User's Manual of QCF90 (Draft)

Yutaka Masuda

March 6, 2018

Contents

1 Introduction

1.1 What is this?

QCF90 is a software tool for quality control on genomic and pedigree data used especially in animal breeding. The "quality-control" detects and optionally removes errors or inaccurate information in the data. The data include single nucleotide polymorphisms (SNP) as genetic markers and a pedigree list with an animal and its sire and dam. The program reports markers and genotyped animals with problems e.g. low call-rate, Mendelian inconsistency among relatives, and so on. Such unqualified markers and individuals can be removed from the original files.

QCF90 can efficiently handle a large number of animals genotyped with 50K SNP chips. A genotype is represented as a 2-bit code and multiple genotypes are packed into an array in memory. Some quality controls are performed directly for the packed genotypes using bitwise operations. This strategy saves both a computing time and memory requirements.

QCF90 is a command line tool like Linux/Unix tools and some scientific software (e.g. `plink`). All required information should be supplied as options by the user. Although this program has been developed from scratch, generated reports are very similar to PREGSF90, a BLUPF90 family program. The differences between two programs will be explained in a later section.

It is written in Fortran 2003 and compiled with Intel Fortran Compiler. The compiled binary is available on the website of Animal Breeding and Genetics Group at University of Georgia (`http://nce.ads.uga.edu`) for Linux, macOS, and Windows.

1.2 Features

QCF90 checks the physical format for genomic data. Once this check is passed, QCF90 calculates the following several statistics to determine unqualified markers and genotyped animals. A user can compute specific statistics only.

- Call rate for each marker
- Call rate for each animal
- Statistic for Hardy-Weinberg equilibrium (χ^2 value)
- Allele frequency

The program can also check the duplicated genotypes (identical genomic data) in the data. Only a few markers can be used for the check. The following two modes are available.

- Duplication checks within a genomic file
- Duplication checks between two files

QCF90 optionally accepts a pedigree file. The program checks the physical format of the file and finds logical errors (e.g. loops) in pedigree. If both genomic and pedigree files are supplied, the program performs the following checks.

- Mendelian conflicts between a parent and its progeny

The supplied pedigree file is not necessarily pre-processed by RENUMF90. The pedigree file is allowed to have alphabets and symbols as well as integers.

After the checks, the program makes a report (a status file) on quality of markers and animals. The program creates "clean" files by removing all unqualified markers and animals from the original files. This process can be done in the same run on quality control. Or, in the separate run, the program can read the status file and create clean files without computations for quality control. In this way, the user can check the details and remove particular markers and animals with specific failure.

1.3 Difference between QCF90 and PREGSF90

All BLUPF90 family programs can check the quality of genomic data, remove unqualified genomic markers, and calculate relationship matrices. PREGSF90 is an independent tool only to performs genomic-related checks and computations. This program has the same interface and design to BLUPF90 so it is useful especially when all files have already prepared for the BLUPF90 programs. All the required files are supposed to be processed with RENUMF90.

QCF90 has the same functionality and gives (almost) the same results to PREGSF90. Required files do not have to be prepared with RENUMF90. QCF90 is newly designed and specialized to quality check on a large genomic data.

See the following descriptions for details.

- Interface. PREGSF90 needs a parameter file to describe the instruction for quality-control. QCF90 directly reads options from the command line.
- Files. PREGSF90 always needs at least 4 files i.e. phenotypic, pedigree, genomic, and cross-reference (XrefID) files no matter what you don't really need such files. QCF90 needs a genomic file and/or a pedigree file only.
- Format. PREGSF90 requires files prepared by RENUMF90 (a "renumbering" process). QCF90 doesn't need the "renumbering" process. It can also accept the "renumbered" files.
- Checks. PREGSF90 shows minimal messages on erroneous genomic data and none on abnormal pedigree. QCF90 shows precise messages both on genomic and pedigree errors.
- Efficiency. PREGSF90 needs a large amount of memory if the data is big. QCF90 needs less memory and runs faster in many cases.
- Purposes. PREGSF90 performs more than quality control. QCF90 focuses on quality control.

1.4 Condition of use

2 Basic usage

2.1 Overview of the program

2.1.1 Option specification

QCF90 is a command line tool like Linux/Unix commands. All required information is supplied with options in the command line on your shell (or a command window). Here we introduce a general usage of the program. Options related to actual quality control will be described in the next section.

