



hmmSNP User Guide

**Prepared by: Glen Beane, Jin Szatkiewicz**

Computational Sciences

The Jackson Laboratory

Document modification history

|  |  |  |  |
| --- | --- | --- | --- |
| ***Revision number*** | ***Date*** | ***Who*** | ***Comments*** |
| ***0*** | 10/26/06 | gbeane | Initial draft of user guide |
| ***1*** | 11/7/06 | gbeane | Minor updates |
| ***2*** | 11/09/06 | gbeane | Added info about training/test data, updated example PBS script, updated command line options |
| ***3*** | 11/12/06 | gbeane | Added suggestions from Jin |
| ***4*** | 4/12/07 | gbeane | Updating for version 1.1.0 |
| ***5*** | 5/25/07 | gbeane | More updates for 1.1.0 |
| ***6*** | 8/29/07 | gbeane | Added program flow |
| ***7*** | 9/5/07 | gbeane | Added info about perl script tools, running with –no\_em option |
| ***8*** | 09/25/07 | jinpengs | Made corrections |
| ***9*** | 10/03/07 | gbeane | Corrections |
| ***10*** | 10/11/07 | gbeane | RIL information and examples |
| ***11*** | 10/16/07 | gbeane | Updated current version and release date |
| ***12*** | 02/13/08 | gbeane | Updated for version 1.2.0 |
| ***13*** | 02/17/08 | gbeane | Update for version 1.2.2 |
| ***14*** | 03/28/08 | gbeane | Version 1.2.3 |
| ***14*** | 03/31/2008 | gbeane | Minor updates for version 1.2.3 |
| ***15*** | 05/09/2008 | gbeane | Added data cleaning steps |
| ***16*** | 05/06/2009 | gbeane | Minor corrections |
| ***17*** | 6/9/09 | Gbeane | Add use cases from Jin,  Remove old JSNP file format info |
| ***18*** | 01/08/10 | gbeane | Updated for 1.3.0 |

About this document

Document Purpose

This document describes how to compile and run the hmmSNP program. If you have any suggestions or comments on this user guide please send them to glen.beane@jax.org.

Table of contents

1 Notes on Algorithm 2

2 Current Version 2

3 Platform 2

4 Compiling and Installation 2

5 Running the program 2

5.1 Usage Summary 2

5.2 Input File Formats 2

5.2.1 Single .csv File Format 2

5.2.2 Multi-File Format (JSNP format) 3

5.2.3 Adding new strains to processed input files 4

5.3 Command Line Options 4

5.4 RIL 6

5.5 Non-nuclear DNA or specifying any chromosome 6

5.6 The –no\_em and –sort\_only options 6

5.7 Example command line 7

5.8 Output Files 7

6 Generating Graphs with Graphviz 8

7 Perl Tools 8

7.1 strain\_trace.pl 9

7.2 graph\_subset.pl 9

8 Running on clusters 9

9 Future modifications 10

9. Reference 10

# Notes on Algorithm

Jin Szatkiewicz designed the algorithm under the supervision of Gary Churchill; and a Matlab version of the algorithm is available from Jin. This document describes the use of the C implementation of Jin’s algorithm. Glen Beane programmed the C implementation with assistance from Jin. For details see Szatkiewicz et al. “An imputed genotype resource for the laboratory mouse”.

# Current Version

The most up to date version of hmmSNP is version 1.2.3. This user guide describes the features of this version of the program.

# Platform

This program has been developed for use in a Unix-like environment, and has been tested on Apple OS X and Linux. It may run on Microsoft Windows through the use of Cygwin (a POSIX environment for Windows), however this is untested and currently not supported.

# Compiling and Installation

As of hmmSNP version 1.2.3 we are now using the GNU build system. Consult the README and INSTALL documents included in the source distribution to find out how to compile and install the program.

# Running the program

## Usage Summary

hmmSNP <options> [input data file]

## The Input File

The input file is a single comma-delimited text file, which may contain SNP data for multiple chromosomes. The file is expected have the column headers as the first line. By default it is assumed the file will contain 4 annotation columns, followed by the sequence columns. One of the 4 annotation columns must contain the chromosome number (the order is unimportant). The user may indicate that their input file contains a different number of annotation columns, see section for details. The chromosome column is identified by a column header starting with the characters “chr” (this string is case insensitive). Text preceding the chromosome number will be ignored, so the program will correctly parse SNPs with chr1 or 1 in their chromosome column as chromosome 1. One of the four annotation columns should be the snpid, but this is not required. Including the SNP id lets hmmSNP identify bad SNPs to the user.

