MVtest Documentation

Release 1.0.0

Todd Edwards, Chun Li and Eric Torstenson

March 03, 2016

CONTENTS

1 Installation **1**
- 1.1 Install with PIP . . . 1
- 1.2 Manual Installation . . . 1
- 1.3 System Requirements . . . 2
- 1.4 Running Unit Tests . . . 2
- 1.5 Virtual Env . . . 2
- 1.6 Miniconda . . . 2

2 What is MVtest? **5**
- 2.1 Documentation . . . 5
- 2.2 Command-Line Arguments . . . 5

3 mvmany Helper script **13**
- 3.1 The Default Template . . . 13
- 3.2 Command Line Arguments . . . 14

4 Development Notes **17**
- 4.1 MVtest authors . . . 17
- 4.2 Change Log . . . 17

Index **19**

CHAPTER

ONE

INSTALLATION

MVtest requires python 2.7.x as well as the following libraries:

- NumPy (version 1.7.2 or later) www.numpy.org
- SciPY (version 0.13.2 or later) www.scipy.org

MVTest's installation will attempt to install these required components for you, however, it requires that you have write permission to the installation directory. If you are using a shared system and lack the necessary privileges to install libraries and software yourself, you should please see one of the sections, *Miniconda* or *virtual-env* below for instructions on different options for setting up your own python environement which will exist entirely under your own control.

Installation can be done in two ways:

1.1 Install with PIP

To install using python's package manager, pip, simply use the following command:

$ *pip install MVtest*

If you have proper permission to install packages, this will attempt to download and install all dependencies along with MVtest itself.

1.2 Manual Installation

For users who do not use pip or wish to run the bundled tests as well as have a local copy of the manuals, manual installation is almost as easy.

For users with Git installed, you can simply clone the sources using the following command:

$ *git clone https://github.com/edwards-lab/MVtest*

Or you may visit the website and download the tarball directly from github: https://github.com/edwards-lab/MVtest

Once you have downloaded the software, simply extract the contents and run the following command to install it:

$ *python setup.py install*

If no errors are reported, it should be installed and ready to use.

Regarding PYTHON 3 I began the process of updating the code to work with both python versions 2 and 3, however, there are some real issues with some library support of version 3 that is discouraging. So, until those have been resolved, I have no plans to invest further time toward support for python 3.

1.3 System Requirements

Aside from the library dependencies, MVTest's requirements depend largely on the number of SNPs and individuals being analyzed as well as the data format being used. In general, GWAS sized datasets will require several gigabytes of memory when using the traditional pedigree format, however, even 10s of thousands of subjects can be analyzed with less than 1 gigabyte of RAM when the data is formatted as transposed pedigree or PLINK's default bed format.

Otherwise, it is recommended that the system be run on a unix-like system such as Linux or OS X, but it should work under windows as well (we can't offer support for running MVTest under windows).

1.4 Running Unit Tests

MVTest comes with a unit test suite which can be run prior to installation. To run the tests, simply run the following command from within the root directory of the extracted archive's contents:

$ *python setup.py test*

If no errors are reported, then mvtest should run correctly on your system.

1.5 Virtual Env

Virtual ENV is a powerful too for python programmers and end users alike as it allows for users to deploy different versions of python applications without the need for root access to the machine.

Because MVTest requires version 2.7, you'll need to ensure that your machine's python version is in compliance. Virtual Env basically uses the the system version of python, but creates a user owned environment wrapper allowing users to install libraries easily without administrative rights to the machine.

For a helpful introduction to VirtualEnv, please have a look at the tutorial: http://www.simononsoftware.com/virtualenv-tutorial/

1.6 Miniconda

Miniconda is a minimal version of the package manager used by the Anaconda python distribution. It makes it easy to create local installations of python with the latest versions of the common scientific libraries for users who don't have root access to their target machines. Basically, when you use miniconda, you'll be installing your own version of Python into a directory under your control which allows you to install anything else you need without having to submit a helpdesk ticket for administrative assistance.

Unlike pip, the folks behind the conda distributions provide binary downloads of it's selected library components. As such, only the most popular libraries, such as pip, NumPY and SciPy, are supported by conda itself. However, these do not require compilation and may be easier to get installed than when using pip alone. I have experienced difficulty installing SciPy through pip and setup tools on our cluster here at vanderbilt due to non-standard paths for certain required components, but mini-conda always comes through.

