Bioinformatics Analysis Tools and the PoPLAR Science Gateway

Author: Mary Lauren Harris (Baylor University)
Mentor: Bhanu Rekepalli, PhD (JICS, UTK-ORNL)
Eduardo Ponce (JICS, UTK-ORNL)

Recent advances in Next Generation Sequencing (NGS) technology, generate a large volume of genetic data which is now accessible for analysis. From raw data to published results, an efficient and automated pipeline for the analysis of genetic data will revolutionize modern research. Individual programs can be optimized and placed in a science gateway for researchers to customize their pipelines. The gateway provides the ability to upload data and an interface for users to select the desired command line parameters through graphical means. Additionally, the programs themselves are wrapped and scaled to large parallel architectures, improving the performance to a level that is out of reach for single machines or small clusters. In particular, the addition of reliable high performance computing (HPC) programs to the science gateways opens the doors to computational ability, even for scientists with little to no programming experience or access to their own HPC resources. My research hopes to further the efforts to make computational power accessible for all scientists. I have looked at several downstream analysis programs, including the genotype imputation tool MaCH, and HSP-BLAST. The compilation and improvement to MaCH's current documentation will make the program easier for researchers to understand and use. Also, scripts created for HSP-BLAST provide an automated way to look at all vs. all BLAST hits, while capitalizing on HSP-BLAST's high performance capabilities. Continuing to automate and simplify the analysis processes will greatly improve future research efforts in the future.

1 INTRODUCTION

Since the first massive discovery by Watson and Crick in 1953, advances in the world of DNA sequencing continue to be made and the ideas have become ubiquitous in the scientific community. DNA is comprised of four nucleotides, adenine, thymine, guanine, and cytosine. These puzzle pieces pair with each other to ultimately form the double helix structure that is well-established in the scientific community. The genetic code found in DNA is then transcribed into RNA, and subsequently translated into amino acids that create the proteins critical for all of life's processes [1].

When starting the analysis of genetic data, there is a general workflow that can be followed. It typically begins with sequencing the collected data, and then sending that data through an assembly process. After you have an assembled sequence or genome the data can be passed to a number of downstream analysis programs. Each of these analysis programs can have a different task, like MUSCLE for multiple sequence alignment or BLAST for sequence similarity between a query and a database. After the data has been analyzed for the needs of the researcher, the results must be parsed into readable output formats or properly visualized for publications and presentation [7].

Challenges arise with the sheer abundance of data in the realm of genetic research. Terabytes of data are being generated in a matter of days, and scientists studying this data must pioneer new methods for the storage, management, and analysis of it. Computer science, specifically High

Performance Computing (HPC), had become increasingly enmeshed with the biology to fill that need. Although, while it brings unparalleled processing power to the table, it also requires a new set of skills that many biology researchers do no possess. Software developed for bioinformatics is often highly specialized and intricate. Many programs don't have a graphical interface for users, require tedious installation and runtime procedures, provide convoluted and highly technical documentation, and produce output that is unhelpful without further processing [7]. Today's researchers require access to the computational power of these methods, but unless improvements are made to the system itself knowledge will remain out of reach for scientists with few computational skills.

2 METHODS

This research focused on two downstream analysis applications: MaCH and HSP-BLAST. MaCH is an open-source genotype imputation program developed by the University of Michigan. Genotype imputation is a useful tool for scientists looking to fill in missing data from a set of genotypes [3]. The program accepts a set of genotypes, labeled with specific markers, and a set of reference files labeled with those same markers. Based on a statistical model, the program then imputes the missing genotypes and outputs the completed file along with a file containing data to describe the accuracy and quality of its prediction [5] . MaCH currently utilizes a command line interface, and the user must be able to install and configure the program on their own. Unfortunately, as is the case with much of the available free software, the documentation for MaCH was scattered across several webpages, outdated, and difficult to understand. To solve this problem and make MaCH more accessible to its users, the current README file in the source code was modified and extended to improve readability. Sections were added to describe each of the input file requirements, and explain the statistics found in each of the output files. Links to reference data in the correct file formats were specified as well. Every command line option mentioned on the webpages was documented and sample commands were given to illustrate MaCH's problem solving abilities in a variety of situations. At the end, the documentation demonstrated quality assurance tests and gave links to common programs developed for further analysis of MaCH output data. Additionally, a step-by-step tutorial on how to download, install and configure MaCH was placed at the beginning of the file.

BLAST works by taking a query sequence and using a heuristic algorithm to find subject sequences from a selected database that are similar to the query. BLAST results can be extremely useful when trying to determine a gene's function in a newly sequence genome. All v. all BLAST results are especially useful in the prediction of orthologs: a gene that performs the same function in different organisms which have evolved from a common ancestor but have since diverged due to a speciation event. Again, given an gene with an unknown purpose, finding an ortholog for that gene can pinpoint its function [6]. For this particular problem, HSP-BLAST was used. HSP-BLAST was developed by Rekepalli, et al and provided a parallel means to run the same algorithms as the serial NCBI version. HSP-BLAST scales linearly to large numbers of cores, and substantially reduces the compute time needed to run large-scale analyses [7]. An all v. all BLAST was performed on this dataset using the Darter supercomputer and Oak Ridge National Laboratory. Given three different datasets of varying sizes, a database was created for each dataset. Afterwards, every sequence in the dataset was used as a query sequence over its corresponding database. The hits were recorded and output in the default tabular format. From that tabular output, a python script was developed to parse the useful information from the file, remove all matches above a certain threshold value specified by the user, and calculate the percentage overlap between each file in the database. The percentage was calculated by determining the

number of matches, or hits, found in each file of the dataset from a given file of sequences from the same dataset. That unique number of hits per subject file was divided by the total number of sequences in the subject file to yield the percentage of the subject file matched by sequences from the query file.

3 RESULTS

Improving the quality of documentation available to MaCH users will increase the usability of the program, especially for those who have limited computational experience. Before it was difficult to find information about completing basic tasks, but now researchers have a single guide to reference and can spend more time where it matters—applying the results. This model should be a goal for all open-source software, especially that bioinformatics where many of the target users are experts in areas other than computer science. (For the contents of this file, see Appendix A.)