The program can be invoked by typing the command name qcf90.

```
qcf90
```

By default, the program just shows a list of options and stops immediately. The program performs the quality control by putting the option characters followed by qcf90. For example, following command run the quality control for genomic data stored in a file snpdata.txt (here we don't explain what this command will do; just see the structure of the command line).

```
qcf90 --snpfile snpdata.txt
```

The sequence of characters --snpfile is called "option". The option always starts with two letters --. For this particular option, an argument (snpdata.txt) is supplied as an additional information. Some options don't need an argument and some needs more 2 or more arguments. The user should put one or more white spaces among the command name (qcf90), the options, and the arguments.

Two or more options are acceptable. The following example specifies 2 options in the same command line.

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt
```

All options and arguments should be separated by white spaces. This program assumes the marker file and the pedigree files share the same ID for animals.

The program accepts the files generated with renumf90. You should supply the cross-reference (XrefID) file when you combines marker and pedigree files. See the later sections for details.

2.1.2 Help

The option --help shows selected options on screen and the program stops.

```
qcf90 --help
```

With this option, all the other options will be ignored. The Complete option-list is available with --long-help.

```
qcf90 --long-help
```

2.1.3 Script file

QCF90 can read options from a file as well as the command line. The file is called a script as used in the plink software. In a script file, each line has to have one option with its arguments. For example, assume a script file make_clean_files.txt with the following lines.

```
--snpfile snpdata.txt
--pedfile pedigree.txt
--save-clean
```

The user can run the program with the script file.

```
qcf90 --script make_clean_files.txt
```

This is equivalent to the following command.

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt --save-clean
```

QCF90 reads the options from a script file the first, then from the command line the next. If the same option with different arguments is specified in both sources, the command line option will be primarily used. In other words, the command line options overwrite the options in the script file.

2.1.4 Dry-run mode

QCF90 has many options to control the computations. It is useful for the user to know what is actually happening with specified options without any actual computations nor modifications. QCF90 has an option --dry-run just to do this. The user can specify regular options in addition to the dry-run option.

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt --dry-run
```

QCF90 kindly shows what will be calculated and which file will be created. All messages will be shown on screen and no files will be created. This option is recommended to all users before running serious quality controls.

2.2 Quality control on genomic and pedigree data

2.2.1 Genomic data

Here we use an example marker file snpdata.txt. The first several lines in this file are as follows.

```
A57309  12221102222000211111221211011012011011
A57310  12221102222000211111221211021012011011
A57311  11220011021110220011012212100010020202
A57312  12221102222000211111221211011012011011
A57313  01221111222111110011112202100002211110 1
```

The file should have the same format for the BLUPF90 family programs (See the technical details chapter for details). Briefly, the data consists of at least 2 fields per row: 1) ID for genotyped animal and 2) the animal's genotypes. The 2nd field must start with the same position (for example, the above genotypes starts at the 9th column) in all rows. You can insert any fields between ID and genotypes but the fields should be separated by spaces.

The option `--snpfile` specifies a marker file. The following example reads a marker file `snpdata.txt` and checks the format, applies basic quality control to the genomic data, and writes a report.

```
qcf90 --snpfile snpdata.txt
```

By default, the program perfroms through the following menu.

1. Check for format for genomic data
2. Calculate call rate, allele frequency, and Hardy-Weinberg statistics for all available markers
3. Remove markers with low call rate
4. Remove markers with low allele frequency
5. Remove markers with 0 or 1 allele frequency (monomorphic markers)
6. Calculate call rate for animals using remained markers
7. Remove animals with low call rate
8. Recalculate allele frequency with remaining markers and animals
9. Make a report

The criteria to remove markers and animals can be supplied with options. Once a marker or an animal is marked as "unqualified", such markers or animals will be removed in the subsequent calculation. `QCF90` will stop if there is a serious problem (e.g. format error) in the file. In this case, the program will show a detailed explanation of what is wrong and how to fix the problem.

During the quality control, the following 5 files will be created by default.

- `freqdata.count` = allele frequency for all available markers (saved in the step 2 above)
- `Gen_call_rate` = call rate for animals (saved in the step 6)
- `freqdata.count.after.clean` = allele frequency after quality control (saved in the step 8)
- `qcf90.status` = status for markers and animals (saved in the step 9)
- `qcf90.log` = a copy of the message displayed on screen

The first 3 files are compatible with the files created with `PREGSF90`. The status file is readable and contains the test-result for each marker and animal.

2.2.2 Pedigree data

The program can check pedigree information without genomic data. The option `--pedfile` specifies a pedigree file that has at least 3 fields for animal, sire, and dam identification codes. Hereafter, we use `pedigree.txt` as a pedigree file with the following lines.

