MSG: Multiplexed Shotgun Genotyping

A User's Manual (version: 0.1)
22 Sept. 2015

David Stern, Molly Schumer, Greg Pinero, Peter Andolfatto

MSG is a software package that performs the computational tasks associated with Multiplexed Shotgun Genotyping (Andolfatto et al., 2011). MSG requires (1) a fastq format file containing multiplexed bar-coded sequence reads (or pre-parsed fastq files), (2) a barcode file that associates each barcode with an individual identifier, (3) a primary genome reference sequence used to establish genome coordinates, (4) genomic reference sequence data for the second species used in the experiment and (5) sequence data for the specific strains of the two reference genomes used in your experiment. The quality values in the fastq file must be in Sanger format. MSG will calculate the posterior probability that an individual is homozygous or heterozygous for parental ancestry at a given genomic position. Posterior probabilities of ancestry ("genotypes") are reported at each nucleotide position for which at least one individual carries an informative allele. A secondary function of MSG is to provide a one-step pipeline for "updating" parental genomes with new sequence reads. The genotyping and updating pipelines are run separately. Since the initial release of MSG in 2011, we have added several new options to MSG . This manual provides a guide to installing and running MSG, and suggestions for how to interpret the results and use the output files in downstream applications, such as quantitative trait locus (QTL) mapping using R-qtl (Broman, Wu, Sen, & Churchill, 2003).

MSG is computationally intensive and we recommend running the code only on a high-performance computer cluster. Although MSG can be run on a desktop, a run with only 50-100 individuals can take 24 hours or longer.

How to set up MSG on your own cluster.

The current version of MSG is maintained at https://github.com/JaneliaSciComp/msg.
Either download a zipped directory of msg from github and unzip

```
$ unzip msg-master.zip
```

or clone from github

```
$ git clone git://github.com/JaneliaSciComp/msg.git
```

Navigate to the msg folder and build the C files

```
$ cd msg_version/
```

```
$ make
```

MSG requires multiple specific dependencies and some of these dependencies are not the current versions of the software. Since we have not maintained MSG to run with current versions of these dependencies, it is critically important that MSG is installed with the proper versions of these dependencies. We therefore provide all required dependencies with the package in the folder `msg/dependencies`. See the Getting_Started.txt file for instructions about installing these dependencies.

Running MSG on Amazon Web Services

If you do not have access to a compute cluster, you can run msg on Amazon Web Services. Please see documentation at
https://github.com/JaneliaSciComp/msg/blob/master/instructions/MSG_Running_on_the_Cloud.doc
Please note that you are responsible for all charges accrued on Amazon Web Services.

First, run the test data!

In the folder `msg/example_MSG_toy`, we provide a very small dataset of real data that you can download from github and run to ensure that everything is working properly. Take all the files within this folder and place them within a new folder on your cluster. Within the same folder, link to the folder where you have installed MSG.

```
$ ln -s your_installed_location/msg
```

First, update the parental genomes by typing

```
$ perl msg/msgUpdateParentals.pl
```

When this run is done, you will see two new files

```
parent1.fa.msg.updated.fasta
parent2.fa.msg.updated.fasta
```

These are the updated parental genomes used in the next step.
To run MSG, now type

```
$ perl msg/msgCluster.pl
```

MSG will produce a bunch of output and your jobs should be sent to the cluster. If you are not running a PBS system you will need to edit the cluster submission options before sending the jobs to the cluster; read below for more details. Since this is a very small data set, the entire analysis should take no more than a few minutes. It should produce a batch of folders and files

with results that match the output that you can find in `msg/example_MSG_toy/example_results`.

If this toy example runs properly, then you may want to test a larger dataset. We provide information and links to a larger data set in `msg/example`.

Updating parental genomes

As you probably inferred from running the toy data, often the first step in MSG is to generate parental genomes that have been updated to reflect the specific parental strains under consideration. If you happen to be in the enviable position of having complete genome sequences that were generated from the relevant strains, then you can skip this step. In fact, many genome sequences are now derived by mapping reads to a pre-existing genome sequence. This is all that the `msgUpdateParentals` script does, by combining all of the commands for mapping reads to a reference genome and for generating a new consensus sequence. To perform this updating, you provide a file containing the reference genome sequence for either or both parents, the reads for updating in fastq format for either or both parents, and an `update.cfg` file. If you are updating two parental genomes, then you must provide two separate files for the reference genome sequences. If the two reference genomes are in fact identical, simply name them differently. `msgUpdateParentals` will crash if you try to update a single genome with reads from two parentals. We find that ½ of a lane of Illumina single-end reads (~100 million 100bp reads) is usually sufficient to generate excellent updating of *Drosophila* genomes. These reads can be provided in gzipped format, and we recommend this.

update.cfg file

You can specify multiple parameters in `update.cfg` to influence how genome updating is performed.

cluster (0/1): If running on a cluster, set to 1 (default), if not, set to 0. Your cluster may have different parameters than our cluster, so queue and threads may not be relevant. Talk with your system administrator about defining system-specific commands (see more below under *Custom qsub options*).

debug (0/1): 1 for verbose output and to retain all intermediate files. This setting dramatically increases the storage space required and is not recommended unless you are trying to debug the code. Default = 0.

parent1 & *parent2*: Reference genome filenames for parent1 and parent2. These genomes can be identical, but must have different names. These genomes can be provided in gzipped format.

parent1_reads & *parent2_reads*: Fastq files containing sequencing reads for genome updating. These files can be provided in gzipped format.

bwaindex1 & *bwaindex2* (*bwtsw*/*is*): Algorithm for constructing bwt index. *bwtsw* (default) is appropriate for most cases, *is* should be used with genomes smaller than ~ 10 Mbp.

bwa_alg (*aln*/*bwasw*): *aln* (default) should be faster and more suitable for short reads, *bwasw* should be more accurate. This parameter is ignored if *use_stampy* is set to 1.

bwa_threads (integer): Number of threads used by BWA when threads is set > 1. (default = 8)

use_stampy (0/1): Use stampy (1) instead of BWA for read mapping. stampy is considerably slower than bwa, but allows mapping of more divergent reads. Default = 0.

stampy_premap_w_bwa (0/1): Pre-map reads with BWA prior to running stampy. This is recommended. This parameter is ignored if *use_stampy* = 0. Default = 1.

stampy_pseudo_threads (0/1): If *use_stampy* = 1 and *cluster* = 1, then you can set the number of pseudo-threads that stampy uses during computation. This can dramatically reduce computation time when using stampy. Default = 0.

update_minQV (Phred score: ≥0): Minimum Phred-scaled consensus quality required for accepting updating base in new consensus sequence. Setting >0 will mask regions that do not reach this criterion.

min_coverage (integer): Minimum number of reads needed at a base to accept for updating. Default = 2.

max_coverage_stds (positive real): If defined, the maximum number of reads allowed at a site, in units of standard deviations, to accept for estimating consensus sequence. Default = no maximum.

max_coverage_exceeded_state (N/D): Base pair state to use if maximum coverage is exceeded. Default = “N”, set to D to use existing base. (Apologies for use of D, meaning “default”, to default to the existing base.)

quality_trim_reads_thresh (Phred score: ≥0): MSG can perform quality trimming of your sequencing reads prior to mapping. If you have not already quality trimmed your reads, then we STRONGLY recommend that you allow MSG to perform this quality trimming. Phred score of 20 is recommended. Default = 0.

quality_trim_reads_consec (≥0): Trim reads to high quality base pairs of at least this length. Reads shorter than 30 bp aren't terribly useful, so best to exclude them. Default = 30.

parent1_stampy_substitution_rate & *parent2_stampy_substitution_rate* (>0-<1): When using stampy for read mapping, set expected divergence between reads and reference. This is recommended when expected divergence is > 3%. (Default = 0.001.)

parent1_mapq_filter & *parent2_mapq_filter* (≥0): Mapping reads with bwa generates a quality score for each mapped read. In principle, this can be used to filter out reads that have mapped poorly. Set minimum quality score for acceptance of mapped read. (Default = 0)

Making and using disambiguated parental genomes

Polymorphism that is shared between the parental strains can introduce errors that reduce the accuracy of hmm inference in MSG. MSG uses only sites in the parental genomes that are called as either A, T, C, or G. Ambiguous bases are ignored (although they could be incorporated into the hmm in the future). Thus, if a parental genome is incorrectly updated as monomorphic at a particular site when it is actually polymorphic, then this can introduce error at this site. When performing crosses between species, one simple way that we have discovered to reduce this error is to compare multiple genomes within a species and to retain only positions that are monomorphic across all genomes. This step removes many polymorphic sites that are segregating in the species and that may be segregating in your cross. This step necessarily removes many sites that are considered by the hmm, but the reduced error associated with the remaining sites substantially improves accuracy of the hmm.

