a threshold line (e.g. 8 = 10^-8)</td>
</tr>
<tr>
<td>p.threshold</td>
<td>numeric</td>
<td>0.01</td>
<td>Threshold for plotting variants (i.e. p-values &gt; 0.01 will not be plotted). Setting a higher threshold will significantly increase plotting time</td>
</tr>
<tr>
<td>check.columns</td>
<td>logical</td>
<td>TRUE</td>
<td>Whether to check input columns for invalid values</td>
</tr>
</tbody>
</table>

Examples:

```r
Input.data <- fread("/path/to/resultFile")

man.plot(data = Input.data,
         chr='Chromosome',
         pvalue = 'P_value',
         position = 'Position',
         fileName='"/home/document/sampleFolder/m_plot.png")

man.plot(data = Input.data,
         chr='Chromosome',
         pvalue = 'P_value',
         position = 'Position',
         fileName='"/home/document/sampleFolder/m_plot.png",
         plot.title = 'Manhattan plot',
         plot.subtitle = 'sample research result',
         sig.threshold.log = 8,
         p.threshold = 0.01,
         beta= 'EFFECT',
         std.err= 'STD_ERR')
```
Output file naming conventions
In order to perform a comprehensive QC, GWASinspector, produce a series of tables and graphs. This section details which files are produced and what their purpose is.

Note: Keep in mind that only some of the following files may be generated and saved, depending on the configuration parameters and the integrity of input data.

Naming convention for a saved file is:
filename_output_tag +
name of input file without extension +
file_description +
extension

Notes:
- filename_output_tag: this is set in the configuration file (default value is QC)
- file_description: this describes the type of the file and its content. The following tables indicate possible values.

Examples:
- results1_cohort1_graph_EAF_SR.png
- results1_cohort2_SNPs_improbable_values.txt

Note: the main log and report files are always saved with the following names (overwriting any previous files with the same name).

- [filename_output_tag]_log.txt
- [filename_output_tag]_Report.txt
### Main output, report and log files

<table>
<thead>
<tr>
<th>File description</th>
<th>extension</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>txt</td>
<td>Summary of the QC process steps including any warnings</td>
</tr>
<tr>
<td>report</td>
<td>txt / HTML / xlsx</td>
<td>Report of the QC process</td>
</tr>
<tr>
<td>input_file_name</td>
<td>txt or txt.gz</td>
<td>Cleaned results file</td>
</tr>
<tr>
<td>object</td>
<td>rds</td>
<td>QC results as R data object</td>
</tr>
</tbody>
</table>

### Lists of incorrect variants encountered during QC

<table>
<thead>
<tr>
<th>File description</th>
<th>extension</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars_invalid_allele</td>
<td>txt</td>
<td>Variants with invalid alleles</td>
</tr>
<tr>
<td>vars_removed</td>
<td>txt</td>
<td>Variants with at least one missing or invalid value in a crucial column</td>
</tr>
<tr>
<td>vars_improbable_values</td>
<td>txt</td>
<td>Variants with at least one invalid value in a non-crucial column</td>
</tr>
<tr>
<td>vars_duplicates</td>
<td>txt</td>
<td>Variants with duplicated IDs</td>
</tr>
<tr>
<td>vars_monomorphic</td>
<td>txt</td>
<td>Monomorphic variants</td>
</tr>
<tr>
<td>vars_mismatches_BA</td>
<td>txt</td>
<td>Bi-allelic variants whose alleles did not match the reference</td>
</tr>
<tr>
<td>vars_mismatches_MA</td>
<td>txt</td>
<td>Multi-allelic variants whose alleles did not match the reference</td>
</tr>
<tr>
<td>vars_ambiguous</td>
<td>txt</td>
<td>Check “phase 3h” from <strong>this section</strong></td>
</tr>
</tbody>
</table>
## Graphs

<table>
<thead>
<tr>
<th>File description</th>
<th>extension</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph_M</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>graph_histogram</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>graph_EAF_SR</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>graph_EAF_AR</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>graph_p_correlation</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>graph_QQ</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>beta</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>Checkgraph_effect-size</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>Checkgraph_precision</td>
<td>(png</td>
<td>jpeg</td>
</tr>
<tr>
<td>Checkgraph_skew_kurt</td>
<td>(png</td>
<td>jpeg</td>
</tr>
</tbody>
</table>
Setting QC parameters in configuration file

GWAS Inspector uses a configuration (.ini) file to configure the QC algorithm. A sample file is included in the package, which can be used as a template (refer to get.config function tutorial). This file is edited by the user and then processed by the setup.inspector function. Please refer to the function’s tutorial for more detail.

This file has three components:

1. Key or property: these are the basic parameters for configuring the QC. Every key has a name and a value, delimited by an equals sign (=).
2. Sections: Keys are grouped into named sections. The section name appears on a line by itself, in square brackets ([ and ])
3. Comments: Semicolons (;) or hash (#) at the beginning of the line indicate a comment. Comment lines are ignored by setup.inspector.