Firstly, download and install the appropriate version of miniconda at the project website. Please be sure to choose the Python 2 version: http://conda.pydata.org/miniconda.html

While it is doing the installation, please allow it to update your PATH information. If you prefer not to always use this version of python in the future, simple tell it not to update your .bashrc file and note the instructions for loading and unloading your new python environment. Please note that even if you chose to update your .bashrc file, you will need to follow directions for loading the changes into your current shell.

Once those changes have taken effect, install setuptools and scipy: $ *conda install pip scipy*

Installing SciPy will also force the installation of NumPy, which is also required for running mvtest. (setuptools includes easy_install).

Once that has been completed successfully, you should be ready to follow the standard instructions for installing mvtest.

CHAPTER

TWO

WHAT IS MVTEST?

TODO: Write some background information about the application and it's scientific basis.

2.1 Documentation

Documentation for MVtest is still under construction. However, the application provides reasonable inline help using standard unix help arguments:

> *mvtest.py -h*

or

> *mvtest.py --help*

In general, overlapping functionality should mimic that of PLINK.

2.2 Command-Line Arguments

Command line arguments used by MVtest often mimick those used by PLINK, except where there is no matching functionality (or the functionality differs significantly.)

For the parameters listed below, when a parameter requires a value, the value must follow the argument with a single space separating the two (no '=' signs.) For flags with no specified value, passing the flag indicates that condition is to be "activated".

When there is no value listed in the "Type" column, the arguments are *off* by default and *on* when the argument is present (i.e. by default, compression is turned off except when the flag, --compression, has been provided.)

2.2.1 Getting help

`-h, --help`
: Show this help message and exit.

`-v`
: Print version number

2.2.2 Input Data

MVtest attempts to mimic the interface for PLINK where appropriate.

All input files should be whitespace delimited. For text based allelic annotations, 1|2 and A|C|G|T annotation is sufficient. All data must be expressed as alleles, not as genotypes (except for IMPUTE output, which is a specialized format that is very different from the other forms).

For Pedigree, Transposed Pedigree and PLINK binary pedigree files, the using the PREFIX arguments is sufficient and recommended if your files follow the standard naming conventions.

Pedigree Data

Pedigree data is fully supported, however it is not recommended. When loading pedigree data, MVtest must load the entire dataset into memory prior to analysis, which can result in a substantial amount of memory overhead that is unnecessary.

Flags like --no-pheno and --no-sex can be used in any combination creating MAP files with highly flexible header structures.

`--file <prefix>`
: (filename prefix) Prefix for .ped and .map files

`--ped <filename>`
: PLINK compatible .ped file

`--map <filename>`
: PLink compatible .map file

`--map3`
: Map file has only 3 columns

`--no-sex`
: Pedigree file doesn't have column 5 (sex)

`--no-parents`
: Pedigree file doesn't have columns 3 and 4 (parents)

`--no-fid`
: Pedgiree file doesn't have column 1 (family ID)

`--no-pheno`
: Pedigree file doesn't have column 6 (phenotype)

`--liability`
: Pedigree file has column 7 (liability)

PLINK Binary Pedigree

This format represents the most efficient storage for large GWAS datasets, and can be used directly by MVtest. In addition to a minimal overhead, plink style bed files will also run very quickly, due to the efficient disk layout.

`--bfile <prefix>`
: (filename prefix) <prefix> for .bed, .bim and .fam files

`--bed <filename>`
: Binary Ped file(.bed)

`--bim <filename>`
: Binary Ped marker file (.bim)

`--fam <filename>`
: Binary Ped family file (.fam)

Transposed Pedigree Data

Transposed Pedigree data is similar to standard pedigree except that the data is arranged such that the data is organized as SNPs as rows, instead of individuals. This allows MVtest to run it's analysis without loading the entire dataset into memory.

`--tfile <prefix>`
: Prefix for .tped and .tfam files

`--tped <filename>`
: Transposed Pedigre file (.tped)

`--tfam <filename>`
: Transposed Pedigree Family file (.tfam)

Pedigree/Transposed Pedigree Common Flags

By default, Pedigree and Transposed Pedigree data is assumed to be uncompressed. However, MVtest can directly use gzipped data files if they have the extension .tgz with the addition of the --compressed argument.

`--compressed`
: Indicate that ped/tped files have been compressed with gzip and are named with extensions such as .ped.tgz and .tped.tgz

IMPUTE output

MVtest doesn't call genotypes when performing analysis and allows users to define which model to use when analyzing the data. Due to the fact that there is no specific location for chromosome within the input files, MVtest requires that users provide chromosome, impute input file and the corresponding .info file for each imputed output.