We provide a simple script for removing polymorphic sites between two genomes in the `msg/tools` folder, `DisambiquateGenome.py`. This step can be performed iteratively to remove all polymorphism between any number of genomes.

Setting up files for running MSG in the default mode

Fastq file

MSG accepts a fastq file and will parse the file by barcode according to the individual identifiers provided in a separate barcode file. MSG will run on uncompressed or gzipped fastq files. We provide two parser options: (1) a slower, but more careful option that will search for reverse complemented adaptors and other problems, (2) a faster parser that looks only for perfect matches to the barcode and filters out a user defined sequence (normally a reverse complemented adaptor). Generally, we use parser 1 if we have short barcodes (~6bp) and parser 2 with longer or more complex barcodes. Note, the first step of parser 2 is to check if the file is compressed and if it is, then it uncompresses the file, because the search algorithm runs much more quickly on uncompressed files. So, if you have an uncompressed file, you may just want to leave it, despite the fact that it is likely to be very large. After parsing, the uncompressed

file (`temp.fq`) is emptied, but the empty file remains in the folder. This empty file is left as a signal to MSG that the file has already been parsed.

If your data consist of multiple fastq files that have not been parsed, then you should concatenate them into a single file. Do not directly concatenate gzipped files, despite the fact that some websites recommend this. Directly concatenating these files inserts an extra text line between the files that will break MSG. Instead, gunzip each file, concatenate them, and then, if you want, gzip them.

Setting up files for running MSG with pre-parsed fastq files

If you have already parsed your files using a different program, then you can use these files directly in msg but they will require some reformatting. In this case, you need to create a dummy file to indicate to MSG the name of the sequences, and a folder that includes the matching name appended with "_parsed". For example, if the dummy file is called "sample.fq", then the folder is called "sample.fq_parsed". The dummy reads file can be empty. For example, if you want your dummy reads file to have the name sample.fastq, you can make an empty reads file and a reads directory as follows:

```
$ touch sample.fastq

$ mkdir sample.fastq_parsed
```

Then, copy all of the parsed files to this folder and compress them with gzip, if they are not already compressed.

```
$ cp parsed_file* sample.fastq_parsed/
$ gzip sample.fastq_parsed/*
```

If you want to move the files instead of copying them, you can instead type:

```
$ mv parsed_file* sample.fastq_parsed/
```

It is also possible that on your system you can generate a symbolic link to these files, which may be stored elsewhere, as long as you system allows rapid access to these files. This has the advantage of not duplicating large files. For example, type:

```
$ ln -s storage_location/parsed_file* sample.fastq_parsed/
```

For MSG to run properly on these files the parsed file names need to follow a particular format relative to the barcode file. (Barcode file format is discussed in the next section.) If the reads

were parsed outside of MSG, it is likely that you need to rename them before running the MSG pipeline.

If a line in your barcode file reads

```
ATGACA  sample11
```

then MSG will look for the file containing the sequence reads with the name indivsample11_ATGACA.gz. Variations on this file name (including indivsample11_ATGACA.fq.gz) will cause the program to crash. To rename your files, use the mv command in unix. For example:

```
$ mv sample11_index1.fq.gz indivsample11_ATGACA.gz
```

Setting up the other files required by MSG

Barcode file

The barcode file contains 2-4 columns, separated by tabs, with the barcode in the first column and the specimen name in the second. Optional third and fourth columns are plate ID and sex. The file does NOT contain a header! For example:

```
AATCTA     COACB11    1     Male
GTGAGA     COACB12    1     Female
CGACTA     COACB13    2     Female
ACTAGC     COACB14    3     Female
GTGACT     COACB15    4     Male
ACTGAC     COACB16    4     Male
```

The file must use Unix encoding and LF line break style. The simplest way to do this is to generate the file in a text editor, select all, copy, then, type

```
$ cat > barcodes
```

paste the list

then press Control Z (on a Mac). Check the file with

```
$ cat barcodes
```

If you import the file in another way, check it with the `cat` command. If you see lots of strange symbols (most commonly `^M`), then you have used the wrong formatting and MSG will crash.

You can change formats easily using TextWrangler or similar text editors. Often the problem can be fixed by using the conversion command mac2unix or dos2unix. Avoid the Mac's TextEdit program. Improperly encoded barcode files are the most common causes of MSG failure. Fortunately, they are fixed easily!

WARNING: The barcode file is sensitive to spaces and certain other characters. Including these characters will cause MSG to fail. For example, formats such as:

```
AATCTA    COAC.B11
AATCTA    COAC/B11
AATCTA    COAC B11
AATCTA    COAC,B11
AATCTA    COAC(B11)
```

shout **NOT** be used. However,

```
AATCTA    COAC_B11
```

and

```
AATCTA    COAC-B11
```

will behave properly.

Illumina index files

Normal MSG runs don't need this option. When the illumina index system is used. There will be three input files:

1. A normal fastq file with all of the reads
2. A fastq file with the same reference ids as the reads file but with the index as the sequence
3. A small indices file that contains a label for each index sequence.

1 would be used for the *reads* parameter in msg.cfg. The *index_file* parameter in `msg.cfg` should point to 2, and the *index_barcodes* parameter should point to 3.
When not using the Illumina index files leave *index_file* and *index_barcodes* commented out in your `msg.cfg` file.

msg.cfg file

If you specify everything correctly in the `msg.cfg`, then things should go swimmingly. Incorrect specification of file names or parameters in the `msg.cfg` file is the second most common source of MSG failure!

addl_qsub_option_for_exclusive_node: Add qsub options for customized cluster configuration for requesting an exclusive node. e.g. addl_qsub_option_for_exclusive_node=-l mem96=true,excl=true

custom_qsub_options_for_all_cmds: Add qsub options for customized cluster configuration. e.g. custom_qsub_options_for_all_cmds=-q customer.q

addl_qsub_option_for_pe: Add qsub options for parallel computation in customized cluster configuration. e.g. addl_qsub_option_for_pe=-pe batch

barcodes: This is the name of the barcodes file.

bwaindex1 + *bwaindex2* (*bwtsw*/*is*): Algorithm for constructing bwt index. *bwtsw* (default) is appropriate for most cases, *is* should be used with genomes smaller than ~ 10 Mbp.
The appropriate algorithm depends on genome size, details can be found at http://bio-bwa.sourceforge.net/bwa.shtml.

bwa_alg (*aln*/*bwasw*): Algorithm for alignment of reads with bwa. aln is more efficient with the number of reads typically used in MSG, but bwasw may be more accurate.

bwa_threads (integer): Number of threads to use for bwa.

