**M+ manual**

**Purpose**

M+ is a computer program for producing core collections from multilocus genotype data. Two algorithms are available. The A\* algorithm (pronounced “A star”) is an artificial intelligence approach that guarantees that resulting core collections contain all alleles in the minimal number of accessions (Kim et al. 2007). We call this an “ideal core”. The A\* algorithm implemented in M+ uses OpenMP directives to parallelize the search across all processing threads available to the CPU. While it guarantees a minimal core with maximum diversity, only one such core is returned, and other, equally optimal cores may exist. The search space for A\* rapidly increases with the number of accessions, meaning that datasets of even modest size may be intractable.

A second algorithm, here called “M+” (based on Schoen and Brown 1993), uses a hill-climbing approach to produce optimized cores of a given size. While it is not guaranteed to find an ideal core, it often does and can be used with much larger data sets than A\*. The M+ algorithm implemented in the software M+ uses a master/slave MPI instruction set to parallelize the search across a defined number of processing elements. An advantage of MPI parallelization over OpenMP threading is that searches can be conducted on independent CPUs distributed across a network or housed within a supercomputer. However, because data structures are not shared among processing elements, MPI programs are memory intensive and may easily exceed system resources when using multi-core processors. We recommend that users have at least 4GB of RAM per processing element before tackling genomewide genotypic data sets with M+.

**Installation**

While M+ was primarily designed for high performance computing facilities, it may also be installed on desktop workstations. M+ is maintained as a C++ git repository located at: <https://github.com/NCGRP/MplusMPI>

Things will go easier if you have git installed on your machine. You will also need OpenMP and Open MPI prior to compilation. OpenMP libraries can be linked by most compilers, but Open MPI will usually have to be installed separately. To install:

git clone git://github.com/NCGRP/MplusMPI.git

cd MplusMPI

make

The repository includes program files (.cpp, .hpp), example files (.var, .dat), and a README document with abbreviated instructions. M+ was developed on Mac OS 10.6, and has also been compiled and run on desktops with Mac OS 10.7, 10.9, and Ubuntu Linux 14.04, a CRAY running SUSE Linux Enterprise 11 using GNU compilers, and two Dell clusters, one running Rocks 5.5 (Mamba) using GNU compilers, the other running Red Hat/CentOS 5.10 using Intel compilers.

**Input file format**

M+ requires two input files (.dat, .var) and accepts a third optional input file type (.ker). All files should be supplied as text with Unix line breaks. File formats follow the popular software MSTRAT (Gouesnard et al. 2001). Please refer to the example files provided with the git archive for the following discussion.

Genotypic (or categorical phenotypic) data are supplied in the .dat file. Each row contains data for one individual. Rows are split into columns on whitespace (e.g. space or tab characters). The first three columns are population (or accession) name, individual number, and individual name, respectively. Two individuals sampled from the same accession will be identical for column 1, but should differ for columns two and three. Genotypic data begins in column four. A diploid locus will require two consecutive columns, a triploid locus three columns, and so on. Alleles can be indicated using names or numbers, provided there is no whitespace within them. Missing data must be indicated with the code 9999.

The .var file defines the columns in the .dat file and indicates how M+ is to treat the various loci. The M+ .var file format is a simplification of the .var file format used for MSTRAT. The first three rows should not deviate from that seen in the example files:

code 0

individu 0

Sample 1 0 0 1 5

These rows provide no meaningful information to M+ but must be included. Starting at row four, corresponding to column four in the .dat file, locus names are specified in the first column. A diploid locus will occupy two rows, triploid three, and so on. Use the exact same name for all rows belonging to a single locus. Five additional columns are used to designate how the locus will be treated in the analysis. Numeric codes are used for this. To summarize:

Column 1: name of locus

Column 2: 1 (ignore this row), 2 (use this row)

Column 3: 1 (reference locus), 0 (not a reference locus)

Column 4: 1 (target locus), 0 (not a target locus)

Column 5: used by MSTRAT to define a weight, ignored by M+

Column 6: used by MSTRAT to define the number of classes for quantitative data, ignored by M+

For example, to define a diploid reference locus named “CA13new”:

CA13new 2 1 0 1 5

CA13new 2 1 0 1 5

To define a tetraploid target locus named “BvFL1”:

BvFL1 2 0 1 1 5

BvFL1 2 0 1 1 5

BvFL1 2 0 1 1 5

BvFL1 2 0 1 1 5

To define a triploid locus named “509” that should be ignored during analysis:

509 1 0 1 1 5

509 1 0 1 1 5

509 1 0 1 1 5

While the fourth column suggests that this is a target locus, M+ will ignore it because column 2 has precedence.