⚠️ Notes:

- Key-names and section-names should **NOT** be edited or renamed. Otherwise the algorithm will not work properly.
- Lines that start with ‘#’ character are comments about a parameter. Do **NOT** un-comment the information line by deleting this character.
- Properties that start with ‘;’ character are commented parameters. In this case, the default value for that parameter is used. You can un-comment the parameter and set a value by removing the semicolon and changing the value after ‘=’ sign.
- Avoid duplicated keys.
## Configuration file sections and properties

Every part of the QC is configured in this file.

<table>
<thead>
<tr>
<th>Section</th>
<th>parameter</th>
<th>default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paths</td>
<td>filename</td>
<td></td>
<td>the name of the file(s) to be QC'ed (see below for explanation).</td>
</tr>
<tr>
<td></td>
<td>filename_output_tag</td>
<td>QC</td>
<td>output files will be prefixed with this tag.</td>
</tr>
<tr>
<td></td>
<td>dir_data</td>
<td></td>
<td>the folder where input files are located.</td>
</tr>
<tr>
<td></td>
<td>dir_output</td>
<td></td>
<td>folder where results are saved.</td>
</tr>
<tr>
<td></td>
<td>dir_references</td>
<td></td>
<td>folder where reference and header files are located.</td>
</tr>
<tr>
<td>supplementaryFiles</td>
<td>header_translations</td>
<td></td>
<td>path to header file</td>
</tr>
<tr>
<td></td>
<td>allele_ref_std</td>
<td></td>
<td>path to the standard allele reference file</td>
</tr>
<tr>
<td></td>
<td>allele_ref_std_population</td>
<td></td>
<td>Which population data of the standard reference should be used. If the file has only one population, COMMON should be used.</td>
</tr>
<tr>
<td></td>
<td>allele_ref_alt</td>
<td></td>
<td>path to alternative reference file</td>
</tr>
<tr>
<td></td>
<td>beta_ref_std</td>
<td></td>
<td>path to Beta (Effect-size) reference file</td>
</tr>
<tr>
<td>input_parameters</td>
<td>column_separator</td>
<td>[&quot;\t&quot;, &quot;,&quot;, &quot;,&quot;, SPACE]</td>
<td>Characters that used as column separators in the input files</td>
</tr>
<tr>
<td></td>
<td>na.string</td>
<td>[&quot;NA&quot;, &quot;nan&quot;, &quot;NaN&quot;, &quot;&quot;]</td>
<td>Characters used as missing values in the input files</td>
</tr>
<tr>
<td></td>
<td>imputed_T</td>
<td>[&quot;1&quot;, &quot;TRUE&quot;, &quot;t&quot;, &quot;YES&quot;, &quot;y&quot;]</td>
<td>Character used to indicate TRUE/FALSE in Imputation Status column</td>
</tr>
<tr>
<td></td>
<td>imputed_F</td>
<td>[&quot;0&quot;, &quot;FALSE&quot;, &quot;F&quot;, &quot;NO&quot;, &quot;N&quot;]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>effect_type</td>
<td>BETA</td>
<td>Beta or Odds Ratio? [BETA or OR]</td>
</tr>
<tr>
<td>output_parameters</td>
<td>save_final_dataset</td>
<td>TRUE</td>
<td>should the post-QC dataset be saved?</td>
</tr>
<tr>
<td></td>
<td>gzip_final_dataset</td>
<td>FALSE</td>
<td>should the post-QC dataset be compressed?</td>
</tr>
<tr>
<td></td>
<td>out_header*</td>
<td>standard</td>
<td>translation table for the column-names of the output file.</td>
</tr>
<tr>
<td></td>
<td>out_sep</td>
<td>\t</td>
<td>character-strings of the column-separator</td>
</tr>
<tr>
<td></td>
<td>out_na</td>
<td>NA</td>
<td>character-string to use for missing values</td>
</tr>
<tr>
<td></td>
<td>out_dec</td>
<td>.</td>
<td>character string to use for decimal points</td>
</tr>
<tr>
<td></td>
<td>html_report</td>
<td>TRUE</td>
<td>save final report in HTML format</td>
</tr>
<tr>
<td></td>
<td>object_file</td>
<td>TRUE</td>
<td>save an R object containing QC results from each input result file</td>
</tr>
<tr>
<td></td>
<td>add_column_multiallelic**</td>
<td>FALSE</td>
<td>Marking multi-allelic variants in the cleaned result file.</td>
</tr>
<tr>
<td></td>
<td>add_column_AFmismatch***</td>
<td>FALSE</td>
<td>Marking variants with a high allele frequency difference in the cleaned result file.</td>
</tr>
<tr>
<td></td>
<td>ordered</td>
<td>FALSE</td>
<td>Ordering the output cleaned files on CHR:POSITION value</td>
</tr>
</tbody>
</table>
### remove_chromosomes
- **remove_X**: FALSE  
  whether X-chromosome, Y-chromosome, pseudo-autosomal and mitochondrial SNPs are removed
- **remove_Y**: FALSE
- **remove_XY**: FALSE
- **remove_M**: FALSE

### plot_specs
- **make_plots**: TRUE  
  should the various QC steps create and save plots?
- **graphic_device**: png  
  file format for saving plot files (png, jpeg, tiff)
- **plot_title**:  
  this title is displayed at the top of plots (only for single-file QC)
- **plot_cutoff_p**: 0.01  
  Threshold for excluding low-significance SNPs from the QQ & Manhattan plots. Increasing this value will drastically slow QC.