chroms (specific contigs or *all*): A list of chromosome/contig names, separated by commas, in the parent1 genome that you wish to include in the hmm analysis. Note that many "assembled" genomes contain many—sometimes thousands of—unassembled contigs. MSG will run for noticeably longer (sometimes many times longer) if you specify "all" and the genome contains thousands of contigs. If you are paying for computer time, you will notice this in your bill! Typically, we provide the "complete" "assembled" genome in *parent1* to allow mapping to all possible contigs, but we calculate the hmm results only for large chromosomes of interest (for *Drosophila*, this is 2,3,4,X). If you have many chromosomes, you may also want to run batches of several chromosomes at a time.

chroms2plot: A list of chromosome/contig names, separated by commas, for which you want to produce genotype plots. Warning: plotting many contigs will slow the pipeline and possibly prevent it from finishing. Second warning: the names of these chromosomes/contigs must match exactly the names in the fasta file.

cluster: If running on a cluster, set to 1. Your cluster may have different parameters than our cluster, so queue and threads may not be relevant. Talk with the your system administrator about customizing the `msg.cfg` file for your cluster. We provide parameters that allow you to send cluster-specific qsub commands.

debug: Leave this set to 0 unless you are actively developing the MSG code-base (or if you are an expert user with plenty of disk space).

deltapar1 & *deltapar2* (0-1): The expected error rate for ancestry tracts derived from each parent. Error in MSG can be generated by shared polymorphism at a site that is represented as fixed between the two parental genomes, mismapping, contamination, and sequencing error. This appropriate value for this parameter is difficult to know definitively but we suggest starting with values 0.01-0.05 and inferring from the data. The MSG pipeline outputs a file called error_gama.pdf in the hmm_fit_images folder which shows the distribution of inferred error in the samples (inferred from homozygous blocks), and this can be used to update the error parameter for subsequent runs.

email_host: MSG can send an e-mail when the run is completed. If you want this, specify <email_host> and <notify_emails>. An email is not sent if the run crashes. Use email_host to specify the URL that receives the e-mails. Ask your systems administrator if you are unsure. e.g. email_host=10.11.5.81

gff_thresh_conf : MSG will output a GFF formatted file (for importing into Geneious, http://www.geneious.com/) for each individual for each chromosome in the hmm_fit folder. This value specifies the confidence threshold to use when selecting breakpoints to be included in the Geneious files. The default is 0.95 (95%) which means it selects breakpoints at 5% and 95% confidence.

index_file: See Illumina index files section above.

index_barcodes: See Illumina index files section above.

indiv_mapq_filter (≥0): Specify a minimum mapping quality score to retain alignments. If mapping quality is less than this number (recommended setting: 20), the alignment will be removed. Set to 0 to skip.

indiv_stampy_substitution_rate (0-1): This parameter is used when use_stampy is set to 1 and gives the expected proportion of sites that differ from the reference sequence. The default value for stampy is 0.001 (0.1% of sites differ). Commenting out this option will result in the default value being used. NOTE: if the expected substitution rate is ~0.03 this parameter should be specified as it will prevent pre-mapping with bwa and possible segmentation faults.

linker_system (Tn5-IonTorrent|Dros_SR_vII): This is the linker system enzyme used. Normally use Dros_SR_vII. Set *re_cutter* to null when *linker_system* = Tn5-IonTorrent.

max_mapped_reads: Uses only the first <max_mapped_reads> of the sam files for further analysis. This will reduce computation time. We find that the resolution of breakpoints tends to asymptote after a certain number of reads, so adding more reads doesn't improve results. In

addition, capping the reads reduces the chances that one_site_per_contig ends up undersampling the genome (See explanation below in one_site_per_contig). Finally, individuals with very high read counts often produce small, clearly incorrect, switches of the hmm probabilities. We do not know the cause of this in every case and it could be due to multiple causes (small regions of genomes misassembled, small repetitive DNA regions, etc.), but this problem is ameliorated by reducing the number of reads used. NOTE: This option truncates the same files, so to try different # reads, you must delete the *sam_files folder.

new_parser (0/1): Set to 0 to use the older, slower parser. Set to 1 to use the newer, faster parser. The older parser performs a bit more quality control than the newer parser.

new_parser_offset (integer): Remove this number of bases following the barcode sequence. This is useful if you have used a library prep method that includes an adaptor sequence 3' to the barcode sequence.

new_parser_filter_out (a DNA sequence): Search for this sequence in each read and hard trim starting at this sequence. For example, use the reverse complement of an adaptor sequence to trim adaptor sequences.

notify_emails: Specify recipients of e-mail notification when MSG finishes successfully. Separate multiple e-mails with semicolon (;). e.g. notify_emails=test@example.com

one_site_per_contig (0/1): This is the number of SNPs to use per contiguous sequence of mapped reads. This requires a little explanation. When reads are mapped to the genome, sometimes they will overlap and result in a sequence longer than the read length that has been mapped to the reference. As sequencers have improved, this has become more common and it is most extreme when shotgun sequence data are used with high coverage. In principle, for example, complete read coverage would result in a single contiguous sequence per chromosome. In such a case, setting this option to 1 would result in only one SNP being used from the entire chromosome! So, one would think that using all SNPs (setting *one_site_per_contig* to 0) would be beneficial, because we would use all the data. In practice, however, we have found that accuracy is usually much higher when this parameter is set to 1. We believe that the reason is that usually with MSG we are dealing with relatively short contiguous regions (on the order of the read length). So, when reads map correctly, there is little gain from using all SNPs in a single contiguous region (which is short). However, when reads map incorrectly, multiple SNPs from this region will provide incorrect ancestry information. That is, errors are correlated in space along the chromosome. Thus, using all sites usually increases the local rate of incorrect SNP calling, reducing the accuracy of the hmm. If you have higher coverage data (e.g. >5X shotgun sequencing) that is likely to collapse many reads into one you can cap the number of reads used (see above).

pepthresh (0-1): A threshold to denote the coverage cutoff for including sites in the hmm_fits_est.csv file (which is explained below). We normally use 0.5. If this option is omitted

from the *msg.cfg* file, then several individual summary files normally found in the hmm_data folder will not be generated. This can save space if you are not planning on using this output.

parent1 + *parent2*: The two parental genomes in fasta format. All MSG out is with reference to the chromosomes and coordinates of the chromosomes/contigs in the *parent1* file. The chromosome structure of *parent2* is irrelevant. That is, you can take one well-assembled genome for parent 1 and one poorly assembled genome for parent 2 and MSG will calculate ancestry with respect to the *parent1* chromosomes. These genomes can also be provided in gzipped format. The chromosome identifications provided in the parameter *chroms* must match the *parent1* genome. If they are not, msg will run to completion, but it will not produce any ancestry estimates of the requested chromosomes.

pnathresh (0-1): The maximum proportion of missing data values for a single individual sample to include this individual in the summary plots.

priors: A comma-separated list of the expected probability of ancestry for each possible genotype (homozygous par1, heterozygous, homozygous par2), which must add to 1. The first probability should match the probability for being homozygous for the parent specified as parent 1 by the fasta input files. For an F2 intercross the probabilities are 0.25,0.5,0.25 for an F2 backcross the probabilities are 0.5,0.5,0 or 0,0.5,0.5. Priors have the largest effect in regions of low information, but incorrect priors can generate major problems genome-wide.

quality_trim_reads_thresh (≥0): Phred base quality for trimming. To skip set to 0.

quality_trim_reads_consec (≥0): Number of consecutive high quality base pairs required to retain a read. We recommend discarding reads shorter than 30 base pairs after trimming.

reads: This is the fastq file containing all of the reads. If you created a dummy reads file for pre-parsed data, this is the name of your dummy reads file (e.g. sample.fastq).

recRate (≥0): The expected number of recombination events per genome per meiosis. If *recRate* = 0, msg assumes one recombination event per genome. (For studies of *Drosophila melanogaster*, we set this at 3, which is equivalent to one recombination event per large chromosome).