Example data in this format:

snpid,Source,bp,chromosome,129S1/SvImJ,129X1/SvJ,A/J,C57BL/6J,DBA/2J

rs31953890,Celera2,3000296,chr1,N,N,A,A,G

rs32186899,Celera2,3000331,chr1,N,N,T,T,C

rs31079645,Celera2,3000346,chr1,N,N,G,G,A

rs30948750,Celera2,3001206,chr1,N,G,T,T,G

rs32647232,Celera2,3002022,chr1,N,A,G,G,N

rs30617825,Celera2,3002583,chr1,A,A,G,G,A

rs31471951,Celera2,3002632,chr1,G,G,A,A,G

### Adding new strains to processed input files

It is possible to add new strains to input files that have already been processed and then use the saved model to determine the haplotype path and impute missing genotypes for all strains without having to rerun the em training.

Add the additional columns to the original input file. Then run hmmSNP with the –no\_em, --lambda\_outp\_file, and –training\_strains options with the proper arguments (see Section ). Note that you must make sure that adding the additional strain columns does not cause a SNP to be removed by the cleaning process if that SNP was used to generate the saved model, or that adding the strains does not cause a SNP that was previously cleaned to be included. This would cause the cleaned SNPs to align improperly with the imported transition and emission matrixes, leading to a crash or incorrect results.

### Data Cleaning

The program will perform a number of automatic data cleaning steps, some of which can be controlled or affected by some command line options.

First, any non A,G,T,C values are replaced with ‘N’. Then the following rules are used, in order, to filter out “bad” SNPs.

1) Remove SNP if all strains are ‘N’.

2) Remove SNP if more than two non–N values are observed.

3) Remove SNP if missing percentage is too high (see –maxmiss in section ).

4) Non Recombinant inbred line data:

4a) Remove SNP if only one non-N type observed (if –remove\_constant\_snp command line option was used)

5) 2-way recombinant inbred line data:

5a) Remove SNP if both parents are missing

5b) Remove SNP if parents are the same type (constant SNP)

5c) Remove SNP if one parent is missing and its type can not be “guessed”\*

\*If one parent is missing in a 2-way RIL data set and there is a second observed type in the derived strains then we will assume that this is the type for the missing parent since the derived strains will have the same type as one of the two parents.

Passing the option –warn\_snp\_removed allows users to see a warning message printed to stdout for each SNP removed. See section for more information about command line options.

## Command Line Options

**General Parameters:**

--**c**hromosome (-c): specifies which chromosome to process (typically 1-19,x,y). Required.

--no\_em: do not run the EM training algorithm and use saved model parameters (requires

--lambda\_outp\_file)

--**g**raphviz (-g): turns on output of graphviz file

--**m**axmiss (-m): maximum allowable percentage of missing nucleotides per SNP. (Default 0.8)

--pre**f**ix (-f): prefix for output files. (Default chr<chromosome>\_)

--**v**ersion (-v): print version number and exit

--help (-u): usage information and exit

--remove\_constant\_snps: remove any SNP whose observed genotypes are same among all sequences before running HMM

--warn\_snp\_removed: print a warning message about SNPs that are removed by the data cleaning process. The message gives the rule that was used to remove the SNP and the SNP ID (if the SNP ID column can be determined)

--annotation\_columns: number of annotation columns in csv file (default 4)

--confidence\_threshold: minimum confidence score used when creating the filled\_filtered.csv files. See Section 5.7 for more information. (default 0.6)

**E-M Parameters:**

--miss**o**pt (-o): Missing Option.

c (or C) complete data, r (or R) missing as random (default), e (or E) missing as a type of emission

--parents (-p): number of parent strains for RIL data. Currently only 2-way RILs supported

--pseudo\_option: pseudocount. Default 0.1, which means adding 1/10th of sequence.