```
A70269 A58649 A63290
A44516 A36831 A29636
A7769 A8 A999
A19781 A14427 A8539
A52523 A32283 A25587
```

By default, the 1st field is for animal ID, the 2nd field for sire ID, and the 3rd field for dam ID (you can change the order using an option). The ID can contain any alphabets, symbols, and numbers (but tested only with ASCII characters). The following command entirely checks the pedigree information.

```
qcf90 --pedfile pedigree.txt
```

The program checks 1) missing entries i.e. an animal that appears as a parent in any entries but doesn't have its entry, 2) sire-dam inconsistency i.e. an animal that appears as a sire in an entry but as a dam in another entry, 3) pedigree loops i.e. an animal that appears in its ancestor. The program shows warning with the case 1 and 2; it stops in the case 3.

`QCF90` writes a file `qc_pedigree_loop.dot` if there are pedigree loops. An external software, GraphViz (http://www.graphviz.org) can read this file and draw a graph that shows the loops. This helps the user to find a problematic animal in a pedigree loop.

2.2.3 Both genomic and pedigree data

The user can specify both options in the same command line. The program will check Mendelian inconsistency in addition to the standard genomic and pedigree checks.

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt
```

The program checks Mendelian inconsistency (conflicts) among pairs of parent-progeny in addition to separate quality control for genomic and pedigree data. This is performed just after removing animals with low call rate (after the step 7 in the genomic quality control shown above).

The program assumes the pedigree is correct and compares genotypes between a parent and its progeny. The algorithm is described in Hayes (2011) and is equivalent to the implementation in `PREGSF90`. The following file will be created as a report.

- `Gen_conflicts` = detailed information for Mendelian checks

Note that the animal ID must be common both in genomic and pedigree files. This is not a case if the pedigree file is prepared with `RENUMF90`. If using such a file, the user should supply the XrefID file (See the next section).

2.2.4 Already renumbered files

QCF90 accepts files prepared ("renumbered") with RENUMF90. QCF90 recognizes this situation by specifying a XrefID file using the option --xrefid. In such case, the program cares both the original animal ID (in genomic data) and the renumbered ID (in pedigree data). The following example uses the XrefID file.

```
qcf90 --snpfile snpdata.txt --pedfile renadd02.ped --xrefid snpdata.txt_XrefID
```

The file renadd02.ped is supposed to be generated with RENUMF90.

2.2.5 Optional map file

QCF90 can read a map file for markers. This is useful when some markers are on the sex chromosomes that should be excluded from the Mendelian checks. The map file has at least 3 fields with white-space separators: 1st field as the sequential integer number of the marker, the 2nd field as the chromosome number (integer), and the 3rd field as the physical location on the chromosome. The map file can have the optional 4th field for the marker name. Note that qcf90 uses only the chromosome code (the 2nd column) and you can put any data in the other columns. The following is an example of map file; see the BLUPF90 manual for details.

```
   1  1    125
   2  1   1028
   3  1   3372
(snip)
5720  7  63701
5721  7  65417
```

The option --mapfile specifies the map file and --sex-chr determines the sex-chromosome numbers. See the following example.

```
qcf90 --snpfile snpdata.txt --mapfile map.txt --sex-chr 30
```

In this example, QCF90 relates a marker and a chromosome and knows the chromosome number 30 is sex-related. Those markers on the sex chromosomes will be temporarily excluded from the Mendelian checks. The option --sex-chr accepts multiple numbers separated by comma (or space) like --sex-chr 30,31 or --sex-chr 30 31.

2.3 Status file and clean files

QCF90 puts a status code (so-called "flag") for unqualified markers and animals and removes them from the subsequent quality controls. By default, at the end of the program, QCF90 doesn't change the original files but it reports the flags in a status file qcf90.status. There are two ways to remove the unqualified markers and animals from the original files and generate new files to have the qualified subjects.

2.3.1 1-pass way

The first way is to do in 1-pass. If you put the option `--save-clean`, the program creates "clean" files containing only qualified markers and animals. See the following example.