### filters
- **HQfilter_FRQ**: 0.01  
  Allele frequency threshold value for the high-quality (HQ) variant selection
- **HQfilter_HWE**: 1e-6  
  HWE-pvalue threshold value for the high-quality (HQ) variant selection
- **HQfilter_cal**: 0.95  
  Call rate threshold value for the high-quality (HQ) variant selection
- **HQfilter_imp**: 0.3  
  Imputation quality threshold value for the high-quality (HQ) variant selection
- **threshold_diffEAF**: 0.15  
  If the difference between reported allele frequency and the frequency in the reference file(s) exceeds this amount, the variants will be reported.
- **minimal_impQ_value**: -0.5  
  The minimal and maximal possible (i.e. non-invalid) imputation quality values.
- **maximal_impQ_value**: 1.5

* Available standard formats are:
  - “standard” (default setting) retains the column names used by inspect function
  - “GWAMA”, “PLINK”, “GenABEL” and “META” set the column-names to those used by the respective program. The columns not used by those programs retain the standard names. Note that META’s allele_B corresponds with QC_GWAS effect-allele.

** As determined by comparison with the standard reference dataset. A new column named "MULTI_ALLELIC" is added to the cleaned result file, which contains either 0/1 or NA (if variant was not found in the standard reference).

*** The threshold_diffEAF parameter is used as the difference threshold. A new column named "highDiffEAF" is added to the cleaned result file which contains either 0/1 or NA (if variant was not found in the standard reference).
Selecting input names using the filename key

Input files should be in the 'dir_data' folder and are selected using the 'filename' parameter in the configuration file. This parameter serves as a template and can be used to load multiple files. This command is case-insensitive (e.g. File or FILE or file are identical).

Regular expression can be applied in this parameter to filter or select the input files. The following table provides some sample filters. You can refer to standard RegEx tutorials for more complex patterns.

Example:

<table>
<thead>
<tr>
<th>command</th>
<th>Description</th>
<th>Example of successful matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename = txt</td>
<td>All files in the dir_data folder that have ‘txt’ in their name will be selected.</td>
<td>gwas_file.txt gwas_file.txt.gz samples_txt_file.csv</td>
</tr>
<tr>
<td>filename = txt$</td>
<td>All filenames that end with ‘txt’ will be selected.</td>
<td>gwas_file.txt</td>
</tr>
<tr>
<td>filename = gz$</td>
<td>All filenames that end with ‘gz’ will be selected.</td>
<td>gwas_file.txt.gz</td>
</tr>
<tr>
<td>filename = ^gwas.+txt$</td>
<td>All filenames that start with ‘gwas’ and end with ‘txt’ will be selected.</td>
<td>gwas_file_h9v11.txt GWA5_sample_file.txt</td>
</tr>
<tr>
<td>filename = ^gwas.+txt.gz$</td>
<td>All filenames that start with ‘gwas’ and end with ‘txt.gz’ will be selected.</td>
<td>gwas样本1.txt.gz gwas_sample2_file.txt.gz</td>
</tr>
<tr>
<td>filename = ^(gwas</td>
<td>ewas).+txt$</td>
<td>All filenames that start with either ‘gwas’ or ‘ewas’ and end with ‘txt’ will be selected.</td>
</tr>
<tr>
<td>Filename = ^(gwas</td>
<td>ewas).+(txt</td>
<td>dat)$</td>
</tr>
</tbody>
</table>
| filename = ^(gwas|ewas).+(v10).+txt$ | All filenames that start with either ‘gwas’ or ‘ewas’, have ‘v10’ in the middle and end with ‘txt’ will be selected. | ewas_file_V10.txt gwas_v10_Cleanedfile.txt }
Step-by-step guide to run a QC

System requirements:

This package requires R (> 3.2) and is tested on popular operating systems (e.g. Microsoft Windows, Ubuntu, CentOS, macOS). System requirements depend on the size of the result files and references. A PC with Intel Core i5 CPU or equivalent (> 2.4GHz) and at least 36 GB of RAM is recommended for analyzing a set of routine GWAS result file. A 64-bit operating system is not required but strongly recommended.