re_cutter (MseI|NdeI|Hpy188I|HphI|MboII|XcmI|Hpy188III|AhdI|HpyAV|null): This is the restriction enzyme used in the original MSG library preparation method. Normally use MseI. If data has been pre-parsed this parameter is not used (leave as default MseI).

rfac: This is an arbitrary scalar of RecRate, default = 1. If the HMM is too jumpy in your opinion (too many recombination breakpoints), you can set this <1. You can also dial this >1 to increase sensitivity (too few recombination breakpoints).

sexchroms: A list of sex chromosomes/contigs, which are assumed to be hemizygous in the heterogametic sex.

stampy_premap_w_bwa (0/1): If *use_stampy* is set to 1, setting this option to 1 and premapping with bwa will significantly improve the speed of the pipeline.

stampy_pseudo_threads (≥0): If *use_stampy* is set to 1 and the cluster option is set to 1, mapping with stampy can be parallelized and the number input here will specify the number of parts to split the stampy job into.

theta (0-1): The amount of dependence between reads. If each read is expected to be completely independent (no PCR duplication) then theta can be set to 1. If reads are not completely independent MSG is more accurate when this value is <1. See doi:10.1101/gr.9.9.868 and doi:10.1101/gr.078212.108 for further details.

threads: Specify threads to use on cluster configuration. e.g. threads=8

use_stampy (0/1): Set to 1 to use stampy for mapping instead of bwa. Stampy is able to map reads with higher divergence than bwa (1% or greater) but is *significantly* slower than bwa. If the two parental species are highly divergent, we recommend using stampy but pre-mapping with bwa. To use this option, *use_stampy* must be set to 1.

Parental genomes

As stated above, you need at least one genome that contains large contigs and preferably chromosome-length contigs. This should be parent1. Parent 2 does not need to be well assembled. As an example, we often use the published well-assembled genome of *Drosophila simulans* as parent1 and the poorly-assembled genome of *Drosophila sechellia* as parent2. By convention, we considered parent2 to be the backcross parent (and set the priors to 0,0.5,0.5), but if you perform a backcross to parent1, simply set the priors to 0.5,0.5,0.

MSG will use only the regions of the parental genome that are denoted in capital letters. Lower case letters are often used to indicate regions of low quality and these are ignored by MSG. Therefore, it is important to check that your parental genomes are provided as uppercase sequences.

Troubleshooting

Common causes of complete failure

1 - Barcode file is not in UNIX style text with LF returns.

2 - All individual names in barcode file include spaces or problematic characters (e.g. \ / . , ( )). Use _ in place of “space” in individual names
3 - Formatting problems in the input parental genomes (extra spaces, not in fasta format)
4 - If you input pre-parsed data, lack of an exact match between expected fastq file names based on the barcodes file and the actual file name

Common causes of partial failure

1 - barcode file includes barcodes not in the experiment
2 - Some individual names in barcode file include spaces or problematic characters (e.g. \ / . , ( )). Use _ in place of “space” in individual names
3 - fastq file corrupted or not, actually, a fastq file. Be careful when manually trimming fastq files that you specify to save the files in fastq format (often, default is fasta).

Common causes of long run times

1 - Parameter *chroms* set to “all” and genome contains many small contigs. Even many “fully assembled” genomes contain additional unassembled contigs. Because the hmm analysis runs on a chromosome by chromosome basis, analyzing each chromosome involves considerable I/O overhead and analyzing thousands of such chromosomes can result in dramatically reduced performance. Since the hmm on these small contigs is usually uninformative anyway, we strongly recommend that parameter *chroms* defines only the chromosomes/contigs that you want to analyze.

2 - Some individuals contain many more reads than other individuals. We provide simulation software to determine the optimal number of reads for MSG. We recommend capping the number of reads used using the parameter *max_mapped_reads*.

3 - Many individuals are included in the run and summary plotting scripts cannot complete in msgRun3. Increasing the resources available to msgRun3 can help with this problem.

Using MSG for population analysis

MSG can be used to analyze non-standard hybrids such as natural hybrids and introgressed lines, but caution is required! In both cases, ancestry tracts are smaller than expected with an early generation cross and if there is insufficient data, incorrect parameters, or very small ancestry tracts then the results can be highly inaccurate. We recommend simulating data to determine the appropriate parameters and number of reads required for your particular application. An example of this type of approach can be found at http://dx.doi.org/10.7554/*eLife*.02535. A simulator tool which we have developed for this purpose can be found at https://github.com/melop/simMSG.

Interpreting the results from your first run

In our experience, optimizing the parameters for MSG is an iterative process. We therefore recommend first analyzing a subset of your data to ensure you set up everything correctly and to rapidly (and more cheaply) identify suitable parameters.

After examining the output from the first run, you may wish to alter some of the parameters (such as the error rate, recombination rate factor) and run MSG again. In this case, you can re-run MSG from each of multiple points by simply deleting the folders produced subsequently and running again. MSG checks to see which folders are present, and works from the last one available. Folders are produced in the following order:

*Filename.fastq*_parsed
*Filename.fastq*_sam_files
hmm_data
hmm_fit
hmm_fit_images

Finally the ancestry files are produced.

So, for example, if you want to rerun MSG with different parameters for the HMM (this is the most common scenario), you would delete the ancestry files and the folders hmm_fit and hmm_fit_images. DO NOT delete hmm_data in this case. The results in hmm_data are computationally intensive to produce, so you will save a lot of time and money just rerunning the "fit".

How to know when a run has completed

If the run has completed with no fatal errors at any step, individual barcodes files will have been removed and all msgRun error and output files will have been moved to folders called msgError.jobid and msgOut.job id. In addition, msg will have made the following three folders: hmm_data, hmm_fit, and hmm_fit_images as well as a series of ancestry*.tsv files with individual ancestry information (see below).

How to think about the results

The output of the MSG pipeline that is typically used in downstream analyses is the ancestry*tsv files (ancestry-probs-par1.tsv, ancestry-probs-par1par2.tsv, ancestry-probs-par2.tsv). These files contain individual genotype calls for the posterior probability of being homozygous parent 1, heterozygous, and homozygous parent 2 respectively. These “genotype” calls are combined

information from ancestry informative SNPs detected in each individual and interpolation of genotypes over missing data during the hmm fitting step of MSG. Because the hmm is influenced by the parameters input by the user, some genotype calls will change when parameters are varied. The hmm_fit and hmm_fit_images folders also have plots that are crucial to evaluate to determine if your parameters need to be modified (see next section).

During the hmm fit step of MSG, the program combines information ancestry informative markers identified in the sample (cataloged in the individual .hmmdata files inside the hmm_data folder). The program also takes into account the priors for ancestry you input in the cfg file; in regions of the genome with low information the hmm will rely more heavily on the ancestry priors and will attempt to match the ancestry priors on average. In addition, the recombination rate specified in the cfg file influences the number of recombination events introduced and how much ancestry at neighboring SNPs is relied upon in inferring ancestry at the current site. If the recombination rate is too high, the program may introduce spurious breaks resulting in incorrect genotype calls. Similarly, if the rfac parameter is too high the hmm will be "jumpy," meaning that it has a higher likelihood of switching from the current genotype to a new genotype. rfac values that are too high or too low can reduce accuracy for this reason.

If parameters are incorrect, how can this be determined and corrected? Sometimes it can be difficult to tell from the output if incorrect parameters were used. Some basic biological reality checks can be helpful. For example, how many recombination events do you expect to detect in your cross design? If you have five breakpoints a chromosome and an F2 cross design, recombination is likely too high. What genotypes do you expect to detect in your cross design? If you have backcrossed to a particular parent and you have homozygous genotypes for the other parent this can suggest a problem with the priors (or the set of ancestry informative markers defined by the input genome). Other approaches, such as looking at ancestry proportions (or hardy-weinberg equilibrium in some cross types) are discussed in more detail below. In addition to those discussed above, error rate is another parameter to consider tuning (see below on how to determine error rate from your data) if there are too many breakpoints or unexpected genotypes. If the error rate in the cfg file is set too low, MSG may interpret errors as switches in ancestry.