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt --save-clean
```

All marker, XrefID, and map files will be newly created when the user specifies each of them. The name of a clean file is the original name plus a suffix `_clean`. In above case, the clean genomic file is `snpdata.txt_clean`. The user can change the suffix with an additional argument with `--save-clean`. For example, the following command changes the suffix to `.new`.

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt --save-clean .new
```

2.3.2 With a status file

The second option is to use the status file. When the user specifies a status file with `--statfile`, `QCF90` reads all flags from the file without performing any computations of quality control. In this case, the user can specify particular flags of markers and animals to be removed from the clean files. The following command creates clean files by removing markers with the flags 1, 2, and 5 and animals with the flag 1 (the command-line is wrapped but should be typed in one line).

```
qcf90 --snpfile snpdata.txt --pedfile pedigree.txt --save-clean
      --statfile qcf90.status
      --cleanup-marker-flags 1,2,5 --cleanup-animal-flags 1
```

This operation is useful for selective removal of markers and animals.

3 Options

Brief descriptions for frequently-used options are available with the `--help` option on the command line. All available options will be shown with `--long-help`.

3.1 Source file specification

3.1.1 SNP Marker file

```
--snpfile markerfile
```

The argument `markerfile` is the name of a marker file. The file format should be the same to the one used in the BLUPF90 programs. See the BLUPF90 manual or Technical Details in this manual for details.

The program can perform the quality control on markers and animals: physical format error, low call rate, low allele frequency, and Hardy-Weinberg equilibrium test. Tests for Mendelian inconsistency is available only when a pedigree file is specified.

3.1.2 Pedigree file

```
--pedfile pedigreefile
```

The argument `markerfile` is the name of a pedigree file. The file must have at least 3 fields for an animal's ID, its sire's ID, and its dam's ID. The positions of IDs can be specified with another option (`--ped-position`). Missing parent must have a single character `0` or negative integer (`-` at the beginning of ID and integer code in the subsequent part) and any other characters will not be accepted as missing by default. For example, Any of `00`, `NA`, or `.` are never recognized as a missing animal. The code for missing (unknown) parents can be changed with an option `--upg` (see later sections).

The pedigree doesn't have to be prepared ("renumbered") with `RENUMF90`. Any characters will be used as ID (except `0` for missing animals). If a genomic file is also supplied, the ID must be common in both files. This is not a case when the "renumbered" pedigree is used. In such case, put `--xrefid` with a XrefID file.

The program can perform the following checks for pedigree: physical format error, confusion of sire and dam, missing entry of parents, and pedigree loops. The structure of pedigree loops is saved in a file `qc_pedigree_loops.dot` that Graphviz (http://www.graphviz.org) can read and draw the pedigree chart. Tests for Mendelian inconsistency is available only when a genotype file is specified.

3.1.3 Cross-reference ID (XrefID) file

```
--xrefid xrefidfile
```

The argument `xrefidfile` is the name of a cross-reference ID (XrefID) file generated with `RENUMF90`. This option is required only when using the "renumbered" pedigree created with `RENUMF90`.

3.1.4 Map file

```
--mapfile mapfile
```

The argument `mapfile` is the name of a map file. The marker file is needed when markers on sex chromosomes should be excluded in Mendelian checks.

The file format should be the same to the one used in the BLUPF90 programs. Briefly, the file should contain at least 3 fields: 1) a sequential number, 2) a chromosome number, and 3) a physical location on the chromosome. All numbers should be integer values. The additional 4th field can have any characters for the marker name. The number of rows in this file should be the same to the number of markers in the marker file.

3.1.5 Pre-calculated allele frequency (MAF) file

```
--maffile maffile
```

The argument `maffile` is the name of a file containing pre-calculated allele frequency for each marker. The file contains at least 1 field and the first field should be allele frequency, a real number. The number of rows in this file should be the same to the number of markers in the marker file.

3.1.6 Pre-generated status file

```
--statfile statusfile
```

The argument `statfile` is the name of a status file generated with `qcf90` in the previous run. `qcf90` reads status (qualifications) on markers and animals instead of calculating quality-statistics from the data. Then `qcf90` writes "clean" files without unqualified markers and animals. Therefore, this option needs the marker file specified with `--snpfile`.

With this option, `qcf90` doesn't create a new status file.

3.2 Quality-control for markers and animals

3.2.1 Quality control items

```
--qc items
```

The arguments `items` specify which quality control will be applied to the genomic data. Multiple arguments can be listed separating by a comma (`,`) or a white space ( ). The following items are acceptable.