In the all v. all HSP-BLAST, the results were largely as expected. Theoretically when running an all v. all BLAST each file should have a 100 percent overlap with itself; that is, each query sequence in a file should at least have itself as a match. Additionally, we included a file, "ALL," in each dataset that was the concatenation of all the sequences in the entire dataset. Using the previous assumption, this file was predicted to have a 100 percent overlap with every other input file. Likewise, each input file was expected to have an overlap with the ALL file that was proportional to the number of sequences it contributed to the dataset. In these three ways, our results matched the hypotheses. Every file had a nearly 100 percent overlap with itself, and the ALL file had a nearly 100 percent overlap with every file of the dataset. When plotting the results from each specific file, and comparing the three datasets we predicted the lines to have a similar shape and that the overlap percentages would increase with the size of the database. It was here that the results differed slightly from the theory. In most cases the plotted lines were of a similar shape; however, there were several files where the largest dataset had overlap percentages that dipped below that of the smallest dataset. Similarly, the results were not symmetric—for example, if file A had 60 percent overlap with file B, B did not always have a 60 percent overlap with A. These anomalies and their origin should be the focus of further exploration. (For a sample of these visualizations, see Appendix B.)

4 FUTURE WORK

Further investigation into the results from all v. all HSP-BLAST could lead to valuable insights about the dataset that was used, and continued improvements in the documentation of analysis software will increase their usefulness for researchers.

Ultimately, however, the future lies in the establishment and expansion of science gateways. These gateways will provide a secure, centralized location for researchers to upload their data. Software for all steps of the a genomic analysis workflow will be available from within the portal. These may be customizable, or situated into predefined workflows for the user. After uploading the data, scientists will be able to choose their programs and set the proper parameters for their input through a graphic interface. After the jobs are submitted, output is collected in a centralized location—either for input into another program, or to be interpreted by the researcher.

PoPLAR is one such gateway. The Portal for Petascale Lifescience Applications and Research is the work of Rekepalli, et al, and it aims to improve the quality and accessibility of bioinformatics tools for the modern scientist. PoPLAR is set apart by its use of Highly Scalable Parallel (HSP) architecture which currently allows three programs to make use of the vast resources available on today's supercomputers. BLAST, HMMER, and MUSCLE have already been modified for the HSP architecture, and the goal is for many more to

follow suit. This gateway, much like the generic description above, allows users to sign in, upload data, perform tricky computational procedures with the help of a web-based interface, and download results [7]. While still under development now, in the future this gateway will offer tools from the beginning stages of genome assembly through the latest in downstream analysis tools.

5 CONCLUSIONS

Biological research produces immense amounts of data and its scope continues to grow with each passing year. In order to keep up with such large-scale problems, new computing strategies must be adopted. However, these often come with disadvantages, like insufficient documentation or requiring computational abilities beyond the reach of many scientists. Creating improved documentation for the genotype imputation program MaCH is a step towards making available software more accessible for all its users. By using HSP-BLAST to run all v. all tests on three datasets, valuable information was produced that can now be further analyzed in an additional study. The use of parallel architectures greatly reduced the time needed to preform these tests, and a parsing and visualization script made the results readable for any researcher. Soon a science gateway approach, such as that used by PoPLAR, will replace the scattered command line utilities currently available. The gateways will allow users with all levels of computational experience to upload raw files, analyze data, and share results—all using a graphic interface that eliminates the need for a strong background in computer science or the availability of technical experts.

6 ACKNOWLEDGEMENTS

I'd like to thank the CSURE REU program and the NSF for the funding and the opportunity to be a part of this research. I'd also like to thank my mentor Bhanu Rekepalli for guidance and training during this project, Eduardo Ponce for his hours of help with HSP-BLAST, and Kwai Wong for our initial training and his help debugging.

7 REFERENCES

1. Dewey, F. E. (2012-02-21). DNA sequencing: clinical applications of new DNA sequencing technologies. Circulation (New York, N.Y.), 125(7), 931.
2. Gužvić, M. (2013-10-01). The History of DNA Sequencing / ISTORIJAT SEKVENCIRANJA DNK. Journal of medical biochemistry, 32(4), 301.doi:10.2478/jomb-2014-0004
3. Howie, B. Genotype imputation for genome-wide association studies. *Nature Reviews Genetics*, 499-511.
4. Hunter, J. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, *9*, 90-95.
5. MACH 1.0 - Markov Chain Haplotyping. (n.d.). *MACH 1.0 - Markov Chain Haplotyping*. Retrieved from http://www.sph.umich.edu/csg/abecasis/MaCH/
6. Moreno-Hagelsieb, G., & Latimer, K. (2007, November 26). Choosing BLAST options for better detection of orthologs as reciprocal best hits.
7. Rekapalli et al.: PoPLAR: Portal for Petascale Lifescience Applications and Research. BMC Bioinformatics 2013 14(Suppl 9):S3

APPENDIX A.

MaCH is a tool for haplotyping, genotype imputation and disease association analysis developed by Goncalo Abecasis and Yun Li. MaCH was first used to imputed missing genotypes in our FUSION genomewide association study (Scott et al, Science, 2007) and has since been used in the analysis of many other GWAS.

This file explains how to install MaCH 1.0 on NICS computing resources, although the instructions are relevant for all Linux systems. Below the numbered instructions is a modified and extended version of the README file provided by the developers to explain the options, inputs, and how to run MaCH 1.0. This file includes information from both the MaCH homepage and wiki as well.