Prepare:

1- Input folder: for storing the input result files
2- Output folder: for saving the QC output files and reports
3- Reference folder: for keeping the reference datasets and header-translation table
   o Allele frequency reference datasets can be obtained from our website.
   o A sample header-translation table is embedded in the package. Refer to `get.headerTranslation` function for more detail.
4- Configuration file: for setting the run-time parameters and above-mentioned folder paths. A sample file is embedded in the package that should be used as a template. Refer to `get.config` functions and section “Setting QC parameters in configuration file” for details.

Step 1: Load the package

After installation, try loading the package with the following command.

```
library(GWASinspector)
```

Step 2: Check R environment

Run the following function to check if required and optional libraries are available. Refer to the package dependency list (above) for detail about mandatory and optional libraries.

```
system.check()
```
Step 3: Download the standard allele frequency reference datasets

Standard allele frequency reference datasets are available from our website. This file should be unzipped and copied in the references folder [dir_references] property of the configuration file.

Step 4: Get the header-translation table

A sample header-translation table can be obtained with the get.headerTranslation() function. Refer to function manual and this section for more information about this file.

```
get.headerTranslation("/path/to/referenceFolder")
```

Step 5: Get the configuration file

A sample configuration file can be obtained with the get.config() function. Refer to function manual for information.

```
get.config("/path/to/folder")
```

Step 6: Modify the parameters in the configuration file

Please refer to the previous section for full detail on how to modify the parameters and default values. Parameters in this file are used for reading input files, analyzing the data and saving the reports. There are multiple lines of comment and information about each parameter (lines that start with # and ; are comments and sample possible parameters, respectively). You should only change the lines that contain a key according to your specific needs.
Step 7: Run the QC function

The QC is configured by the ini file, which is imported into R through `setup.inspector` and turned into an object of the Inspector class. To perform the QC, process the object with `run.inspector`. A quick scan of the results can be performed via `result.inspector`, but the primary outcome of the QC are the log files and graphs generated by `run.inspector`. An exhaustive log file indicating the progress and possible warnings is also saved which can be used for localization of any problems during this run.

```r
job <- setup.inspector("/home/user/config.ini")  ## create the object
job  ## check the parameters
job <- run.inspector(job)  ## run the algorithm
```

Note: You can also perform a test-run on the first 1000 lines of result files. Please refer to the section on `run.inspector` function for more detail.
Quality control steps of GWAS result files

The algorithm will process all selected files, one by one, through the following stages.

1. Loading
   a. Loading the dataset
   b. Translating the column headers to the standard names
2. Data integrity & SNP check
   a. Removing variants with missing/invalid crucial values
   b. Removing duplicated variants
   c. Removing monomorphic variants
   d. Removing allosomal or mitochondrial variants
   e. Checking missing/invalid values for non-critical variables (“improbable variants”)
   f. Generating a harmonized ID for each variant
3. Aligning alleles with reference and comparing the allele frequencies
   a. Negative strand switch
   b. Comparing input file with a reference dataset
   c. Flipping of alleles
   d. Switching of alleles
   e. Tagging palindromic variants
   f. Tagging High-Quality (HQ) variants
   g. Removing mismatched variants
4. Quality control of the other statistics
   a. Genomic control lambda, Visscher’s statistic, skewness, kurtosis calculation
   b. Checking P-value correlation between expected and observed values
   c. Calculating allele frequency correlation
   d. Creating histograms, scatterplots, QQ plots and Manhattan plot
5. Creating the quality control summary statistics report as HTML, excel and text files for easy interpretation of the results
6. Saving the cleaned dataset
7. Comparing multiple GWAS results files (if the algorithm is run on a series of files)
Phase 1: Loading the dataset

The QC process starts with reading the GWAS result input file(s).

Phase 1a: finding input files

Input files should be put in the 'dir_data' folder and they are selected using the 'filename' parameter in the configuration file. Refer to this section for details.

Phase 1b: importing the data

All files with tabular format can be selected and loaded. This includes 'txt', 'dat' and 'csv' files in either text or compressed ('zip' or 'gz') format. Loading zipped files takes more time than loading the unzipped version of the same file. In general, gzip files (e.g. *.txt.gz, *.csv.gz) are recommended over zip files format (*.zip), due to:

- faster loading
- by default, the name of the file inside the zipped package (.zip file) and the zipped file should match (for example, sample_file_no2.zip should contain sample_file_no2.txt file). This is not an issue for gzip files.

Input files should contain a header row and all columns should be separated using an identical column separator character. Tab character (\t) is selected by default. If your input file uses another character as column delimiter, this should be defined by setting the column_separator parameter in the configuration file.

Phase 1c: checking column headers

The algorithm requires the dataset to have all standard crucial columns for running the QC. Translating the data column names into below standard names is done using the header translation table (check here). The path to this table is defined in the header_translations argument of the configuration file.