Item	Default	Criterion	Description
`crm`	Yes	0.10	low call rate for markers
`maf`	Yes	0.05	low allele frequency (MAF) for markers
`mono`	Yes	0.00	monomorphic markers (0 or 1 MAF)
`hwe`	No	15.00	Hardy-Weinberg test for markers
`cra`	Yes	0.10	low call rate for animals
`par`	Yes	0.01	Mendelian inconsistency for markers and animals

Qualifying criteria can be changed with the other options (see below for details). Detailed explanations for each quality control is available in Technical Details.

3.2.2 Criterion of call rate for markers

```
--crm n
```

The argument n specifies the criterion of unqualified call rate for markers. The criterion should be a real number between 0 and 1 and the default value is 0.10. If the call-rate is lower than this value, the marker is recognized as "unqualified" and will be removed from the subsequent quality control.

3.2.3 Criterion of call rate for animals

```
--cra n
```

The argument n specifies the criterion of unqualified call rate for animals. The criterion should be a real number between 0 and 1 and the default value is 0.10. If the call-rate is lower than this value, the animal is recognized as "unqualified" and will be removed from the subsequent quality control.

3.2.4 Criterion of allele frequency for markers

```
--maf n
```

The argument n specifies the criterion of allele frequency for markers. The criterion should be a real number between 0 and 1 and the default value is 0.10. If the AF is lower or 1 - AF is bigger than this value, the marker is recognized as "unqualified" and will be removed from the subsequent quality control.

3.2.5 Criterion of Hardy-Weinberg statistic for markers

```
--hwe n
```

The argument n specifies the criterion of unqualified Hardy-Weinberg statistic for markers. The criterion should be a real number between 0 and 1 and the default value is 0.10. If the statistic is bigger than this value, the marker is recognized as "unqualified" and will be removed from the subsequent quality control.

3.2.6 List for unqualified markers

```
--exclude-marker-list markerlist
```

This option forces specific markers supplied by `markerlist` be unqualified. In other words, the markers will be marked as unqualified and removed from the quality control. This is useful especially when you have known some markers should be removed from the marker file. Note that the file format depends on which other options are specified.

- If a map file is used, `markerlist` must be the same format to the map file.
- Otherwise, `markerlist` contains integer numbers corresponding to the positions of markers in the genomic file.

3.2.7 List for unqualified animals

`--exclude-animal-list animallist`

This option forces specific animal supplied by `animallist` be unqualified. In other words, the animals will be marked as unqualified and removed from the quality control. This is useful especially when you have known some animals should be removed from the marker file. Note that the file format depends on which other options are specified.

- If a cross-reference ID (XrefID) file is used, `animallist` must be the same format to the XrefID file.
- Otherwise, `animallist` contains integer numbers corresponding to the positions of the animals in the genomic file (i.e. in which rows the animals locate).

3.2.8 Removal of unqualified markers

`--remove-markers`
`--no-remove-markers`

By default, unqualified markers will be removed from the subsequent quality control. The option (`--remove-markers`) supports this default behavior. If the other option `--no-remove-markers` is specified, the unqualified markers will NOT be removed from the subsequent quality control (in other words, the unqualified markers will be still used in all quality control) although such markers have a status, "unqualified".

3.2.9 Removal of unqualified animals

`--remove-animals`
`--no-remove-animals`

By default, unqualified animals will be removed from the subsequent quality control. The option (`--remove-animals`) supports this default behavior. If the other option `--no-remove-animals` is specified, the unqualified animals will NOT be removed from the subsequent quality control (in other words, the unqualified animals will be still used in all quality control) although such animals have a status, "unqualified".

3.2.10 Precise checks for input files

`--check-format`
`--no-check-format`

By default, the program precisely checks physical format for all specified files. The option (`--check-format`) supports this default behavior. If the other option `--no-check-format` turns off this feature and just quickly checks the format. It would save the running time if the user knows the file format is correct.

3.3 Checks for identical (duplicated) genotypes

3.3.1 Genomic identity search

```
--check-identity
--check-identity secondary_markerfile
```

This option checks identical genotypes among genotyped animals. Just specified `--check-identity` without an argument, the program will check all possible pairs of animals in the primary marker file supplied with `--snpfile`. For example, when the SNP file with n animals is supplied, the program compares $n(n-1)/2$ pairs of animals.