Contents: Installation, Input, Options, Execution/Examples, Output Quality Assessment, Further Analysis, Troubleshooting, Additonal Resources

```
====================
= INSTALL MACH 1.0 =
====================
```

1. Get the tar file from the MaCH website using the following command:

   ```
   wget http://www.sph.umich.edu/csg/abecasis/MaCH/
   download/mach.1.0.18.source.tgz
   ```

2. Untar the files using the tar command.

   ```
   tar -xf mach.1.0.18.source.tgz
   ```

3. Navigate to the directory containing 'Makefile,' if necessary.

   ```
   cd directory/with/new/files
   ```

4. Ensure you are using g++ as the compiler. If you use NICS resources, you should use the CC wrapper; and you may have to swap modules. Use 'module list', 'module avail' and 'module swap <old> <new>' commands to accomplish this.

```
        make all

        OR (NICS-- ie. Darter)

        module swap PrgEnv-cray PrgEnv-gnu
        module list (look for gcc)

        make all CXX=CC (to use the CC compiler wrapper)
```

5. Install the binaries in /usr/local/bin by using the
   following command:

```
        make install

        OR

        make install=/directory/to/install
```

6. Follow the instructions in the next section to create input
   files.

7. Run MaCH 1.0 using the examples and options provided below,
   or use the provided PBS script. (darter.pbs)

```
===============
= INPUT FILES =
===============
```

MaCH 1.0 needs a Merlin format data and pedigree files as input.

A simple MaCH data file lists the names of a series of genetics markers. These are preceded by an 'M' and each marker name has its own line. The data file should look like this:

```
 M marker1
 M marker2
 ...
```

The pedigree file contains the actual genotpyes. It should list one individual per row. Each row should start with an family id and individual id, followed by a father and mother id (which should both be 0, 'zero', since mach1 assumes individuals are unrelated), and sex. These initial columns are followed by a series of marker genotypes, each with two alleles. Alleles can be coded

as 1, 2, 3, 4 or A, C, G, T.

For example:

```
FAM1001   ID1234  0   0   M   1 1   1 2   2 2
FAM1002   ID1234  0   0   F   1 2   2 2   3 3
...

OR

FAM1001   ID1234  0   0   M  A A   A C   C C
FAM1002   ID1234  0   0   F  A C   C C   G G
...
```

To make the genotypes easier to read, '/' may be used between the two alleles. This pedigree is equivalent:

```
FAM1001   ID1234  0   0   M   A/A   A/C   C/C
FAM1002   ID1234  0   0   F   A/C   C/C   G/G
...
```

Missing genotypes may be encoded with a '.' or a '0'. The following two lines are
both missing the first genotype.

```
FAM1001   ID1234  0   0   M   ./.   A/C   C/C
FAM1002   ID1234  0   0   F   0/0   C/C   G/G
...
```

For many analyses, but in particular for genotype imputation, it can be very helpful to provide a set of reference haplotypes as input. Reference haplotypes can include genotypes for markers that were not examined in your own sample but which can, often, be inputed based on genotypes at flanking markers. Most commonly, these haplotypes might be derived from a public resource such as the International HapMap Project and, eventually, the 1000 Genomes Project.

You can retrieve a current set of phased HapMap format haplotypes from:
   http://hapmap.org/downloads/phasing/2007-08_rel22/phased/

HapMap III phased haplotypes are in different format, you will need to use converted haplotypes available at:
   http://www.sph.umich.edu/csg/yli/mach/download/HapMap3.r2.b36.html

Additional reference files (e.g., those based on data from the 1000 Genomes Project; combined reference files) can be found through links at:
http://www.sph.umich.edu/csg/yli/mach/download/

Phase haplotype information is encoded in two files. The first file (which MACH calls the "snp file") lists the markers in the phased haplotype. The second file (which MACH calls the "haplotype file") lists one haplotype per line. If you retrieved these files from the HapMap website, simply combine the –hapmapFormat option with the --snp option to indicate the name of the HapMap legend file and the --haps option to indicate the name of the file with phased haplotypes.

The snp file should look like this:

```
marker1
marker2
...
```

If including your own haplotype reference file, it should be formatted in the following way:

```
FAMILY1->PERSON1 HAPLO1 CGGCGCGCTTGGC
FAMILY1->PERSON1 HAPLO2 CGGCGCGTCCAGC
FAMILY2->PERSON1 HAPLO1 GGGCGCGCTTGGC
FAMILY2->PERSON1 HAPLO2 GGAAGCACTCGGC
...
```

If you provide a MACH a set of reference haplotypes as input, the marker order in
the phased haplotypes overrides any marker order that may be specified in the pedigree and data files that contain the genotype data.

Additionally, to save space, all input files can be compressed using gzip. MaCH automatically recognizes this and will decompress the files for you.

```
==========================
=  COMMAND LINE OPTIONS  =
==========================
```

The command line options for MaCH 1.0 are listed below. For example usage, see the next sections.

Option		Description
-d (or --datfile) <file>	*	Specify the name of the data file (MERLIN format)
-p (or --pedfile) <file>	*	Specify the name of the pedigree file (MERLIN)
-h (or --haps) <file>	+	Specify a reference haplotype file
-s (or --snps) <file>	+	Specify the list of SNPS for haplotype file
--hapmapFormat		Used with the -h and -s options for reference haplotypes from HapMap that AREN'T in MaCH format. (-h now indicates the file with phased haplotypes and –s indicates the legend)
--phase		Request the output of phased chromosomes
--rounds <k> (or -r <k>)		Use k iterations of Markov sampler; recommended = 50
--states <k>		Use a random subset of k haplotypes as reference; default = ALL,recommended >= 200
--weighted		Favor individuals with more genotype data as the template for haplotyping other individuals
--greedy		Use only reference haplotypes (-h)
--geno	~	Infer genotypes at untyped markers
--crossovermap <file>		Specify the file of a crossover map (.rec) from a previous MaCH run to be used in imputation
--errormap <file>		Specify the file of an error map (.erate) from a previous MaCH run to be used in imputation