Note: Input file will be excluded from further analysis if there are duplicated column names or any of the crucial columns are absent. Columns whose name cannot be translated (“unknown columns”) are ignored from further analysis, but will be preserved in the output file.
Table 1: columns that are analyzed during QC process, their valid values, and whether they are crucial

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Valid value</th>
<th>Crucial</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHR</td>
<td>Chromosome</td>
<td>1-26, x, y, xy, mt, X, Y, XY, MT</td>
<td>✓</td>
</tr>
<tr>
<td>POSITION</td>
<td>Variant position on chromosome</td>
<td>Integer numbers</td>
<td>✓</td>
</tr>
<tr>
<td>EFFECT_ALL</td>
<td>Effect or Reference allele</td>
<td>G,C,T,A,I,D,-,0,R</td>
<td>✓</td>
</tr>
<tr>
<td>OTHER_ALL</td>
<td>Other or Alternate allele</td>
<td>G,C,T,A,I,D,-,0,R</td>
<td>✓</td>
</tr>
<tr>
<td>BETA</td>
<td>OR</td>
<td>Effect size (either beta or odds ratio)</td>
<td>Numeric value</td>
</tr>
<tr>
<td>STDERR</td>
<td>Standard error</td>
<td>Numeric value</td>
<td>✓</td>
</tr>
<tr>
<td>MARKER</td>
<td>Variant marker (e.g. rsID, hID)</td>
<td>string</td>
<td>✗</td>
</tr>
<tr>
<td>STRAND</td>
<td>Which strand</td>
<td>+ or -</td>
<td>✗</td>
</tr>
<tr>
<td>PVALUE</td>
<td>P-value</td>
<td>Numeric value</td>
<td>✗</td>
</tr>
<tr>
<td>EFF_ALL_FREQ</td>
<td>Allele frequency</td>
<td>Numeric value</td>
<td>✗</td>
</tr>
<tr>
<td>HWE_PVAL</td>
<td>HWE P-value</td>
<td>Numeric value</td>
<td>✗</td>
</tr>
<tr>
<td>IMP_QUALITY</td>
<td>Imputation quality</td>
<td>Numeric value</td>
<td>✗</td>
</tr>
<tr>
<td>IMPUTED</td>
<td>Variant is imputed or not</td>
<td>List of numbers (default: 0-1) Can be set in the configuration file</td>
<td>✗</td>
</tr>
<tr>
<td>CALLRATE</td>
<td>Call rate</td>
<td>Numeric value</td>
<td>✗</td>
</tr>
<tr>
<td>N_TOTAL</td>
<td>Number of samples</td>
<td>Numeric value</td>
<td>✗</td>
</tr>
</tbody>
</table>

Phase 2: Data integrity & SNP check

The aim of this step is to check that all variants have the required data and that all columns only contain valid values. Count of all removed lines from the following steps are save in the log and report files.

This phase is divided into below sections:

Phase 2a: Removing variants with missing/invalid crucial values

Variants with an invalid or missing value in any of the crucial columns are removed from the dataset and the first 100 variants are saved as a text file in output folder (check the “Report files” section of this tutorial for further information).

Phase 2b: Removing duplicated variants

Variants with duplicated IDs and alleles are removed from the dataset. By default, the combination of chromosome, position, effect-allele and other-allele of the variants are checked for duplicated information. The first 100 duplicated variants are saved as a text file in output folder.
**Phase 2c: Removing monomorphic variants**

Variants with allele frequency of 0 or 1 and variants with identical alleles are monomorphic and will be removed from dataset.

**Phase 2d: removing allosomal and mitochondrial variants**

This step is optional and will delete variants on a specific chromosome (X or 23, Y or 24, XY or 25, M/MT or 26). The default setting is to keep all variants; to change this, set the following parameters in the configuration file to TRUE:

- remove_X
- remove_Y
- remove_XY
- remove_MT

**Phase 2e: Checking missing/invalid values for non-crucial variables**

Variants with missing or invalid non-crucial values will be counted and reported.

**Phase 2f: Adding harmonized ID for each variant**

A harmonized ID will be generated for each variant from other variables. The structure of the hID is:

\[
\text{hID} = \text{Chromosome:Position:VT}
\]

Note: Variant Type (VT) is either 1 for SNPs or 2 for non-SNP variants. This is determined by evaluating both alleles of a variant.
Phase 3: Aligning alleles with reference and comparing the allele frequencies
The aim of this step is to make sure that the reported allele frequency is for the correct allele (effect allele) and of the expected order. This is accomplished by matching the data with a reference panel (e.g. HapMap, 1000G) and then adjusting the alleles.

Phase 3a: Negative strand switch
The STRAND column is checked at first (if present) and negative-stranded variants will be “strand-switched”. This means that an “A” allele becomes “T”, “T” becomes “A”, etc.

Phase 3b: comparing input file with reference dataset
Each variant is looked up in the reference dataset by the harmonized ID.

Phase 3c: Flipping of the alleles
Variants whose effect allele matches the reference major allele and non-effect allele matches the reference minor allele will be “flipped”: their alleles are reversed, and their effect-size and allele frequency inverted (i.e. effect-size = 1.32, AF = 0.76 becomes effect-size = -1.32, AF = 1 - 0.76 = 0.24)

Phase 3d: Switching of the alleles
If a variant is found in the reference data but alleles do not match, function will attempt to fix this by carrying out a strand-switch. This assumes that variants are on the negative-strand, but have not been listed as such. These variants are counted and reported as switched variants in the output report.