If the argument `secondary_markerfile` is specified, the program compares the genotypes between the primary and the secondary files. In this case, the program doesn't compare any pairs within the same file. For example, when the primary marker file has n animals and the secondary file has m animals, the number of comparisons is mn.

3.3.2 Threshold of identity

```
--threshold-identity n
```

This option specifies the threshold of how much similar two genotypes are as the identical genotypes. The identity statistic is calculated as the number of the identical genotypes divided by the number of compared genotypes excluding missing. When the statistic exceeds the threshold, two genotypes are considered as duplicated. The default value is 0.99.

3.3.3 Use of randomly selected markers

```
--random-markers-identity n
```

The identity check takes a long time if all the markers are used in the check. This option defines the number of markers used in the operation. The markers are randomly selected from all possible loci.

3.3.4 Use of user-specified markers

```
--fixed-markers-identity markerlist
```

This option specifies a set of markers used in the identity check. The user should supply the name of the marker list. The format of the list file is the same as explained in the option `--exclude-marker-list` (see this option for details).

3.3.5 Marker ID used in the check

```
--save-markers-identity
```

This option saves the marker ID used in the identity check. It may be useful when you use the random markers.

3.4 Chromosome configuration

3.4.1 Sex chromosome

```
--sex-chr n..
```

The arguments `n..` specify the numbers corresponding to the sex-chromosome code defined in a map file. With this option, the map file must be also supplied. Multiple arguments can be listed separating by a comma (`,`) or a white space ( ).

The markers on the sex chromosomes will be temporarily excluded from the Mendelian checks.

3.4.2 Removal of markers on particular chromosome

```
--exclude-chr n..
```

The arguments `n..` specify the numbers corresponding to the chromosome code defined in a map file. With this option, the map file must be also supplied. Multiple arguments can be listed separating by a comma (`,`) or a white space ( ).

The markers on the chromosomes will be permanently removed and marked as *removed-by-user*.

3.5 Pedigree configuration

3.5.1 Position of animals

```
--ped-position a,s,d
```

The arguments `a` specifies the position of an animal's ID in a pedigree file and `s` for the sire's ID, and `d` for the dam's ID. All three positions should be different and must exist in the pedigree file.

3.5.2 Unknown parents

```
--upg zero
--upg negative
--upg zero+negative
```

The argument specifies what kind of ID should be treated as missing (unknown) parents.

Argument	Missing parents
`zero`	a single `0`
`negative`	the negative sign `-` and integer code in the subsequent part
`zero+negative`	a combination of above two (default)

3.6 Job control and performance options

3.6.1 Script file

`--script file`

`QCF90` reads options from the script `file` specified here. Each row in the script has one option and its argument(s).

Even with this option, `QCF90` still reads options from the command line. If the same option is specified in both sources, the command line option will be used.

3.6.2 Strict check

`--halt-on-warning`

`QCF90` give you warnings and continues the process if the issue is not crucial. This option make the program stop when any small problems are detected.

3.6.3 Dry run

`--dry-run`

`QCF90` shows what is actually happening with specified options without any actual computations nor modifications. This is useful just to make sure the options work correctly as the user intends to do.

3.6.4 Fast reading the marker file

`--fastread`

This option accelerate to read a marker file when using the packed format. This is always recommended for any users unless there is any trouble.

3.6.5 Internal format of marker-storage on memory

`--marker-storage type`

The argument `type` specifies which marker-storage strategy is used in this program. The following types are available.

Type	Default	Description
real	No	Use a double precision (8-byte) variable per marker
int	No	Use a 1-byte integer variable per marker
pack	Yes	Use a 4-byte integer variable per 16 markers

The user doesn't have to care much about this option. See the Technical Details section for details.

3.6.6 Number of threads

```
--num-threads n
```

This option defines the number of threads used in this program if a functionality supports multi-threading. Currently, only the identity (duplication) check supports the parallel computation.

3.7 File output options

3.7.1 log file

```
--save-log file
--no-save-log
```

QCF90 saves all the log to a file `qcf90.log` by default. This option can change the name of the log file. You can also suppress to create the log with `--no-save-log`.

3.7.2 status file

```
--save-status file
--no-save-status
```

QCF90 creates a status file `qcf90.status` by default. This option can change the name of the status file. You can also suppress to create the log with `--no-save-log`.