Finally, you can use simulations to get an idea of how sensitive your data is likely to be to variation in certain parameters. In our experience, some parameters (like error rate) have a more minor impact on accuracy, while others, like rfac and incorrect priors have a much more significant impact on accuracy. A simulator tool which we have developed for this purpose can be found at https://github.com/melop/simMSG.

What to do after a run finishes

Look at the data

First, look at the figures in the hmm_fit_images folder.

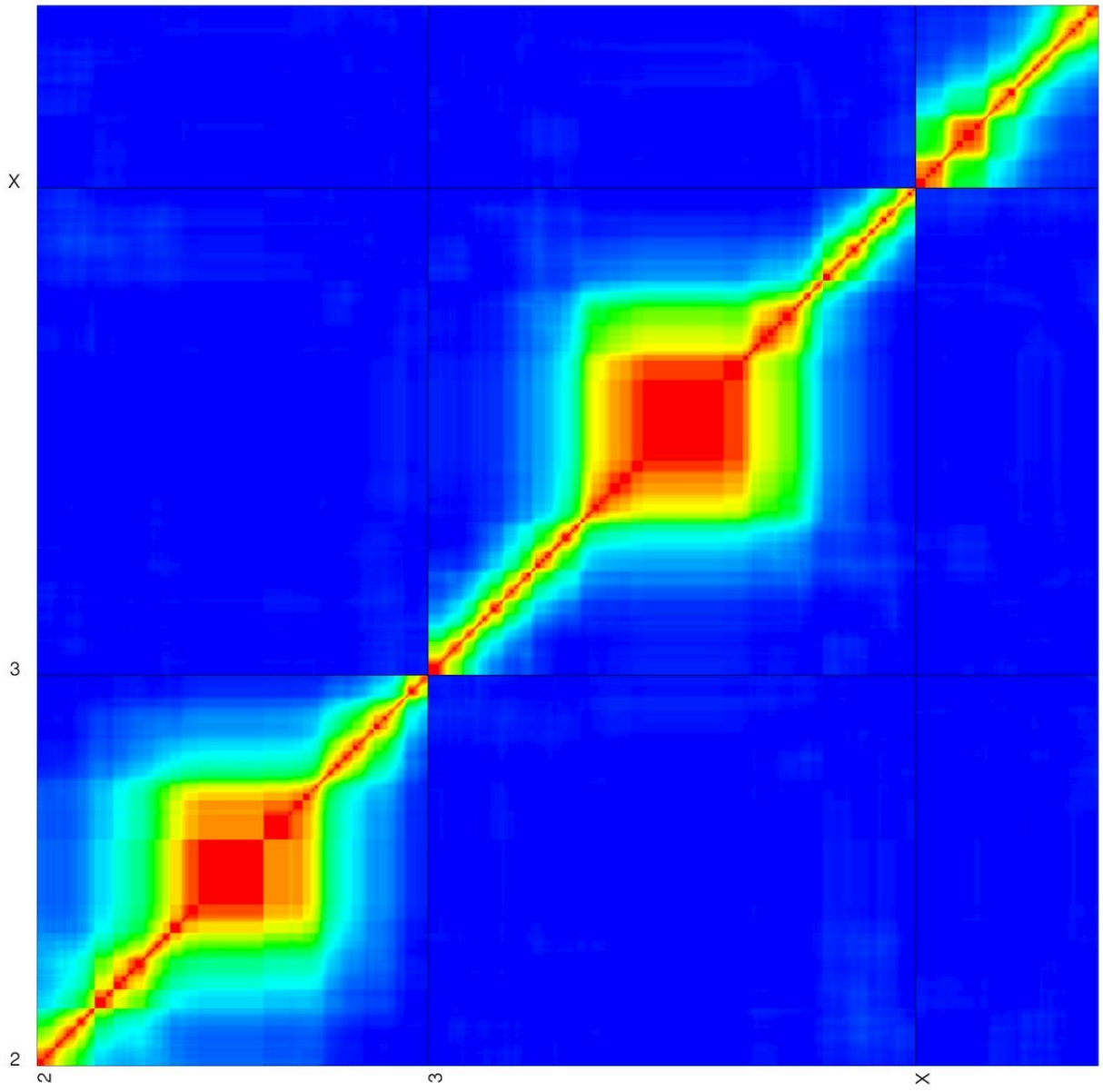
lod-matrix.bmp

We often look first at the lod-matrix.bmp. This illustrates linkage (as LOD scores) between pairs of markers on all chromosomes that pass a threshold of missing data (set in the msg.cfg file as pnathresh). Usually, not all markers are shown, since the nXn matrix of all markers would take too long to calculate and plot. Instead, this is a plot of thinned data (from file ancestry-probs-par2-*-*-*.thinned_plot.rda). The three parameters masked with the * wildcard are the thinning parameters that can be set, optionally, in the msg.cfg file.
For example,
ancestry-probs-par2-1.000000-0.010000-0.500000.thinned_plot.rda
corresponds to
thinfac=1
difffac=.01
pnathresh=0.5

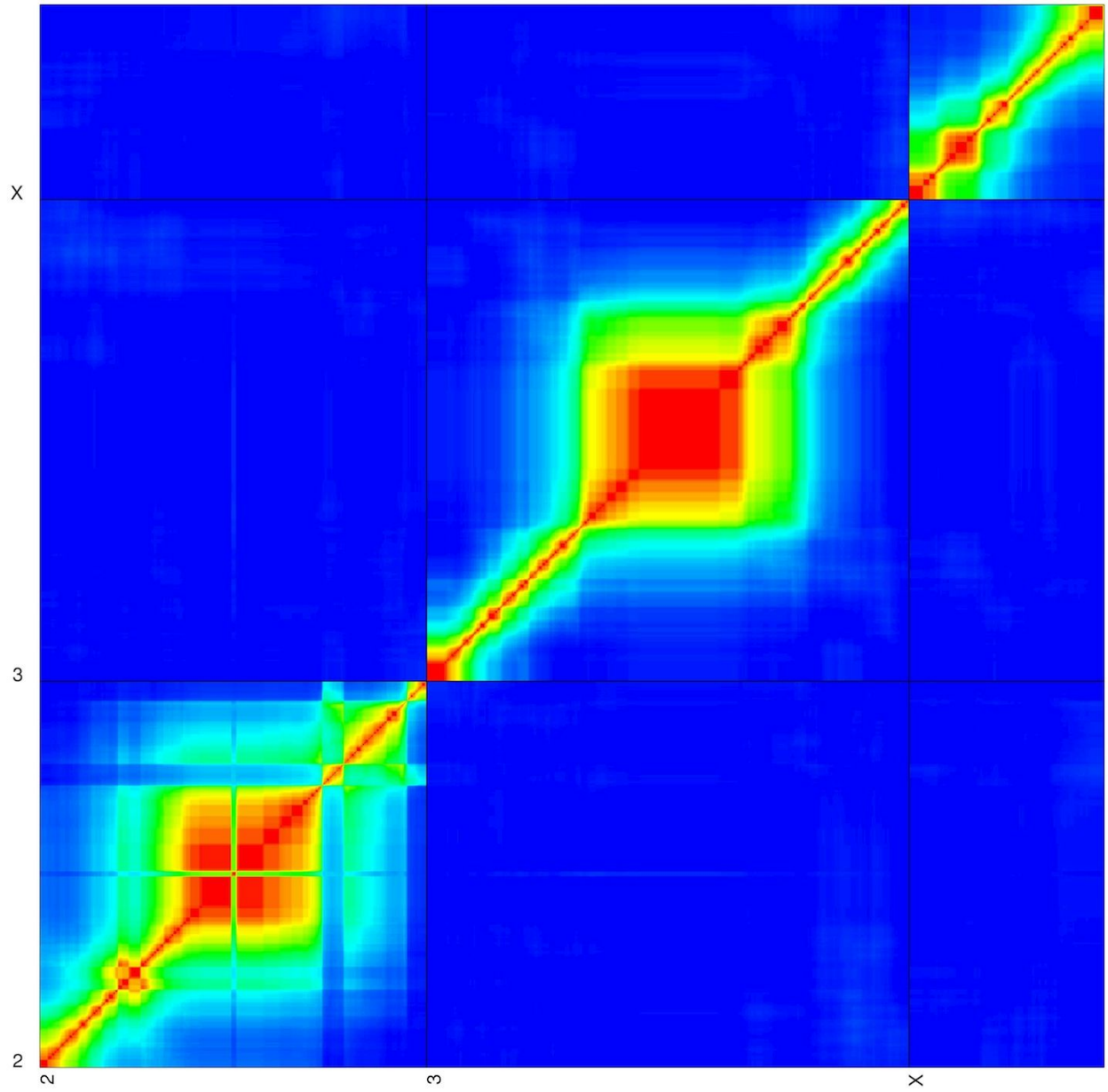
The default parameters are
thinfac=1
difffac=.01
pnathresh=0.03