☞ Note: the process of flipping and switching of the alleles is more complicated for multi-allelic variants. Input variant alleles will be checked against the combination of alleles for the matched multi-allelic variant and then modified accordingly.

Phase 3e: Tagging palindromic variants
Palindromic variants are variants with a strand-independent allele configuration (i.e. A/T or C/G), and will be reported in the output report file.

Phase 3f: Tagging High-Quality (HQ) variants
High-quality (HQ) variants are selected through the settings specified by thresholds on allele frequency, call rate, HWE p-value and imputation quality values that are set in the configuration file.
Notes:

- The HQ filter does not remove SNPs from the dataset; it merely excludes them from a number of tests (including the Manhattan plot).
- The allele frequency filter is two-sided (check the below example).
- A variant is considered high-quality if it meets all quality-criteria.
- The thresholds are inclusive; i.e. variants that have a value equal to or higher than the threshold will be included.

Example: by setting the below values in the configuration file, variant whose allele frequency is (above 0.01 OR below 0.99) AND HWE P-value is above 10^-6 AND call-rate is above 0.95 AND Imputation quality is above 0.3 will be considered as HQ variant.

- HQfilter_FRQ = 0.01
- HQfilter_HWE = 1e-6
- HQfilter_cal = 0.95
- HQfilter_imp = 0.3

Phase 3g: removing mismatched variants

Variants that are found in the reference dataset by hID matching but their alleles could not be matched after flipping or switching of the alleles are removed from dataset. The first 100 variants are saved as txt file in the output folder.

Note: Mismatched multi-allelic variants will be saved as a separate file for user observation.

Phase 3h: removing ambiguous variants

Variants that cannot be correctly matched due to the following conditions are removed from the result file (this file almost always includes non-SNP variants). The first 100 variants are saved as txt file in the output folder:

- INS and DEL on the same position with reverse alleles. This mimics a switched variant.
- Absence of alleles for a multi-allelic non-SNP variants (e.g. I,R,0,D).
Phase 4: Quality control of the other statistics

At this point no further changes will be made inside the file. The function first determines if there are sufficient non-missing, non-invalid values for the various QC tests.

**Phase 4a: Visscher’s statistic, skewness and kurtosis calculation**

- Kurtosis is a measure of how well a distribution matches a Gaussian distribution. A Gaussian distribution has a kurtosis of 0. Negative kurtosis indicates a flatter distribution curve, while positive kurtosis indicates a sharper, thinner curve.
- Skewness is a measure of distribution asymmetry. A symmetrical distribution has a skewness of 0. A positive skewness indicates a long tail towards higher values, while a negative skewness indicates a long tail towards lower values. Ideally, one expects both the skewness and kurtosis of effect sizes to be close to 0. But in practice, these statistics can be hugely variable.
- Visscher’s statistic is calculated by the following formula.

\[
\text{Median} \left( \frac{1}{2 \times FRQ \times (1 - FRQ) \times (SE)^2} \right) / N
\]

\[FRQ = \text{allele frequency}, \ SE = \text{standard error}, \ N = \text{sample size}\]

All above values are separately calculated and reported for HQ/LQ variants.

**Phase 4b: Checking P-value correlation between expected and observed values**

Expected p-value for all variants are calculated from \(\chi^2 = \left( \frac{\text{Effect}}{\text{Stderr}} \right)^2\), with 1 degree of freedom. This value is compared with the observed data in the column and the correlation between the two should be close to 1. If this isn’t the case, either a column was misidentified when loading the data or the wrong values were used when generating the dataset.

**Phase 4c: Calculating allele frequency correlation**

The correlation between allele frequency of variants in the input file and the reference dataset are calculated for palindromic and non-palindromic variants. The correlation efficient should be above 0.9.

**Phase 4d: Creating histograms, scatterplots, QQ plots and Manhattan plot**
**Phase 5: Generating the report**

A separate report file will be generated for each input file. Also, depending on your computer’s available modules and installed packages, HTML and Excel format of the report will be generated and saved in the output folder for easy interpretation of the results.

**Phase 6: Saving the cleaned dataset**

The cleaned dataset will be saved in the output folder according to parameters in the configuration file. Check the detail on ‘output_parameters’ section of the configuration file.

**Phase 7: Comparing multiple GWAS results files**

Important summary statistic and key metrics of input study files are compared together and saved in tabular format plus three graphs. This report help find significant anomalies or differences between the files.