--pseudo\_mod: when contributing the pseudocount to the emission matrix we first multiply it by the pseudomod. Default 1.0

--**h**aplotypes (-h): number of haplotypes. Default 3.

--**s**top (-s): stop option.

c (or C): convergence. Stop once convergence criterion reached OR at max iterations (default)

i (or I): max iterations.

--**t**olerance (-t): convergence tolerance (Default 10^-6 in the change of the log-likelihood).

--**i**terations (-i): max iterations. Default 10,000.

--**l**ambda\_ratio (-l): Ranges between 0 and 1. Default = 0.96.

For state transitions, the prior density of transition probabilities is biased towards the transitions between the same haplotype, with probability lambda, and is equally distributed among the other (*H*-1) haplotypes

--lambda\_outp\_file (-F): existing lambda/output file to read in for initial values

--random\_starts (-R): number of random starts for lambda and output matrix initialization. (default = 10)

--random\_iterations (-I): number of iterations to run hmm when evaluating random initialization (default = 100)

--training\_strains (-Y): comma-delimited list of strain columns to use for training strains. Ranges of numbers are allowed. Example: 1,2,3,4,5,10-15 (by default all strains are used for training). If this option is used with –no\_em then the training strains are used to determine major and minor allele and then the haplotype path is determined for each strain using a saved model and the missing genotypes are imputed for all strains. If this option is used with em training, then only the training strains are used by the expectation-maximization function and the haplotype path is determined for all strains and missing genotypes are imputed for all strains.

--no\_fix\_outp: the outp (emission matrix) is normally fixed for RIL data. This option allows outp to be updated during the mstep of the hmm\_em() function.

--partial\_output: specify a number of iterations between writing “partial” lambda and outp values. For example, if the user specifies –partial\_output 100 then during the hmm\_em function the lamboutp file (file containing both transition and emission matrixes) will be written every 100 iterations. After the file is successfully written to disk the previous lamboutp file will be deleted. After final results are written to disk the last partial results file will be deleted. These partial results files end in iteration-N.csv where N is the iteration it was written. (default = 0, no partial results files)

--keep\_partial: keep partial output files (see above)

**Path Options:**

--path (-P): (v)iterbi, (s)mooth, or (b)oth. (Default v)

--smthout: print final smooth values (maxsmth path only)

**Pruning Options:**

**NOTE: All pruning options described below are currently disabled since the pruning algorithm is not working at this time.**

--prune1 [optional prune rule], prune if state marginal probability is less than or equal to prune rule. (Default prune rule = 0.05)

--prune2 [optional prune rule], prune if state marginal probability is less than or equal to prune rule divided by number of strains. (Default prune rule = 0.5)

--tolerance\_relaxed (-r): relaxed tolerance used for pruning hmm models (Default 0.0001 in the change of the log likelihood.)

**Sorting Options:**

--sort (-S): sort output

--sort\_only: when passed with –lambda\_outp\_file skip hmm and sort using lambda and outp from file.

## RIL

hmmSNP supports two-way RIL data sets. To process RIL data, pass the –parents 2 argument to hmmSNP. This tells the program to use the first two strain columns as the parent types. In this case hmmSNP will fix the emission matrix after initialization. If you wish to have the emission matrix updated during the M-step then you must pass the –nofixoutp option.

## Non-nuclear DNA or specifying any chromosome

hmmSNP was designed to process input files containing mouse SNP data, so we support these values for the chromosome: 1-19,x,y. However, some users want to use hmmSNP on non-nuclear DNA. Because of a specific request we also support an additional argument to the –chromosome option: “m” (for mitochondrial DNA). This value has no meaning to hmmSNP other than for searching for SNPs in the input file that mach the specified chromosome or for naming output files. hmmSNP will do a case insensitive comparison when it looks at the value of the chromosome column in a input file, so “m’”or “M” are equivalent.

## The –no\_em and –sort\_only options

As stated above, the –-no\_em command line option allows you run the viterbi algorithm (by default) alone based on the saved model to impute missing data for new strains. The -–lambda\_outp\_file is required. Note that the dimensions of the parameter file and the input data need to be the same (after the cleaning step). Section 5.6 includes an example of the use of –no\_em.

Note that --sort\_only is equivalent to –-sort -–no\_em, except that –sort\_only will not write the unsorted path or graph output files (it will only create the sorted versions of those files)..