Note that msg calculates linkage from only the par2 ancestry results. This assumes, essentially, that the data come from a backcross.

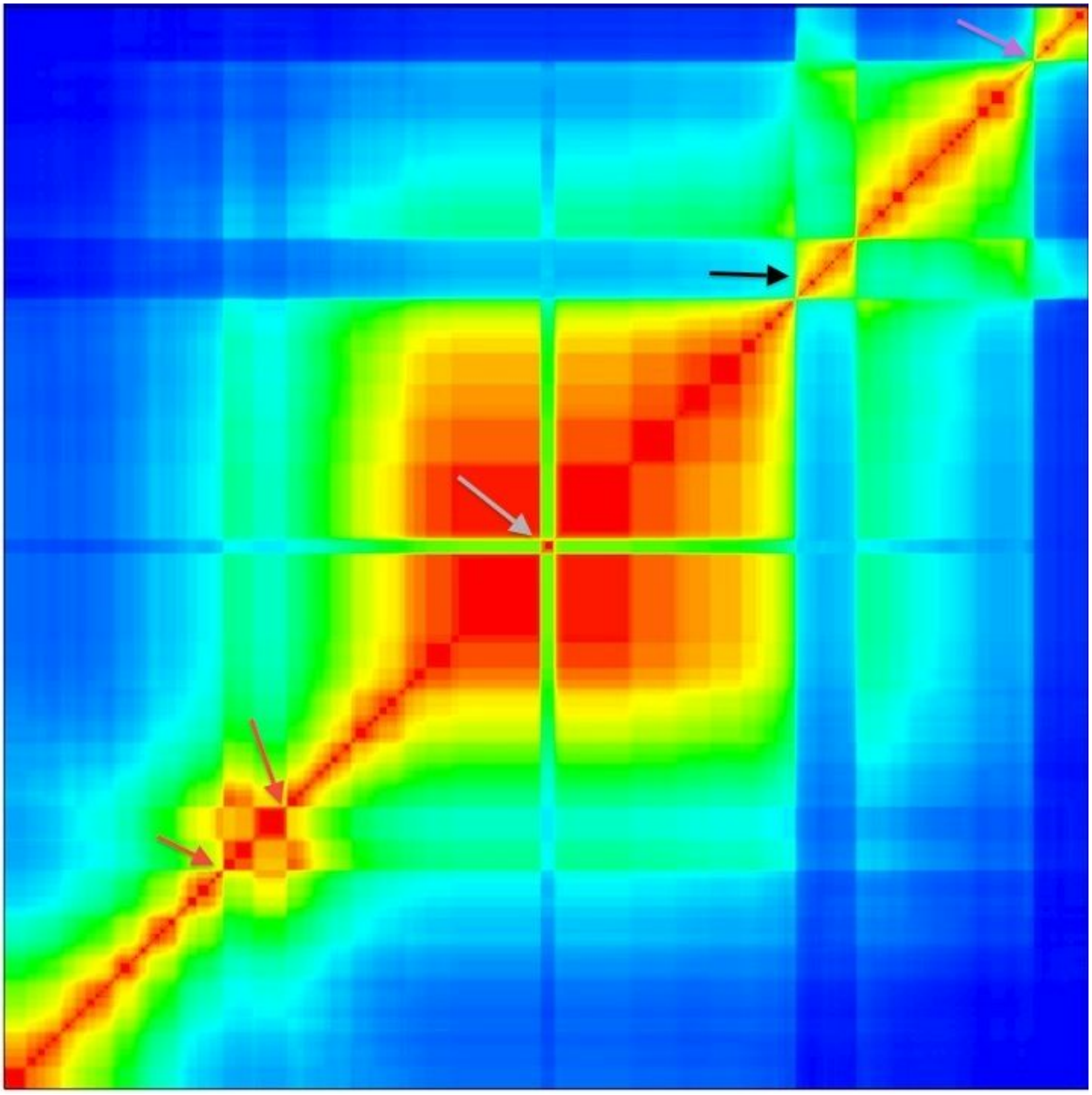
Your lod-matrix.bmp plot should look, hopefully, something like this.



Regions of red represent strong linkage and blue is no linkage. As expected, adjacent markers show strong linkage and are red. Sometimes, however, and especially if you are working with genomes that were not built from rock-solid physical maps, you will observe odd patterns. Here, for example, is a plot from data using the published *D. yakuba* genome for analysis of a backcross of a *D. yakuba* strain to a *D. santomea* strain. These data are taken from Cande et al. 2012.



Chromosomes 3 and X look fine, but there are several odd things happening on chromosome 2. Let's look more closely at chromosome 2



There appear to be three regions displaying discontinuities of the linkage pattern. First, let's consider the cross in the middle with the strong red dot, indicated with a grey arrow. It is not entirely clear what is going on here. Our best guess was that this represents repetitive DNA that is either incorrectly assembled or which tends to attract sequence reads that actually originate from multiple genomic locations. There are other possible explanations. Given our ignorance, it is best to simply mask this region for future analyses. The other odd regions are easier to interpret. On the left, there is a region showing a "turtle" pattern, demarcated with red arrows. Markers on the left (distal) appear to be more strongly linked to markers beyond (proximal to) the second red marker. The converse is also true. This is often a signal of a region that is inverted in the genome assembly relative to the actual order of markers. The block on the right

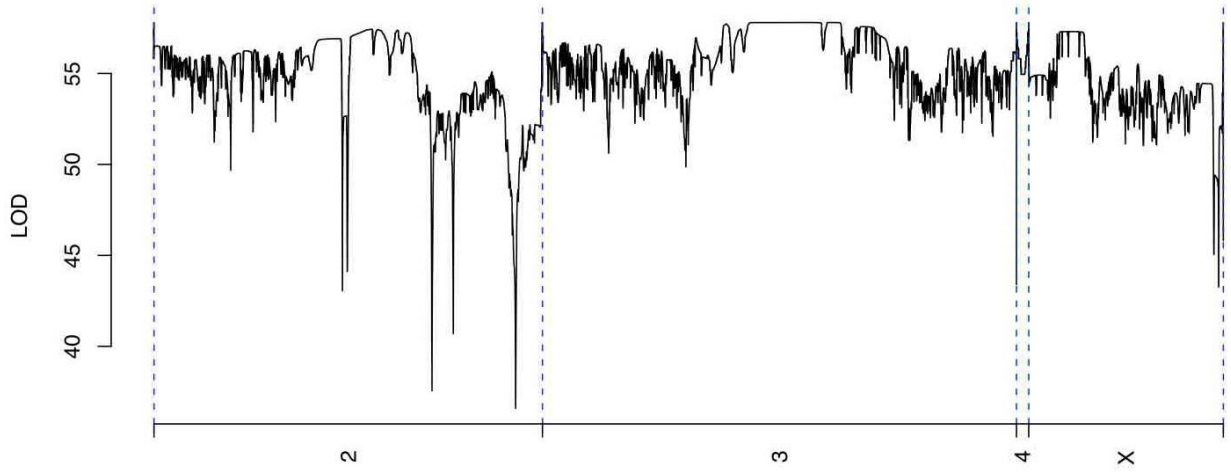
displays two signatures. First, this block is not linked to the markers adjacent to either side. Instead, this region shows linkage to a fourth discontinuity (purple arrow), suggesting that the entire black-arrow block has been transposed and belongs in the location of the purple arrow. In addition, the distal markers of the black-arrow block appear to be more strongly linked to markers to the left of the purple arrow, suggesting that the block is both transposed and inverted relative to its true orientation.