1. **Precision plot**: As sample size increases, the standard error is expected to decrease. Hence, the precision (\(1 / \text{median standard-error}\)) is plotted against the square root of the sample-size for many studies, one a linear relation is expected.
2. **Skewness-kurtosis plot**: Generates a skewness vs. kurtosis plot for input result files.
3. **Effect-sizes box plot**: compares the distribution of effect-sizes between studies, sorted on sample size.
Interpreting the QC reports

The overall report
The first thing to look at when running GWASInspector is the output in the R console. This output will also be copied into a log file (results1_log.txt, in our example). There's a great deal of text in there, but most of it are status flags indicating the process of the QC. The only thing that you need to check is whether all the intended GWAS files were processed, and whether it flagged any WARNINGS or ERRORS.

After skimming through the log file, you should check the QC report. This is the main summary of the QC. Depending on your settings and system, the report will be saved in multiple formats (.txt, Excel and HTML). In our example, this would be results1_report.txt, results1_report.xlsx, etc. The report consists of two tables and, if multiple files were processed, three figures (incorporated in the HTML file, but also saved with the “Checkgraph_” prefix). The topmost table lists the processed files, and serves as a legendum for the below table and figures. In the HTML report, the table is also hyperlinked to the reports of individual files; in the Excel file, the individual file reports are on separate tabs. Regardless of the format chosen, GWASInspector will always produce individual .txt reports for each input file - the information in there is identical to that in the HTML and Excel reports.

The multi-file report is the second table of the QC report (and the third tab of the Excel file) and gives a side-by-side comparison of the QC results for all files. This information is a condensed version of the data given in the individual file reports (so for see below for how to interpret these) - the purpose of the table is to make it easier to spot substantial differences between the individual results files.

This is also the purpose of the three graphs. The topmost plot of the HTML file (checkgraph_precision) is a precision plot. Precision is the inverse of the median standard error (i.e. the uncertainty of the estimates): it’s plotted against the square root of the sample size. As sample size increases, you would expect precision to increase proportionally. As such, you expect all your results to be on a roughly diagonal line.

🔗 Note: This plot will not be generated if sample size is missing in the result file.
Figure 1: Sample precision plot

In the below example (Figure2), however, there is one file that has a clearly higher precision than the others. It indicates that the file is not comparable to others, which means they cannot be combined in a regular meta-analysis. There are several possible causes for differences in precision: it could be that they analyses a different phenotype (or the same phenotype in a different unit or transformation), or a different model. In this case, the culprit was a different software program used in the analysis, which performed an undocumented standardization of the outcome values.

Figure 2: Precision plot that indicates a discrepancy in the file series
The second, Checkgraph\_skew\_kurt, is a **skewness vs. kurtosis plot**. Skewness is a measure of distribution asymmetry; while kurtosis is a measure of how well a distribution matches a Gaussian distribution. Like with the precision plot, you would expect results of comparable analyses to have comparable skewness and kurtosis values. If there are obvious outliers, there is something wrong with the effect-size distribution of that file, which will usually cause further issues that can be spotted in the QC report of that particular file.

![Skewness vs Kurtosis plot](image)

**Figure 3: Sample skewness vs kurtosis plot**

The final plot, Checkgraph\_effect\_size, is a box-plot of effect-sizes. It compares the **effect-size range** (of high-quality markers only) of each input file, and sorts them by sample size. This serves the same purpose as the precision plot: with increased sample size you expect a reduced effect-size range because precision improves. If a file has notably wider or narrower range than expected, this strongly suggests that its results are not comparable to the others.

![Box-plots](image)

Again, we see that file number 4 has a substantially narrower range than the other files. It’s only the 3\textsuperscript{rd} largest sample; yet it’s by far the most precise. Again, this means we cannot do a regular meta-analysis over these 6 files.
The individual file report

In the Excel file, these will be on separate tabs of the overall report.

The individual file report, and the accompanying plots, are the main meat of the QC. The top of the report consists of a recap of the analysis settings, which is just there as a reminder. In the HTML file, this is followed by a table giving the translation of the column headers. Check if it reports a missing column (if it reports a missing column that should actually be present in the dataset, it was probably misspelled in the header translation table), and move on to the next table.

The next table, the column report, is a test of data integrity. It checks if the columns contain the values you would expect (standard errors above zero, p-values between 0 and 1, etc.). It’s pretty common to see a small number of anomalies (standard errors rounded down to 0, and such) per column; and sometimes large numbers (up to 2/3 of the dataset) of effect-sizes and standard errors may be missing (i.e. the software did not produce results for this particular marker, so it simply gave a NA as result). Such outcomes are not problematic. What is problematic is if there are large numbers of invalid values; or entire columns are missing when they should be present. This indicates something went wrong either when formatting the dataset, or when GWASinspector translated the column names (see the table above).

Below that is the variant processing table. This table gives an overview of the number of variants in the dataset; and whether they are excluded during the QC process. Step 1 is the removal of duplicated variants and variants lacking an effect size (or odds ratio), standard error and/or p-value (or, alternatively, have an invalid value, such as a standard error of -1). As these values are the primary result of a GWAS, markers without these are essentially useless and can be safely removed.