Due to the huge number of expected loci, MVtest allows users to specify an offset and file count for analysis. This is to allow users to run multiple jobs simultaneously on a cluster and work individually on separate impute region files. Users can segment those regions even further using standard MVtest region selection as well.

By default, all imputed data is assumed to be compressed using gzip.

Default naming convention is for impute data files to end in .gen.gz and the info files to have the same name except for the end being replaced by .info.

`--impute <filename>`
: File containing list of impute output for analysis

`--impute-fam <filename>`
: File containing family details for impute data

`--impute-offset <integer>`
: Impute file index (1 based) to begin analysis

`--impute-count <integer>`
: Number of impute files to process (for this node). Defaults to all remaining.

`--impute-uncompressed`
: Indicate that the impute input is not gzipped, but plain text

`--impute-encoding`
: (additive,dominant or recessive)

  Genetic model to be used when analyzing imputed data.

`--impute-info-ext <extension>`
: Portion of filename denotes info filename

`--impute-gen-ext <extension>`
: Portion of filename that denotes gen file

`--impute-info-thresh <float>`
: Threshold for filtering imputed SNPs with poor 'info' values

IMPUTE File Input

When performing an analysis on IMPUTE output, users must provide a single file which lists each of the gen files to be analyzed. This plain text file contains 2 (or optionally 3) columns for each gen file:

Chromosome	Gen File	.info <filename> (optional)
N	<filename>	<filename>
...	...	...

The 3rd column is only required if your .info files and .gen files are not the same except for the <extension>.

MACH output

Users can analyze data imputed with MACH. Because most situations require many files, the format is a single file which contains either pairs of dosage/info files, or, if the two files share the same filename except for extensions, one dosage file per line.

Important: MACH doesn't provide anywhere to store chromosome and positions. Users may wish to embed this information into the first column inside the .info file. Doing so will allow MVtest to recognize those values and populate the corresponding fields in the report.

To use this feature, users much use the --mach-chrpos field and their ID columns inside the .info file must be formatted in the following way:

chr:pos (optionally :rsid)

When the --mach-chrpos flag is used, MVtest will fail when it encounters IDs that aren't in this format and there must be at least 2 'fields' (i.e. there must be at least one ":" character.

When processing MACH imputed data without this special encoding of IDs, MCtest will be unable to recognize positions. As a result, unless the --mach-chrpos flag is present, MVtest will exit with an error if the user attempts to use positional filters such as --from-bp, --chr, etc.

When running MVtest using MACH dosage on a cluster, users can instruct a given job to anlyze data from a portion of the files contained within the MACH dosage file list by changing the --mach-offset and --mach-count arguments. By default, the offset starts with 1 (the first file in the dosage list) and runs all it finds. However, if one were to want to split the jobs up to analyze three dosage files per job, they might set those values to --mach-offset 1 --mach-count 3 or --mach-offset 4 --mach-count 3 depending on which job is being defined.

In order to minimize memory requirements, MACH dosage files can be loaded incrementally such that only N loci are stored in memory at a time. This can be controlled using the --mach-chunk-size argument. The larger this number is, the faster MVtest will run (fewer times reading from file) but the more memory is required.

`--mach <filename>`
: File containing list of dosages, one per line. Optionally, lines may contain the info names as well (separated by whitespace) if the two <filename>s do not share a common base name.

`--mach-offset <integer>`
: Index into the MACH file to begin analyzing

`--mach-count <integer>`
: Number of dosage files to analyze

`--mach-uncompressed`
: By default, MACH input is expected to be gzip compressed. If data is plain text, add this flag. *It should be noted that dosage and info files should be either both compressed or both uncompressed.*

`--mach-chunk-size <integer>`
: Due to the individual orientation of the data, large dosage files are parsed in chunks in order to minimize excessive memory during loading

`--mach-info-ext <extension>`
: Indicate the <extension> used by the mach info files

`--mach-dose-ext <extension>`
: Indicate the <extension> used by the mach dosage files

`--mach-min-rsquared <float>`
: Indicate the minimum threshold for the rsqured value from the .info files required for analysis.

`--mach-chrpos`
: When set, MVtest expects IDs from the .info file to be in the format chr:pos:rsid (rsid is optional). This

will allow the report to contain positional details, otherwise, only the RSID column will have a value which will be the contents of the first column from the .info file

MACH File Input

When running an analysis on MACH output, users must provide a single file which lists of each dosage file and (optionally) the matching .info file. This file is a simple text file with either 1 column (the dosage filename) or 2 (dosage filename followed by the info filename separated by whitespace).