With this information, it is possible to mask and reorient the relevant genomic region. The lod-matrix plot shown at the beginning of the section is the result we got after making these changes in the parental genomes and re-running msg. To identify the breakpoints of discontinuities, we have found it useful to exploit an additional dataset, the offdiagonal-lod values, discussed in the next section.

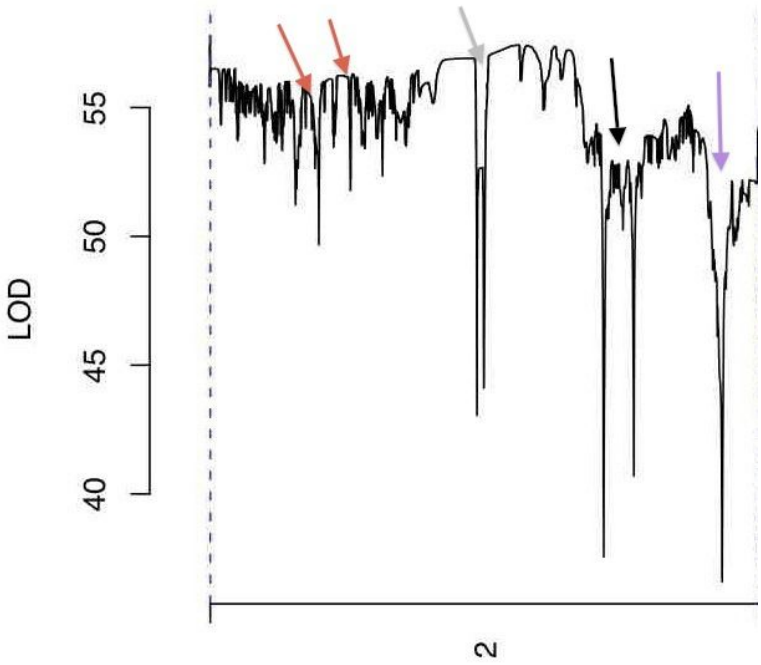
offdiagonal-lod plot

We have found that one convenient way to find the boundaries of discontinuities in the lod-matrix plots is to examine the linkage data that is one step off the main diagonal of the matrix. Neighboring sites that show tight linkage should have high LOD scores in the off-diagonal. In contrast, immediately neighboring sites that are unlinked should show a dramatic reduction in LOD score in the offdiagonal. We can find the breakpoints between physically neighboring, but genetically unlinked sites, as drops in the offdiagonal lod score.

For example, here is the offdiagonal LOD profile for the entire genome in the cross discussed above.



You can clearly see the multiple strong drops in LOD values on chromosome 2. Here is a zoom in of chr 2, with the discontinuities labelled as in the lod-matrix plot above.



You can recreate (and explore) the offdiagonal-lod plot in R by loading the offdiagonal_data.tsv file that is produced by msg.

```
> load("rlod-offdiag.rda")
> l<-rlod.offdiag
> plot(as.integer(names(l)),l,type="l")
```

Alternatively, you can examine the ˆr scores,

```
> r<-load('rhat-offdiag.rda')
> r<-rhat.offdiag
> plot(names(r),r,type="l")
```

We often find the minimum LOD score in a region that appears to contain a discontinuity as our best guess of the location of the discontinuity. For example:

First define a region you want to look at and plot it

```
> a = 1000
> b = 5000
> plot(as.integer(names(l))[a:b],l[a:b],type="l")
```

Then, find the minimum value in this region, and plot it.

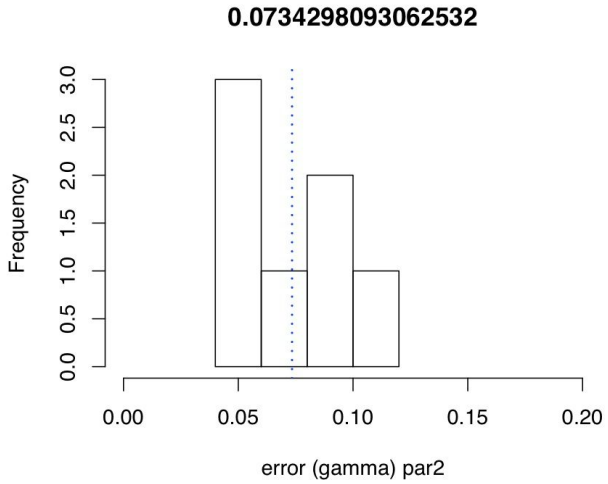
```
> x<-subset(data[a:b,], rlod==min(data$rlod[a:b])); x
    chrom       pos      rhat     rlod
```

2011 api 3075907 0.02601684 118.6057
> abline(v=x$pos, col="blue")

In this example, the minimum LOD score in this region of chromosome api

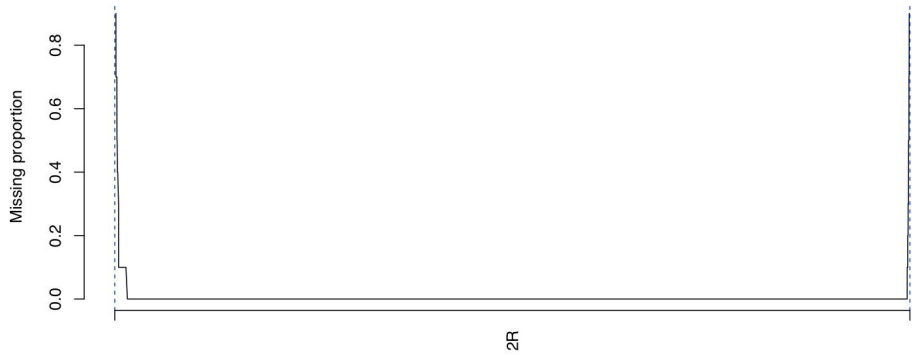
error_gamma.pdf

The error_gamma.pdf shows the distribution of error rates for each individual, as determined by calls to the alternate parent in blocks that MSG calls as homozygous in the analysis. This distribution can give you a better prior for the error parameter in the msg.cfg file. However, if errors or a too-high recombination rate are inducing many erroneous breaks in your homozygous blocks, this error rate may be an underestimate.