The .ker file provides a list of populations that must be retained in the core. It is a simple list of population names, one per row.

**Running M+**

Usage: m+ varfile datfile [-m mincoresize maxcoresize samplingfreq reps outputfile]

[-k kernelfile] [-a idealcorefile]

where,

varfile = path to MSTRAT .var file

datfile = path to MSTRAT .dat file

Options:

-m mincoresize maxcoresize samplingfreq reps outputfile = compute the optimal accessions for a given core size using the M+ search algorithm. Arguments are as follows:

mincoresize maxcoresize = integers specifying minimum and maximum core size. Usually mincoresize = 2, maxcoresize = total number of accessions

samplingfreq = e.g. integer value 5 will cause coresize=2 then coresize=7, then 12, and so on, to be sampled.

reps = number of replicate core sets to calculate for a particular core size

outputfile = path to output

-k kernelfile = use an MSTRAT .ker file to specify mandatory members of the core. The number of mandatory accessions must therefore be less than or equal to mincoresize. Option only applies to -m, and cannot be used with -a.

-a idealcorefile = compute the minimum set of accessions necessary to retain all variation, i.e. the "ideal" or "best" core, using the A\* search algorithm, write output to idealcorefile.

Notes: All input files must have Unix line breaks.

Examples: mpirun -np 2 ./m+ ./beet.var ./beet.dat -m 3 28 2 3 ./beetout.txt -k beet.ker -a beetideal.txt

aprun -n 2 ./m+ ./beet.var ./beet.dat -m 3 28 2 3 ./beetout.txt -k beet.ker -a beetideal.txt

mpirun -np 16 ./m+ ./orientalis.var ./orientalisIND.dat -m 2 50 1 1 orINDout.txt -a orINDidealout.txt

aprun -n 16 ./m+ ./orientalis.var ./orientalisIND.dat -m 2 50 1 1 orINDout.txt -a orINDidealout.txt

mpirun -np 20 ./m+ ./WheatSNP.var ./WheatSNP.dat -m 20 21 1 20 ./WheatSNPout.txt

aprun -n 20 ./m+ ./WheatSNP.var ./WheatSNP.dat -m 20 21 1 20 ./WheatSNPout.txt

**Output files**

The output produced when the –a option is invoked is a text file containing one (of possibly many) ideal core sets determined by applying the A\* algorithm. The –m option also produces a single text file as output. It is a tab delimited table of results from the entire search performed using the M+ algorithm. Columns are as follows:

1. core size = number of populations/accessions in the core

2. random reference diversity = genetic diversity retained at reference loci by selecting ‘*core* *size’* accessions at random, using a standardized metric of diversity

3. optimized reference diversity = genetic diversity retained at reference loci using the M+ algorithm, metric as above

4. random target diversity = genetic diversity retained at target loci, in the random set of accessions chosen in #2, metric as above

5. optimized target diversity = genetic diversity retained at target loci, in the set of accessions chosen in #3, metric as above

6. alt random reference diversity = count of alleles retained in #2

7. alt optimized reference diversity = count of alleles retained in #3

8. alt random target diversity = count of alleles retained at target loci in core set from #2

9. alt optimized target diversity = count of alleles retained at target loci in core set from #3

10. core members = populations/accessions included in the core

**Citations**

Gouesnard B, Bataillon TM, Decoux G, Rozale C, Schoen DJ, David

JL (2001) MSTRAT: an algorithm for building germ plasm core

collections by maximizing allelic or phenotypic richness.

J Hered 92:93–94.

Kim K–W, Chung H–K, Cho G–T, Ma K–H, Chandrabalan D, Gwag

J–G, Kim T–S, Cho E–G, Park Y–J (2007) PowerCore: a

program applying the advanced M strategy with a heuristic

search for establishing core sets. Bioinformatics 23:2155–2162

Schoen, DJ, Brown AH (1993) Conservation of allelic richness in wild crop relatives is aided by assessment of genetic markers. PNAS 90:10623-10627.