3.7.3 "dot" file

```
--save-dot file
--no-save-dot
```

QCF90 creates a "dot" file by default if the pedigree information has a loop. The dot file contains a graph written in "dot" language which is interpreted by the software, Graphviz (http://www.graphviz.org). The file name is the original pedigree file + `dot`. This option can change the name of the dot file. You can also suppress to create the log with `--no-save-dot`.

3.7.4 Create clean files

```
--save-clean
--save-clean suffix
```

QCF90 saves "clean" files after removing all unqualified markers and animals from the original genomic, map, and the cross-reference ID (XrefID) files. The name of a new file is the same to the original file with a suffix `_clean`. If an optional argument `suffix` is supplied, the program will use those characters as the suffix.

3.7.5 Removing markers with specific status

`--cleanup-marker-flags n..`

When saving clean files, the program will remove only particular markers with the status specified here. The arguments `n..` are integer status codes (flags) to describe qualification of a marker. Multiple arguments can be listed separating by a comma (`,`) or a white space ( ). See the Technical Details for the status code.

3.7.6 Removing animals with specific status

`--cleanup-animal-flags n..`

When saving clean files, the program will remove only particular animals with the status specified here. The arguments `n..` are integer status codes (flags) to describe qualification of an animal. Multiple arguments can be listed separating by a comma (`,`) or a white space ( ). See the Technical Details for the status code.

3.8 Miscellaneous

3.8.1 Version number

`--version`

QCF90 shows the current version of the program and immediately stops.

3.8.2 Help message

`--help`

QCF90 shows a list of frequently-used options and stops.

3.8.3 Comprehensive help message

`--long-help`

QCF90 shows a list of all available options and stops.

4 Technical details

4.1 Algorithms

The program uses bit-wise operations on the packed markers. The computational detail is explained in Masuda et al. (in review).

4.2 Work flow of the program

The following is the complete description of the workflow. Some steps are ignored unless appropriate options are specified (e.g. reading a pedigree file only with the `--ped` option). The unqualified markers in a quality-control step will be excluded from the subsequent quality control. If serious problems or errors are found, the program immediately stops with a diagnosis.

- Initialization
  - Read a script file (`--script`)
  - Read the command line
  - Check inconsistency or error in options
- Quick checks for files and some preparations
  - Read the first line of pedigree file and check the format
  - Read the first line of marker file and check the format
  - Read through the genomic file and count the number of lines
  - Prepares a hash table to store ID for genotyped animals
- Precise checks for genomic data
  - Read the first line of marker file and check the format, again
  - Read through the genomic data and precisely check the format
- Load genomic data to memory
  - Allocate memory for genomic data
  - Read genomic data from the file and store it to memory
- Precise checks for pedigree
  - Read entire pedigree from the file and store it to memory
  - Check missing entries for parents
  - Check sire-dam inconsistency
  - Check loops in pedigree
- Precise checks for combined genomic and pedigree data
  - Check genotyped animals not found in pedigree

- Checks for identical genotypes
  - Within/between file(s)
- Computations of statistics for markers
  - Allele frequency
  - Call rate
  - Hardy-Weinberg statistic
- Removal of specific markers specified by user
  - Individual marker
  - Markers on particular chromosomes
- Quality control for markers
  - Low call rate (criterion<0.10; changed by `--crm`)
  - Low allele frequency (criterion<0.10; changed by `--maf`)
  - monomorphic markers
  - Hardy-Weinberg test
- Quality control for animals
  - Low call rate
- Quality control for Mendelian conflicts
  - Calculate Mendelian consistency for markers
  - Low-consistency markers
  - Calculate Mendelian consistency for animals
  - Low-consistency animals
- Post quality-control process
  - Calculate allele frequency with qualified markers and animals
  - Save status for markers and animals to a file
  - Save clean files

4.3 File format

4.3.1 Status file

The program creates a status file as a report of the quality control. It has which markers and animals are determined as unqualified and the reason why they are. It is a human-readable text file consisting of several sections. You can easily extract specific information out of it; but do not edit the file.