--mle	Replaces the --geno option when using crossovermap and errormap options; requests that MaCH do a maximum likelihood genotype imputation
--mldetails	Provide additional details with -mle
--prefix <name>	Specify a prefix for MaCH output files (.rec, .erate, .info/.mlinfo)
--compact	Reduces memory usage
--poll <k>	Request intermediate solutions every k iterations
--sampleInterval <k>	Output every k rounds based on random sampling from the last Markov iteration
--interimInterval <k>	Output every k rounds by building consensus from all previous Markov iterations
--dosage	Generate a .dose file containing dosages (ie. estimated counts) of the reference allele in each individual; range = 0.0-2.0
--quality	Generate a .qc file containing quality scores for each imputed genotype; equal to the posterior probability for the most likely genotype
--mask <n>	Quality assessment; masks a small portion of the genotypes to compare the imputed values with the actual values at these locations; n = the percentage of genotypes to mask (eg. 2% = 0.02)
--autoFlip	Flips alleles in pedigree file according to base pairing if >2 alleles are found to have resolved any labeling inconsistency; will drop markers if the problem remains unresolved

```
* Required parameters for all MaCH runs
+ Additional required parameters to use reference haplotypes
~ Additional required parameters to infer untyped markers

==============================
= HAPLOTYPING USING MACH 1.0 =
==============================

The following will help you use MaCH to haplotype a sample of unrelated
individuals.

FILES:

   Obtain a pedigree file and data file (both in Merlin format). Make sure that
   markers are ordered according to their physical position.

USEFUL COMMAND LINE OPTIONS:

   Specify the data and pedigree files with the -p and -d options,
respectively.

   Use the --phase option to request the output of phased chromosomes.

   The key parameters for managing the quality of inferred haplotypes and the
amount of computational effort expended in generating them are --rounds and --
states.
        --rounds <k> : Larger numbers will result in better solutions. If
        there isn't much missing data, a value of 50 should give a reasonable
        solution. Larger values will provide better solutions.
        --states <k> : Larger values will generate more accurate solutions,
        but may slow things down and require more memory. A value >=200
        typically provides quite good solutions. The default is to use all
        available haplotypes.

   If missing data is not distributed evenly among the available individuals
   consider the --weighted parameter.

   --compact will reduce memory use.

   --poll <k> will request intermediate solutions every k iterations.

EXAMPLE USAGE:
```

```
   mach -d sample.dat -p sample.ped --rounds 50 --states 200 --phase

========================================
= INFER UNTYPED MARKERS USING MACH 1.0 =
========================================
```

Genotypes at untyped markers for each individual are inferred by comparing the available genotypes to those in other individuals that have been typed at higher density. Individuals typed at high density will often come from public resources, such as the HapMap.

The following will help you use MaCH to infer genotypes at untyped markers. There are two strategies, although #2 is more commonly used. In addition, there are a couple methods to speed up the imputation process. The first, labeled as strategy #3, is using data files from previous MaCH runs to improve performance. The second, labeled as #4, is a 2-step imputation process.

In general, you should use the --geno option, regardless of how you include the reference haplotypes. This will cause the behavior described in the first paragraph.

= STRATEGY #1 =
INCLUDE REFERENCE (eg. HAPMAP) GENOTYPES TO YOUR DATASET

FILES:

A simple way to infer missing genotypes is to create one large pooled dataset, following the input guidelines above. Some individuals will have missing data and others will have much more complete genotyping information.

USEFUL COMMAND LINE OPTIONS:

Estimate the most likely genotype for each individual with --geno.

Use the command line options --dosage and --quality options to request additional information about each inferred genotype (See options table above).

= STRATEGY #2 =
USE REFERENCE (eg. HAPMAP) HAPLOTYPES AS INPUT:

FILES:

If you select this option, you should generate a file that includes a set of reference haplotypes. These can be typed at more markers than are available in your sample. You will also need a small file that lists all the markers that appear in the phased haplotypes.

Then, to estimate missing genotypes, you'll need to provide the Merlin format data and pedigree files, the reference haplotypes and the list of SNPs in the reference haplotypes.

USEFUL COMMAND LINE OPTIONS:

Name the reference haplotype and snp list files with --haps and --snps.

If you use the --autoFlip option, MaCH 1.0 will try to automatically resolve problems with alleles that are inconsistently labeled in your sample and the reference panel (by flipping strands and dropping markers where this trivial solution does not help).

Most of the time, you'll get good estimates of genotypes at untyped markers using the --rounds <k> and --greedy option.

If you don't use the --greedy option, you can control computational effort with the --weighted and --states <k> options. However, this alternative strategy generally requires more iterations before converging to a good solution.

EXAMPLE USAGE:

mach -d sample.dat -p sample.ped -h hapmap.haplos -s hapmap.snps --rounds 50 --greedy --geno

mach -d sample.dat -p sample.ped -h hapmap.haplos -s hapmap.snps --rounds 500 --states 200 --geno

mach -d sample.dat -p sample.ped -h hapmap.haplos -s hapmap.snps --rounds 500 --states 200 --weighted --geno

*NOTE: It is very important to ensure that alleles are labeled consistently in your sample and in the reference panel. MaCH 1.0 will automatically warn you about alleles that differ in frequency greatly between your sample and the reference panel or that have different allele names in the two subsets of data. However, these checks will not catch all inconsistently labeled alleles.

= STRATEGY #3 =
USING DATA FROM PREVIOUS RUNS TO SPEED UP IMPUTATION

The standard genotype imputation approach, described as strategy #2 works best when you execute a large number of iterations of the Markov Chain (50-100). These iterations are used to simultaneously update the crossover map (which determines the likely locations for haplotype transitions), to update the errorrate map (which flags unusual markers), and to estimate the missing genotypes.

An alternative approach is to use a single set of estimates for the crossover and error rate maps and, conditional on these, to find the most likely genotypes. This approach seems to work quite well.

FILES:

You will include the files associated with the -d, -p, -s, and -h options as well as the.rec and .erate files to specify estimates of error and crossover rates from a previous MaCH run.

If you don't have an available set of map estimates, you can request that MaCH estimate them using a small number of iterations of the Markov Chain with the --rounds <k> option.

USEFUL COMMAND LINE OPTIONS:

Request the --mle option instead of --genos.

Use --crossovermap to specify the .rec file and --errormap for the .erate file.

EXAMPLE USAGE:

mach -d sample.dat -p sample.ped -h hapmap.haplos -s hapmap.snps --crossovermap mach.rec --errormap mach.erate --greedy --mle

mach -d sample.dat -p sample.ped -h hapmap.haplos -s hapmap.snps --greedy --mle --rounds 5

= STRATEGY #4 =
TWO-STEP IMPUTATION FOR IMPROVED SPEED

1. ESTIMATING MODEL PARAMETERS

The first step is to build a model that relates your samples to the haplotypes in the reference panel. This model includes both an estimate of the "error" rate for each marker (an omnibus parameter which captures both genotyping error, discrepancies between your platform and the reference panel, and recurrent mutation) and of "crossover" rates for each interval (a parameter that describes breakpoints in haplotype stretches shared between your samples and the reference panel).