The 2nd column is only required if the filenames aren't identical except for the extension.

Col 1 (dosage <filename>)	Col 2 (optional info <filename>)
<filename>.dose	<filename>.info
...	...

Phenotype/Covariate Data

Phenotypes and Covariate data can be found inside either the standard pedigree headers or within special PLINK style covariate files. Users can specify phenotypes and covariates using either header names (if a header exists in the file) or by 1 based column indices. An index of 1 actually means the first variable column, not the first column. In general, this will be the 3rd column, since columns 1 and 2 reference FID and IID.

`--pheno <filename>`
: File containing phenotypes. Unless --all-pheno is present, user must provide either index(s) or label(s) of the phenotypes to be analyzed.

`--mphenos LIST`
: Column number(s) for phenotype to be analyzed if number of columns > 1. Comma separated list if more than one is to be used.

`--pheno-names LIST`
: Name for phenotype(s) to be analyzed (must be in --pheno file). Comma separated list if more than one is to be used.

`--covar <filename>`
: File containing covariates

`--covar-numbers LIST`
: Comma-separated list of covariate indices

`--covar-names LIST`
: Comma-separated list of covariate names

`--sex`
: Use sex from the pedigree file as a covariate

`--missing-phenotype CHAR`
: Encoding for missing phenotypes as can be found in the data.

`--all-pheno`
: When present, mv-test will run each phenotypes found inside the phenotype file.

2.2.3 Restricting regions for analysis

When specifying a range of positions for analysis, a chromosome must be present. If a chromosome is specified but is not accompanied by a range, the entire chromosome will be used. Only one range can be specified per run.

In general, when specifying region limits, --chr must be defined unless using generic MACH input (which doesn't define a chromosome number nor position, in which case positional restrictions do not apply).

`--snps LIST`
: Comma-delimited list of SNP(s): rs1,rs2,rs3-rs6

`--chr <integer>`
: Select Chromosome. If not selected, all chromosomes are to be analyzed.

`--from-bp <integer>`
: SNP range start

`--to-bp <integer>`
: SNP range end

`--from-kb <integer>`
: SNP range start

`--to-kb <integer>`
: SNP range end

`--from-mb <integer>`
: SNP range start

`--to-mb <integer>`
: SNP range end

`--exclude LIST`
: Comma-delimited list of rsids to be excluded

`--remove LIST`
: Comma-delimited list of individuals to be removed from analysis. This must

  be in the form of family_id:individual_id

`--maf <float>`
: Minimum MAF allowed for analysis

`--max-maf <float>`
: MAX MAF allowed for analysis

`--geno <integer>`
: MAX per-SNP missing for analysis

`--mind <integer>`
: MAX per-person missing

`--verbose`
: Output additional data details in final report

CHAPTER

THREE

MVMANY HELPER SCRIPT

In addition to the analysis program, mvtest.py, a helper script, mvmany.py is also included and can be used to split large jobs into smaller ones suitable for running on a compute cluster. Users simply run mvmany.py just like they would run mvtest.py but with a few additional parameters, and mvmany.py will build multiple job scripts to run the jobs on multiple nodes. It records most arguments passed to it and will write them to the scripts that are produced.

It is important to note that mvmany.py simply generates cluster scripts and does not submit them.

3.1 The Default Template

When mvmany.py is first run, it will generate a copy of the default template inside the user's home directory named .mv-many.template. This template is used to define the job details that will be written to each of the job scripts. By default, the template is configured for the SLURM cluster software, but can easily be changed to work with any cluster software that works similarly to the SLURM job manager, such as TORQUE/PBS or sungrid.

In addition to being able to replace the preprocessor definitions to work with different cluster manager software, the user can also add user specific definitions, such as email notifications or account specification, giving the user the the options necessary to run the software under many different system configurations.

3.1.1 Example Template (SLURM)

An example template might look like the following

```
#!/bin/bash
#SBATCH --job-name=$jobname
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=$memory
#SBATCH --time=$walltime
#SBATCH --error $logpath/$jobname.e
#SBATCH --output $respath/$jobname.txt

cd $pwd

$body
```

It is important to note that this block of text contains a mix of SLURM preprocessor settings (such as #SBATCH --job-name) as well as variables which will be replaced with appropriate values (such as $jobname being replaced with a string of text which is unique to that particular job). Each cluster type has it's own

syntax for setting the necessary variables and it is assumed that the user will know how to correctly edit the default template to suit their needs.