Comment section A *comment* is a line that will be ignored from the program. The comment line starts with the character # and the entire line will be ignored. At the beginning of the file, all human-readable information is in a comment section. The following example illustrates how the information is placed in the block.

```
# Quality-control status
# Oct 6, 2017 22:22:14
#
# marker file   = newsnp
#
# marker quality:
#    flag 0 = OK                   :         83 markers (checked)
#    flag 1 = Call Rate            :          0 markers (checked)
#    flag 2 = MAF                  :         17 markers (checked)
#    flag 3 = Monomorphic          :          0 markers (checked)
#    flag 4 = Excluded by user     :          0 markers (not checked)
#    flag 5 = Mendelian error      :          0 markers (not checked)
#    flag 6 = HWE                  :          0 markers (not checked)
#    flag 7 = High Correlation     :          0 markers (not checked)
#    flag 8 = Same Position        :          0 markers (not checked)
# animal quality:
#    flag 0 = OK                   :         15 animals (checked)
#    flag 1 = Call Rate            :          0 animals (checked)
#    flag 2 = Parent-progeny confl :          0 animals (not checked)
#    flag 3 = Diagonal outlier     :          0 animals (not checked)
#    flag 4 = Excluded by user     :          0 animals (not checked)
#
```

In this specific example, each marker took 3 tests (call rate, MAF, and monomorphic tests marked as `checked`) and each animal took only 1 test (call rate). The marker file has 100 SNP markers and 15 animals; of which, 17 markers are unqualified but no animals are unqualified. The comment also gives the user an instruction how to remove such unqualified objects from the marker file using this status file.

Variable section A structure of the marker file is saved just after the comment block. The number of markers and the number of animals are saved in the current version.

```
NUMBER_OF_MARKERS   100
NUMBER_OF_GENOTYPES 15
```

Marker status section The marker status section shows which marker is unqualified and why. It has 3 fields: 1) the sequential number of the marker, 2) the flag, and 3) the meaning of the flag. See the following example.

```
MARKER_STATUS
      1  0 OK
      2  2 MAF
      3  0 OK
      (snip..)
     99  2 MAF
    100  2 MAF
```

Animal status section Like the marker status, the animal status section shows which animal is unqualified and why. It also has 3 fields: 1) the sequential number of the animal, 2) the flag, 3) the name of the animal in the genomic data and 4) the meaning of the flag. See the following example.

```
ANIMAL_STATUS
      1  0 1 OK
      2  0 2 OK
      3  0 3 OK
      (snip...)
     14  0 14 OK
     15  0 15 OK
```

4.3.2 SNP marker file

The snp-marker file is a text file which has at least 2 fields. The fiest field is for individual ID and the last field is for genotypes. Each field should be separated with one or more white space. The following is a simple example of marker file.

```
 A001  120120120120
 A002  120120120120
 A003  120120120120
```

The field of genotypes in each row should start from the same column. The leading and trailing spaces will be ignored. For example, in the above example, the genotypes start from the 8th column in all rows. In contrast to genotypes, the individual ID is not necessarily aligned. The following file should be fine for the program (but, again, genotypes should be aligned).

```
 A001   120120120120
A002    120120120120
  A003  120120120120
```

You can insert extra fields between ID and genotypes as long as genotypes are alined to a particular column. The following example is also valid.

```
 A001 10 M  120120120120
A002 10 M   120120120120
  A003 11 F 120120120120
```

There are two possible formats of genotypes.

- Integer genotypes: Each genotype has 3 possible values (`0`,`1`, and `2`) and a missing code (`5`). One genotype occupies 1 letter in the field. No spaces bwteen genotypes are allowed.
- Real genotypes: Each genotype can be a real number with a fixed width like `1.00` and `0.95`. To make a compatibility with BLUPF90, the width should be 4 and the number of digits under dot should be 2. No spaces between genotypes are allowed.

The following is an exmple of real genotypes.

```
A001  1.002.000.001.002.000.00
A002  1.002.000.001.002.000.00
A003  1.002.000.001.002.000.00
```

You must not mix two formats in the same file. The number of genotypes should be the same for all individuals.

The integer genotypes will be efficiently stored in memory but real genotypes are not. The bitwise algorithms are applicable only to the integer genotypes.

4.3.3 Allele frequencies

The allele frequencies are computed with the original genotypes (saved to `freqdata.count`) and the clean genotypes (`freqdata.count.after.clean`) if applicable. The file has two columns: 1) the marker number and 2) its allele frequency. The marker number is a sequential integer-code which ordered the same as the genomic file. See the following example.

```
  1  0.362348
  2  0.587222
  3  0.610498
  4  0.783390
  5  0.730564
  6  0.096242
  7  0.862672
  8  0.717877
  9  0.387042
 10  0.174393
```