The key choices for this first step are the number of iterations expended in estimating model parameters (specified with the --rounds parameter) and the number of individuals in your sample to used for model building. In small samples, it is often okay to include your entire sample in this model parameter estimation step, in larger samples it is usually sufficient to include a random subset of 200-500 individuals in this step.

Once all iterations are completed, MACH will store model parameters in two files, a .rec and a .erate. These files as input for the next step, where model parameters will be fixed.

2. GENOTYPE IMPUTATION

This step is relatively quick and uses the parameters estimated in the previous round and calibrated to your specific dataset and genotyping platform to impute all SNPs in the reference panel in your sampled individuals.

FILES:

Besides the normal files to satisfy the -d, -p, -s, and -h options, you will be using output from the first MaCH run as input into the second.

USEFUL COMMAND LINE OPTIONS:

Use --crossovermap to specify the .rec file from the previous run.

Use --errormap to specify the .erate file from the previous run.

--prefix will allow you to specify a prefix for the MaCH output files other than the default "mach1". This helps differentiate between step 1 and step 2 files.

Use --greedy and --rounds <k> (or --r <k>).

The compact option can help save memory when using large datasets.

Use the two options --mle and --mldetails (instead of --geno) to carry out the maximum lilkelihood genome imputation.

EXAMPLE USAGE:

(1) mach1 -d gwas.dat -p gwas_subset.ped -s hapmap.legend -h hapmap.phased --hapmapFormat --greedy -r 100 --prefix step1
(2) mach1 -d gwas.dat -p gwas.ped -s hapmap.legend -h hapmap.phased --hapmapFormat --crossover step1.rec --errormap step1.erate --greedy --mle --mldetails --prefix step2

= OUTPUT FILES =

For every MaCH 1.0 run, whether just haplotyping or performing genotype imputation, two output files are generated:

Filename	Description
<prefix>.out.rec	This file contains per marker error rates. It has 3 columns labeled Marker, AvgRate, and LastRate.
<prefix>.out.erate	This file contains the mosaic crossover rates. It has 3 columns labeled Interval, AvgRate, and LastRate.

When MaCH is used to haplotype unrelated individuals a <prefix>.out file is generated which contains the phased chromosomes.

MaCH 1.0 generates a table that provides useful information about each marker. The filename for the table has the extension .info when the --geno option is used
or .mlinfo with the --mle option.

.info Column Name	Description
SNP	Marker name for the SNP
AL1	Allele 1 label (numerically coded)
AL2	Allele 2 label (numerically coded)

Freq1	Frequency for allele 1
MAF	Minor allele frequency
Quality	The estimated probability that an average imputed genotype will match an experimental genotype(this should be 1.0 for genotyped markers, and will often be less for untyped markers).
Rsq	An estimate of the r-squared correlation between an estimated genotype scores and true genotypes.

Note that higher quality scores for a given frequency are typically better imputed; however, it is difficut to compare these values for markers with different minor allele frequencies.

Also note that typically a cutoff of 0.30 for the Rsq value will flag most of the poorly imputed SNPs, but only a small number (<1%) of well imputed SNPs.

Filename	Description
<prefix>.out.geno	Contains the best-guess (ie. most likely) genotype for each individual at each SNP (with --geno option)
<prefix>.out.qc	Contains a quality scores for each imputed genotype. The quality score is the posterior probability for the most likely genotype, ranging from 0-1. (with –geno)
<prefix>.out.dose	Contains dosages (ie. estimated counts) of the reference allele (Al1 in .info) in each individual. These estimates may be fractional and range from 0.0 to 2.0 (with --geno)
<prefix>.out.mlgeno*	Contains the best-guess (ie. most likely) genotype for each individual at each SNP
<prefix>.out.mldose	Contains dosages (ie. estimated counts) of the reference allele (Al1 in .info) in each individual. These estimates may be fractional and range from 0.0 to 2.0 (with --mledetails)
<prefix>.out.mlqc *	Contains a quality scores for each imputed genotype.

```
                    |    The quality score is the posterior probability for
                    |    the most likely genotype, ranging from 0-1.

<prefix>.out.mlprob*|    Contains posterior probabilities for the A1/A1 and
                    |    A1/A2 genotypes at each marker for each individual.
```

* Only generated with -mle and -mldetails options. (.mldose and .mlqc can be individually generated using --dosage and --quality options with --mle)

Some files will be compressed to save disk space. Output files with an additional .gz extension contain all the valid information and are usable in MaCH programs.

```
======================
= QUALITY ASSESSMENT =
======================
```

One simple way to empirically assess quality of the solutions generated by MaCH is to use the mask option. This option hides a small proportion of genotypes from the haplotyper and then compares the imputed genotypes at these locations with the actual genotypes.

EXAMPLE USAGE:

```
  mach -d sample.dat -p sample.ped --rounds 50 --states 200 --mask 0.02

  mach -d sample.dat -p sample.ped -h hapmap.haplos -s hapmap.snps --rounds 50
--greedy --mask 0.02
```

A better approach is to mask a small proportion of SNPs (vs. genotypes in the above simple approach). One can generate a mask.dat from the original .dat file by simply changing the flag of a subset of markers from M to S2 without duplicating the .ped file. Post-imputation, one can use CalcMatch and doseR2.pl to estimate genotypic/allelic error rate and correlation respectively. Both programs can be downloaded from:

http://www.sph.umich.edu/csg/ylwtx/software.html

Note that any masking procedure should be performed as a separate quality verification measure. All availabe information should be used for production.

For more information about the interpretation of these quality measures, please see the FAQ page on the MaCH wiki (link below).

```
====================
= FURTHER ANALYSIS =
====================
```

For further analysis of MaCH output, try mach2dat.

After you have performed the imputation, you can directly use MaCH output to assess association for quantitative and qualitative traits in unrelated samples. You will need the .ped and .dat files in Merlin format to specify the disesase status or quantitative trait of interest (indicated with A and T respectively in the .dat file). Then, you can run the association using the following line:

```
   mach2dat -p myfile.ped -d myfile.dat --infofile myfile.mlinfo --dosefile
myfile.mldose
```

where myfile.mlinfo and myfile.mldose are the MaCH output files.