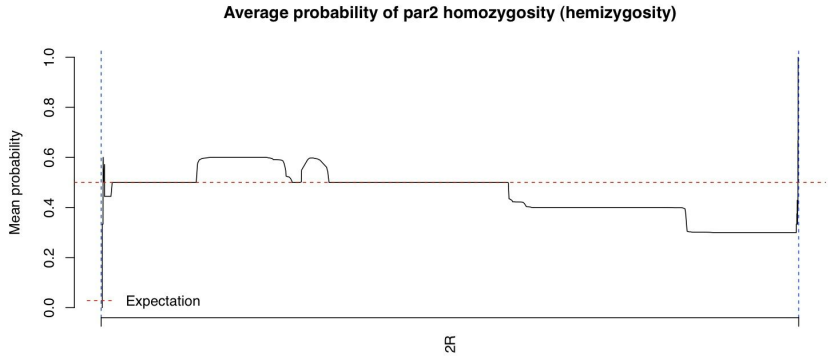
missing.pdf

The missing.pdf file shows the frequency of NA genotype calls across scaffolds (designated in chroms2plot) averaged over all individuals in the analysis. Typically there will be a higher frequency of missing data towards the beginning of the chromosome.



segregation.pdf

The segregation.pdf plot shows the probability of homozygosity for parent 2 across scaffolds (all those designated in chroms2plot) averaged over all individuals in the analysis. If you have enough individuals, this can be used to identify regions with unexpected patterns.



hmm_fits_ests.csv

This is a file that can be used as input to R-qtl. It's a an aggregated view of all individuals' hmmprob.RData information found in the hmm_fit directory.

This CSV formatted file lists one individual per row. Columns contain the est value for that individual at that site or a "-" for a missing est.
(Sites with less than pepthresh % coverage are omitted.)

To regenerate this file manually after a run (or on an old MSG run). Simply go into the working directory for the run and execute this command:

```
python msg/hmmprob_to_est.py -d hmm_fit -t .5 -o hmm_fits_ests.csv
```

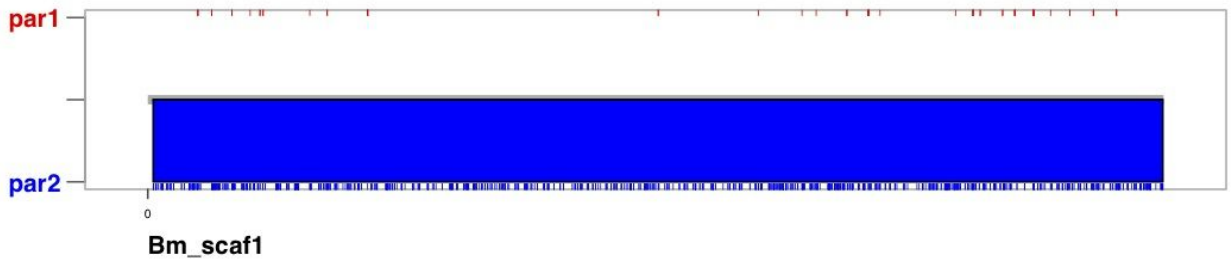
-d should be the location of the hmm_fit output files. -t should be the pepthresh from msg.cfg. -o should be the desired output file path.

It only depends on having data in hmm_fit and doesn't need any other msg files or configurations.

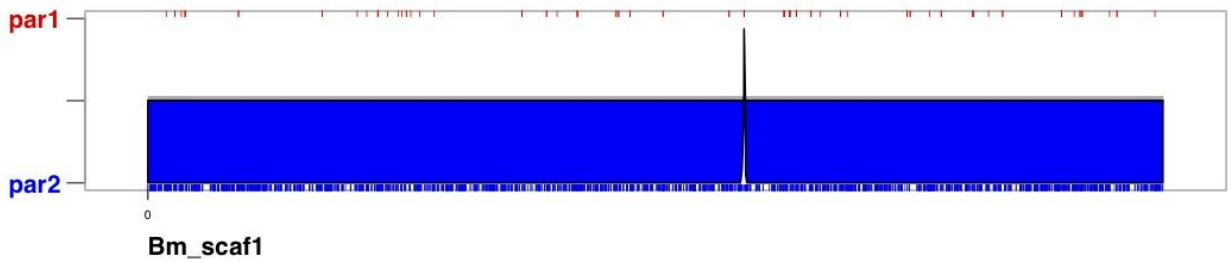
Look at the hmm_fit pdf files

After examining the plots in hmm_fit_images, it is often useful to examine the individual ancestry plots, which can be found in individual folders inside the hmm_fit folder and are called indiv*-hmmprob.pdf. In these images, solid red indicates homozygous parent 1, solid blue indicates homozygous parent 2, and non-shaded indicates heterozygous. The height of the region relative to the central black line corresponds to the posterior probability of ancestry for that block.

A chromosome completely homozygous for parent 2:



You should look at these plots to check on the quality of the HMM fit to the data. Small switches in ancestry can be indicative that the error rate (delta_par1 and/or delta_par2 in the msg.cfg file) has been set too low or recombination rate too high.



It can be handy to move all pdfs to a single folder to scan through all the individual pdf files easily:

```
$ mkdir pdf
$ mv hmm_fit/*/*pdf pdf
```

Inside the individual folders in hmm_fit there are files that indicate recombination breakpoints for chromosomes specified in chroms2plot, which can be useful to examine. Files appended *matchMismatch.csv give information about the location of breakpoints, their ancestry, and the number of markers and likelihood of ancestry by block. These can be used to calculate ancestry tract length distributions and determine whether too many (or too few) breakpoints are being introduced in the hmm compared to expectations and thus whether rfac should be modified. Files appended *breakpoints.csv contain information about uncertainty in the location of the breakpoints.

Columns of *breakpoints.csv files:

x[end_index]-x[start_index]+1, indiv, contig, start_index, end_index, x[start_index], x[end_index], endpoints

There will also be *.gff files in the individual hmm_fit folders; one file per chromosome which can be imported into the Geneious software (http://www.geneious.com/).

Look at the posterior probabilities for each genotype

The last files produced by the MSG pipeline are tab delimited ancestry files called:

```
ancestry-probs-par1.tsv
ancestry-probs-par1par2.tsv
ancestry-probs-par2.tsv
```

These files contain posterior probabilities for each genotype for each individual analyzed at all sites covered in at least one individual. `ancestry-probs-par1.tsv` contains the posterior probability that each individual is homozygous for the parent specified as parent1 in the cfg file, `ancestry-probs-par2.tsv` contains the posterior probability that each individual is homozygous for parent2, and `ancestry-probs-par1par2.tsv` contains the posterior probability that each individual is heterozygous. To see how many markers are present in your data:

```
$ head -n 1 ancestry-probs-par1.tsv | wc -w
```

To look at the ancestry tsv files, you can use:

```
$ less -S ancestry-probs-par1.tsv
```

You will see that the first column contains a list of individuals, with all subsequent columns containing genotype information for each individual at each site (NAs for missing sites). For example:

```
              group1:455    group1:1451    group1:3696    group1:3990
indiv0_ATGACA     NA            0              0              0
indiv1_ATGACA     NA          0.99942        0.999459       0.9996
indiv10_ATGACA    NA            0              0              0
indiv11_ATGACA    NA          0.000132         1e-06          0
```

Using this output, there are several quick sanity checks than can be done to ensure that the output is making sense. One is to check that sites are in hardy-weinberg equilibrium for F2 intercrosses. Another is to determine whether the output admixture proportions match expectations for the cross type.

Frequently Asked Questions (FAQ) and Troubleshooting

Can I use my paired-end data with MSG?
MSG does not currently support mapping of paired end data, but you can treat your PE reads as single end reads and use them in MSG.

How do I re-run the hmm step without re-running the whole pipeline?
Remove the hmm_fit folders:

```
$ rm -r hmm_fit*
```

Remove the ancestry files:

```
$ rm ancestry-probs-*
```

re-run by typing:

```
$ perl msg/msgCluster.pl
```

MSG never seems to finish running some individuals or never finishes running msgRun3
Try restricting the number of reads per individual. In most experiments <1 million reads are required per individual.

*Empty/absent ancestry*tsv data files or empty/absent plot files*
If MSG seems to run but the ancestry*tsv files or summary plot files are empty, check if the chromosome names you specify in chroms and chroms2plot match the parent 1 fasta file

exactly. It this is not the source of the problem, also check to make sure that you have not put in one parental genome in duplicate in the cfg file.