3.1.2 Example TORQUE Template

For instance, to use these scripts on a TORQUE based cluster, one might update ~/.mvmany.template to the following

```
#!/bin/bash
#PBS -N $jobname
#PBS -l nodes=1
#PBS -l ppn=1
#PBS -l mem=$memory
#PBS -l walltime=$walltime
#PBS -e $logpath/$jobname.e
#PBS -o $respath/$jobname.txt

cd $pwd

$body
```

Please note that not all SLURM settings have a direct mapping to PBS settings and that it is up to the user to understand how to properly configure their cluster job headers.

In general, the user should ensure that each of the variables are properly defined so that the corresponding values will be written to the final job scripts. The following variables are replaced based on the job that is being performed and the parameters passed to the program by the user (or their default values):

Variable	Purpose
$jobname	Unique name for the current job
$memory (2G)	Amount of memory to provide each job.
$walltime (3:00:00)	Define amount of time to be assigned to jobs
$logpath	Directory specified for writing logs
$respath	Directory sepcified for writing results
$pwd	current working dir when mvmany is run
$body	Statements of execution

3.2 Command Line Arguments

mvmany.py exposes the following additional arguments for use when running the script.

`--mv-path PATH`
: Set path to mvtest.py if it's not in PATH

`--logpath PATH`
: Path to location of job's error output

`--res-path PATH`
`Path to location of job's results`

`--script-path PATH`
: Path for writing script files

`--template FILENAME`
: Specify a template other than the default

`--snps-per-job INTEGER`
: Specify the number of SNPs to be run at one time

`--mem STRING`
: Specify the amount of memory to be requested for each job

`--wall-time`
: Specify amount of time to be requested for each job

The option, --mem, is dependent on the type of input that is being used as well as configurable options to be used. The user should perform basic test runs to determine proper settings for their jobs. By default, 2G is used, which is generally more than adequate for binary pedigrees, IMPUTE and transposed pedigrees. Others will vary greatly based on the size of the dataset and the settings being used.

The option, --wall-time, is largely machine dependent but will vary based on the actual dataset's size and completeness of the data. Users should perform spot tests to determine reasonable values. By default, the requested wall-time is 3 days, which is sufficient for a GWAS dataset, but probably not sufficient for an entire whole exome dataset and the time required will depend on just how many SNPs are being analyzed by any given node.

In general, mvmany.py accepts all arguments that mvtest.py accepts, with the exception of those that are more appropriately defined by mvmany.py itself. These include the following arguments

```
--chr
--snps
--from-bp
--to-bp
--from-kb
--to-kb
--from-mb
--to-mb
```

To see a comprehensive list of the arguments that mvmany.py can use simply ask the program itself

```
mvmany.py --help
```

Users can have mvmany split certain types of jobs up into pieces and can specify how many independent commands to be run per job. At this time, mvmany.py assumes that imputation data is already split into fragments and doesn't support running parts of a single file on multiple nodes.

The results generated can be manually merged once all nodes have completed execution.

CHAPTER

FOUR

DEVELOPMENT NOTES

4.1 MVtest authors

MVTest is written and maintained by Eric Torstenson <eric.s.torstenson@vanderbilt.edu> based on the algorithm developed by Todd Edwards <todd.l.edwards@vanderbilt> and Chun Li <cxl791@case.edu>.

Much of the command line interface is mimicks that of PLINK http://pngu.mgh.harvard.edu/~purcell/plink/ in order to make it easy for researchers to be able to quickly integrate MVTest into their workflow.

4.2 Change Log

mvtest.py: 1.0.3

- Removed special requirements for MACH input (chr:pos) and made that optional.

mvtest.py: 1.0.2

- added an exception when using improperly formatted MACH info file(s)
- updated documentation to draw attention to the additional MACH info file requirements

mvtest.py: 1.0.1 released

- changes to the setup.cfg and setup.py to accomodate changes made to work with gh-pages.