You can also add covariates to the phenotype .ped and .dat files if you want to adjust your test for other variables.

Please note that mach2dat analyzes only unrelated samples. If you input a pedigree with family relationships, those will be ignored.

Find the latest version at the Li software page:
http://www.unc.edu/~yunmli/software.html

```
====================
= TROUBLESHOOTING =
====================
```

If you see this message-- "undefined symbol: gzopen64" -- recompile the program using the commands below:

```
   make clear
   make all
```

For other problems or questions, consult further documentation on the MaCH homepage or the wiki. If problems persist, email Yun Li or Goncalo Abecasis (contact information provided online).

```
==========================
= ADDITIONAL INFORMATION =
==========================
```

For additional information, tutorials, downloads, and FAQs please see the MaCH homepage at:

http://www.sph.umich.edu/csg/abecasis/MaCH/index.html

Or check out the wiki:

http://genome.sph.umich.edu/wiki/MaCH

If you use MaCH, mach2qtl or mach2dat, please cite:

Li Y, Willer CJ, Ding J, Scheet P and Abecasis GR (2010) MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. Genet Epidemiol 34:816-834.

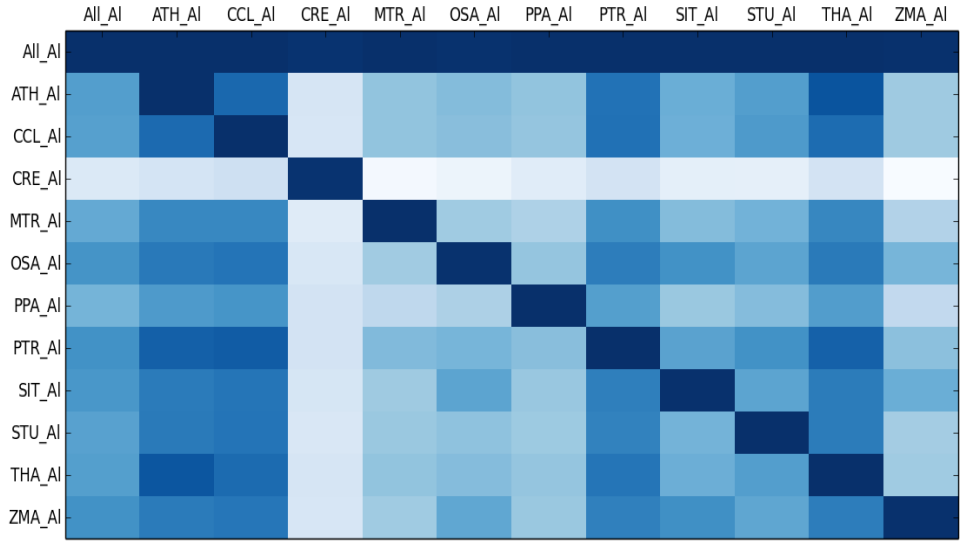
Li Y, Willer CJ, Sanna S and Abecasis GR (2009) Genotype Imputation. Annu Rev Genomics Hum Genet 10:387-406.

And, if you download MaCH, you should register your name and email on the MaCH homepage to receive updates and bug-fixes.

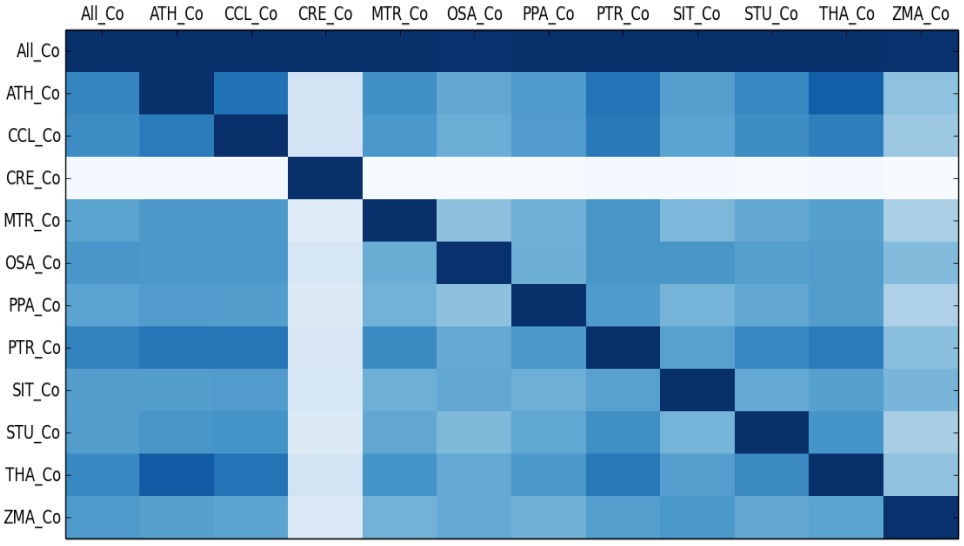
Appendix B.

All of the images in this document were generated with an E-value threshold of $1x10^{-3}$.

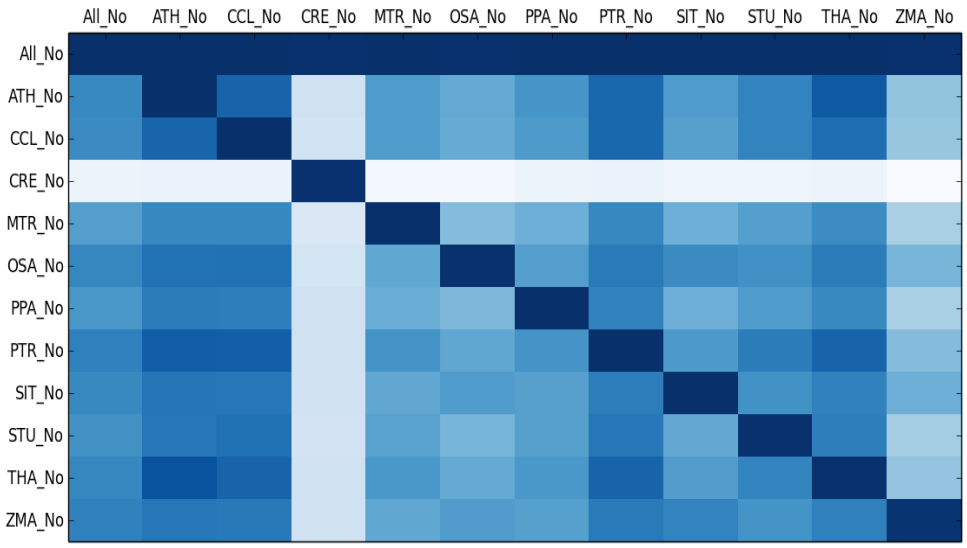
HEAT MAPS:



This heat map represents the all v. all BLAST run using the [file]_AllHits_protein.fa dataset. The darker colors represent higher percentage overlap values between the files. The files on the left side of the image are the query files, and the files at the top of the image represent the file where the "hits" were found. The All_AL file was not found in the original dataset—it is a concatenation of the unique sequences from each of the other files in the dataset. Percentages were calculated by taking the number of unique hits found in subject file B from sequences in query file A, and dividing that number by the number of unique sequences in file B.



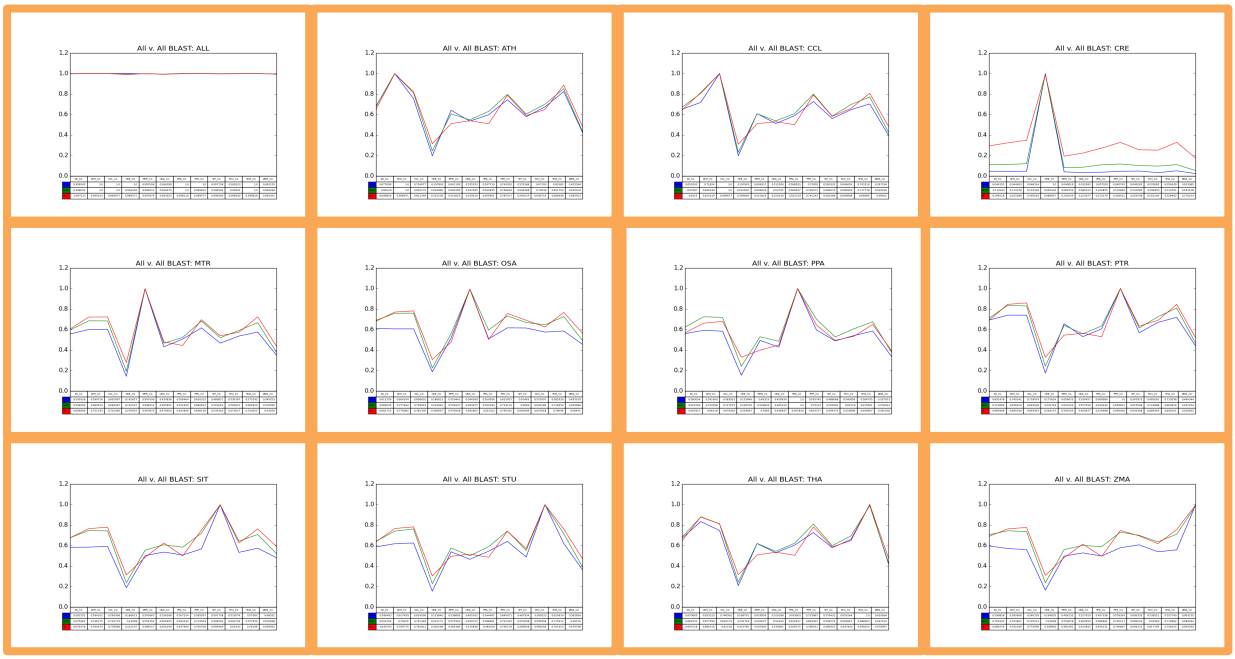
This heat map represents the all v. all BLAST run using the [file]_Con_protein.fa dataset.



This heat map represents the all v. all BLAST run using the [file]_Non_protein.fa dataset.

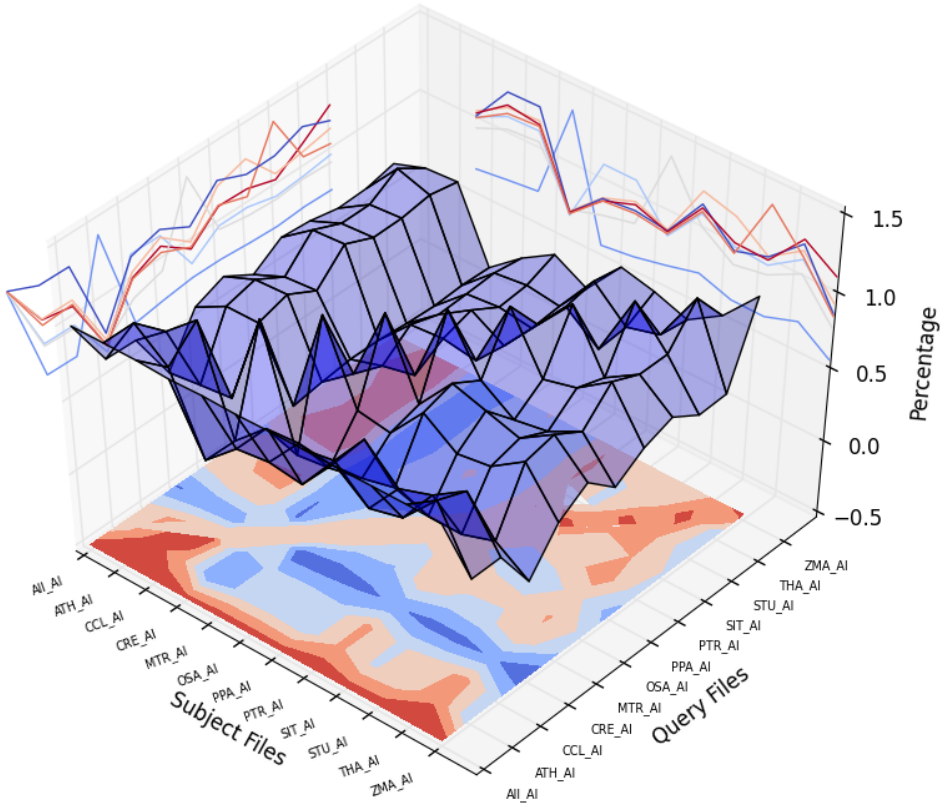
LINE PLOTS:

The title of each line plot reads "All v. All BLAST: [file]" where [file] is the name of the file being compared from each dataset. The file "ALL" is not was not found in the original dataset—it is a concatenation of the unique sequences from each of the other files in the dataset. The blue line represents the [file]_Con_protein.fa dataset, the green line represents the [file]_Non_protein.fa dataset, and the red line represents the [file]_AllHits_protein.fa dataset. These are abbreviated in the tables below the graph as "CON," "NON," and "ALL." Also in the tables below the graph are the actual percentage values (shown as a decimal) for the overlap between the files. The y-axis (labeled with each dataset's abbreviation) represents the query file given in the title and the top x-axis shows each of the subject files. Each row is from a separate dataset, as labeled. The graph itself has an x-axis labeled with the files and a y-axis that represents percent as a decimal (0%-120%).

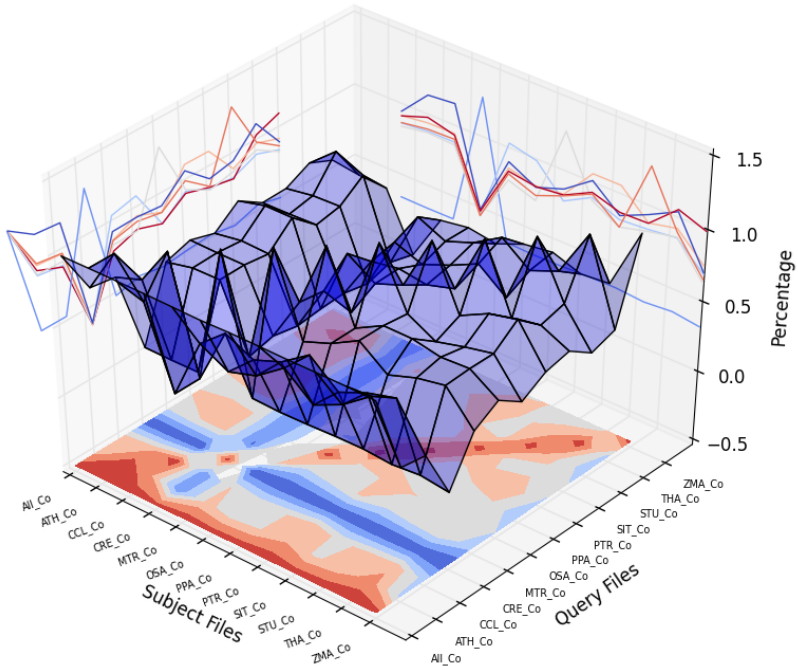


3D VISUALIZATIONS:

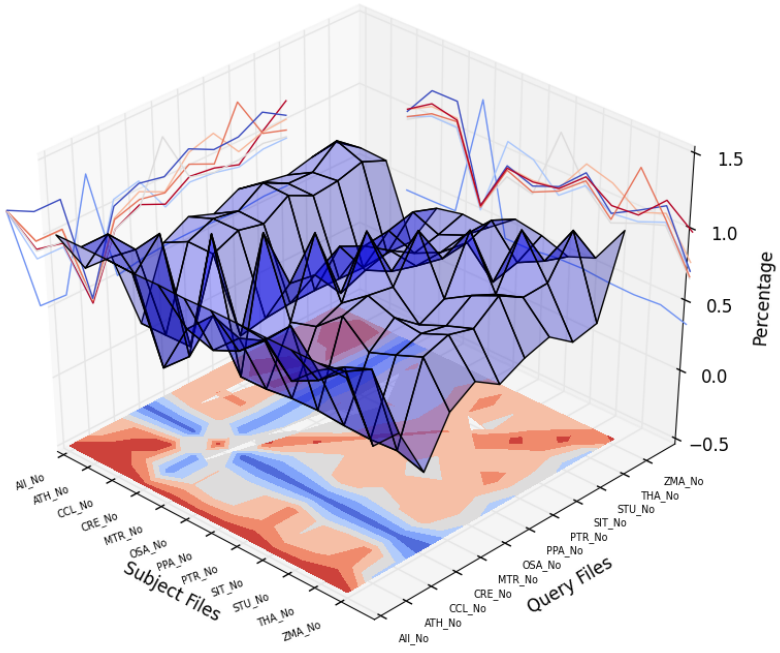
The 3D visualizations below have been created for each of the three datasets. The Query Files axis shows the files where the query sequences were taken from, and the Subject Files axis shows the files that contained hits from the query files' sequences. The heat map at the bottom of the graph shows the percentage overlap (much like the heat maps in the beginning of this document) where the dark red areas have very high percentages and the dark blue areas have very low percentages. Lighter shades of red and blue denote the percentages in between. The purple mesh in the middle shows a 3D representation of this heat map. The peaks represent high percentages and the valleys represent low ones. The percentage scale is on the vertical z-axis. The line graph projected opposite of the Query Files axis renders the peaks and valleys of the mesh in 2D. The line graph projected opposite of the Subject Files does the same thing, but from the opposite perspective. The peaks represent high percentages (and valleys for low ones) in the line graphs as well.



This 3D image is a rendering of data from the all v. all BLAST run using the [file]_All_protein.fa dataset. The All_Al file was not found in the original dataset—it is a concatenation of the unique sequences from each of the other files in the dataset.



This 3D image is a rendering of data from the all v. all BLAST run using the [file]_Con_protein.fa dataset. The All_Co file was not found in the original dataset—it is a concatenation of the unique sequences from each of the other files in the dataset.



This 3D image is a rendering of data from the all v. all BLAST run using the [file]_Non_protein.fa dataset. The All_No file was not found in the original dataset—it is a concatenation of the unique sequences from each of the other files in the dataset.