The second step involves removing monomorphic markers (i.e. markers with an allele frequency of 0 or the same minor and major allele), as these results are uninterpretable. Allosomal and mitochondrial markers are also removed here if the configuration file specified this.

The third step is performed when the results file is matched to an allele reference. In order to interpret the effect-sizes, it is necessary to know the effect (and alternative) allele. Therefore GWASinspector will compare the results to an allele reference and check if the alleles match. This step will also remove variants with more than two alleles, if these could not be verified in the reference.

Following is a series of tables in the Description of variants and the Result from matching variants with reference datasets section - an overview of the remaining variants in the dataset. These are of little importance, though do check whether the majority of markers was matched with the allele reference. If a substantial number of variants did not appear in the reference - it suggests the wrong reference was used, or that your data uses a different marker naming convention.

The main outcome of note here are the allele frequency correlation $r$ values. These indicate how well the reported allele frequencies match that of the reference. Ideally, you want this to be over 0.9. However, in small analysis samples, or when your population doesn’t match the reference, the correlation can be much lower. That’s not necessarily a problem, but negative values are. It indicates that the allele frequency is reported for the wrong allele. This in turn may mean that the effect and reference alleles have been switched; and that your effect-size is reported for the wrong allele as well. Scatterplots of
reported and reference frequencies are also included to give a better view in how the frequencies are distributed (in our example, these would be named QC_results_graph_EAF_SR.png).

The scatter plot is split into 3 sections: SNPs, palindromic SNPs, and non-SNPs. The SNP correlations are fine, but a minority of non-SNPs appears to have the opposite allele frequency to the reference. This is troubling - if their frequency is the exact opposite of what is expected, either the frequency is reported or the effect and non-effect alleles have been flipped (in which case the reported effect-size is suspect as well). Also note that, while this problem did affect the allele frequency correlation, it is still fairly high. Merely looking at the tables wouldn’t have revealed this - you need to check the graphs as well.

Of interest are also the palindromic SNPs: these are SNPs with the same allele on the positive and negative strand (i.e. A/T or C/G SNPs). Because of this, it is not possible to recognize whether palindromic SNPs are on the negative or positive strand. As such, if the correlation is much lower for these SNPs than for non-palindromic ones, there may be unrecognized strand-switching going on. Fortunately, that does not appear the case here.

The final section is the most important one: the **Summary statistics**. The *p-value correlation* is a test of how well the reported p-value actually matches the effect-sizes and standard errors. If this is less than 1, or if the graph does not show an exact 1-to-1 scale; the p-values you have received are modified somehow. In case of the scale not being 1 to 1; this may be due to a genomic control correction. If not, the p-value doesn’t belong to the effect size. Again, don’t just look at the tables but check the plot as well:
Figure 4: Sample P-value correlation scatter plot

Figure 5: Sample of a possible problematic P-value correlation scatter plot
In the above example (Figure 5), the problem is related to the analysis software. Certain analysis models in SNPtest, for example, will show this problem for a subset of low-quality markers. We cannot explain this, but since the markers in question are always low quality it’s a minor problem anyway.

The next table covers the data distribution itself. The lambda value is also very important. It indicates p-value inflation; that is: how much more significant your p-value distribution is compared to a random p-value distribution. Ideally, this value should be 1. Values up to 1.1 are acceptable, but anything higher than that suggest some sort of population stratification. See also the QQ plots, below.

The final cells of this indicate whether HWE p-values, imputation quality, callrate and sample size are fixed. This won’t affect the QC, but it may indicate laziness during formatting of the file. For example sample size should be affected by callrate/imputation quality/genotyping uncertainty. If it is fixed, it suggests that the analyst simply assigned the max sample size to all variants without taking this into account.

The second distribution statistics table gives an overview of the distribution of variables: see also the figure 6 below. Finally, at the bottom there are the QQ plots and Manhattan plots. The Manhattan plot (figure 7) gives you a quick overview of whether there are significant findings present.

![Histogram plots displaying variable distribution.](image)

Figure 6: Histogram plots displaying variable distribution.
The QQ plots (figure 8) are a visual alternative to the lambda value. In a completely random distribution (lambda = 1), the lines follow the 1:1 diagonal. If you found a few significant loci, but the rest of the distribution is random, the line follows the 1:1 diagonal, except near the end where it will rise up. That is the result you are hoping for. If the line is consistently above the 1:1 diagonal, the p-values are too significant overall and there appears to be a form of population stratification in your dataset. One way to correct this would be to perform a genomic-control correction with the lambda value - but you’d first want to check where this stratification is coming from and see if you can correct it that way. Frequently, such oversignificance can be filtered out by removing variants with low AF values or imputation quality.

Figure 7: Sample Manhattan plot
Finally, GWASinspector may (depending on the input file) produce a series of .txt files (check here). These are lists of duplicated markers, markers whose alleles did not match with the reference, etc. They are intended to show the user what kind of data was excluded. Looking at them may give a hint as to what went wrong. Note that the files are not complete; usually only the first 100 entries are saved in order to save disk space and time.