mvtest.py: 1.0.0 released

Symbols

--all-pheno
command line option, 10
--bed <filename>
command line option, 7
--bfile <prefix>
command line option, 7
--bim <filename>
command line option, 7
--chr <integer>
command line option, 11
--compressed
command line option, 7
--covar <filename>
command line option, 10
--covar-names LIST
command line option, 10
--covar-numbers LIST
command line option, 10
--exclude LIST
command line option, 11
--fam <filename>
command line option, 7
--file <prefix>
command line option, 6
--from-bp <integer>
command line option, 11
--from-kb <integer>
command line option, 11
--from-mb <integer>
command line option, 11
--geno <integer>
command line option, 11
--impute <filename>
command line option, 8
--impute-count <integer>
command line option, 8
--impute-encoding
command line option, 8
--impute-fam <filename>
command line option, 8
--impute-gen-ext <extension>
command line option, 8
--impute-info-ext <extension>
command line option, 8
--impute-info-thresh <float>
command line option, 8
--impute-offset <integer>
command line option, 8
--impute-uncompressed
command line option, 8
--liability
command line option, 6
--logpath PATH
command line option, 14
--mach <filename>
command line option, 9
--mach-chrpos
command line option, 9
--mach-chunk-size <integer>
command line option, 9
--mach-count <integer>
command line option, 9
--mach-dose-ext <extension>
command line option, 9
--mach-info-ext <extension>
command line option, 9
--mach-min-rsquared <float>
command line option, 9
--mach-offset <integer>
command line option, 9
--mach-uncompressed
command line option, 9
--maf <float>
command line option, 11
--map <filename>
command line option, 6
--map3
command line option, 6
--max-maf <float>
command line option, 11
--mem STRING
command line option, 15
--mind <integer>

command line option, 11
--missing-phenotype CHAR
command line option, 10
--mphenos LIST
command line option, 10
--mv-path PATH
command line option, 14
--no-fid
command line option, 6
--no-parents
command line option, 6
--no-pheno
command line option, 6
--no-sex
command line option, 6
--ped <filename>
command line option, 6
--pheno <filename>
command line option, 10
--pheno-names LIST
command line option, 10
--remove LIST
command line option, 11
--res-path PATH
command line option, 14
--script-path PATH
command line option, 14
--sex
command line option, 10
--snps LIST
command line option, 11
--snps-per-job INTEGER
command line option, 14
--template FILENAME
command line option, 14
--tfam <filename>
command line option, 7
--tfile <prefix>
command line option, 7
--to-bp <integer>
command line option, 11
--to-kb <integer>
command line option, 11
--to-mb <integer>
command line option, 11
--tped <filename>
command line option, 7
--verbose
command line option, 11
--wall-time
command line option, 15
-h, --help
command line option, 5
-v
command line option, 5

C

command line option
--all-pheno, 10
--bed <filename>, 7
--bfile <prefix>, 7
--bim <filename>, 7
--chr <integer>, 11
--compressed, 7
--covar <filename>, 10
--covar-names LIST, 10
--covar-numbers LIST, 10
--exclude LIST, 11
--fam <filename>, 7
--file <prefix>, 6
--from-bp <integer>, 11
--from-kb <integer>, 11
--from-mb <integer>, 11
--geno <integer>, 11
--impute <filename>, 8
--impute-count <integer>, 8
--impute-encoding, 8
--impute-fam <filename>, 8
--impute-gen-ext <extension>, 8
--impute-info-ext <extension>, 8
--impute-info-thresh <float>, 8
--impute-offset <integer>, 8
--impute-uncompressed, 8
--liability, 6
--logpath PATH, 14
--mach <filename>, 9
--mach-chrpos, 9
--mach-chunk-size <integer>, 9
--mach-count <integer>, 9
--mach-dose-ext <extension>, 9
--mach-info-ext <extension>, 9
--mach-min-rsquared <float>, 9
--mach-offset <integer>, 9
--mach-uncompressed, 9
--maf <float>, 11
--map <filename>, 6
--map3, 6
--max-maf <float>, 11
--mem STRING, 15
--mind <integer>, 11
--missing-phenotype CHAR, 10
--mphenos LIST, 10
--mv-path PATH, 14
--no-fid, 6
--no-parents, 6
--no-pheno, 6
--no-sex, 6
--ped <filename>, 6

--pheno <filename>, 10
--pheno-names LIST, 10
--remove LIST, 11
--res-path PATH, 14
--script-path PATH, 14
--sex, 10
--snps LIST, 11
--snps-per-job INTEGER, 14
--template FILENAME, 14
--tfam <filename>, 7
--tfile <prefix>, 7
--to-bp <integer>, 11
--to-kb <integer>, 11
--to-mb <integer>, 11
--tped <filename>, 7
--verbose, 11
--wall-time, 15
-h, --help, 5
-v, 5
Path to location of job's results, 14

P

Path to location of job's results
command line option, 14