## Example command lines

This following command will run hmmSNP on chromosome 19, with a tolerance of 0.0001, on the .csv file formatted as described in section 5.2.1. ***Any other parameter not specified will use the default value listed in its description above.***

hmmSNP –c 19 –t 0.0001 GNFSNPs\_Build36\_letters.csv

The following command will run hmmSNP on chromosome 1, with a missoption of 0.9, a maximum of 7000 iterations, and we will compute both the viterbi and the maxsmth path. This example shows a multi-file format described in section 5.2.2.

hmmSNP –c 1 –m 0.9 – i 7000 –-path both Perlegen36

Here is an example showing existing lambda/output file to read in for initial values

hmmSNP –c 19 –lambda\_outp\_file \

chr19\_lamboutp\_saved.csv \

GNFSNPs\_Build36\_letters.csv

Here is an example showing the –-no\_em option in use, ie.. it runs the viterbi algorithm only. In this example the user is also running the –graphviz option to generate a .dot file.

hmmSNP –c 19 –h 6 -–no\_em -–lambda\_outp\_file \ chr19\_lamboutp\_saved.csv —graphviz \ GNFSNPs\_Build36\_letters.csv

Here is an example showing how a RIL dataset may be processed:

hmmSNP –c 1 –parents 2 –graphviz chr1\_RIL.csv

## Output Files

When the program is run it will produce a number of output files. Note that the default output filename prefix is “chrN\_” where N is 1..19,X,Y.

**Group 1**: imputed genotypes, confidence scores, viterbi/smth path files

* <prefix>viterbi\_path.csv: comma delimited file containing haplotype path for each sequence, computed with the Viterbi algorithm. This file is not created if the viterbi algorithm is not run.
* <prefix>viterbi\_path\_sorted.csv: if user is running with a sorting option then this file will be created with the sorted path.
* <prefix>viterbi\_filled.csv: this is a comma delimited file containing the data imputed using the Viterbi path. This file is created whenever the Viterbi path has been computed.
* <prefix>ms\_path.csv: comma delimited file containing haplotype path for each sequence, computed by the maxsmth algorithm.
* <prefix>ms\_filled.csv: a comma delimted file containing the data imputed using the maxsmth path. This is created whenever the maxsmth path has been computed.
* <prefix><sorting\_algorithm>\_filled\_filtered.csv: filled data with any imputed genotype with a confidence score lower than the confidence threshold (default 0.6) replaced with an N.
* <prefix><sorting\_algorithm>\_filled\_filtered\_sorted.csv: sorted filled data with any inputed genotypes with a confidence score lower than the confidence threshold (default 0.6) replaced with an N.
* <prefix><path\_algorithm>\_filling\_prob1.csv: filling probability, i.e. confidence score
* <previx><path\_algorithm>\_filling\_prob2\_sorted.csv
* <prefix><path\_algorithm>\_filling\_prob2.csv: filling probability of filled in only
* <previx><path\_algorithm>\_filling\_prob1\_sorted.csv

**Group 2**: parameter files

* <prefix>lamboutp.csv: a comma delimited file containing the lambda and outp values from EM. In this file each SNP occupies H individual lines, where H is the number of haplotypes used for the model. The first two columns are the SNP ID followed by the state ID. After the annotation columns will be H number of lambda columns, so we essentially have an H by H lambda transition matrix for this state. Following the lambda columns there will be two or three Outp columns (2 for random missing, 3 for missing as emission). The final column is the marginal state probability.
* <prefix>lamboutp\_sorted.csv: if user is sorting then the sorted lambda and output parameters will be saved in this file
* <prefix>graph.dot: graphical representation of the model (lamboutp\* file) in dot language. A graphviz file produced if user passes –graphviz.
* <prefix>lamboutp\_init.csv: with random initialization, then final initial values are
* saved in this file.

**Group 3**: log likelihood files, other chosen outputs

* <prefix>loglike.txt: a text file containing 5 tab delimited columns: iteration number, loglikelihood, convergence(data), logsumprior, convergence(data+prior)
* <prefix>smth\_<seq\_num>.csv: If the user passes the –smthout option, a comma delimited file will be created containing the final smth values for each sequence.

If one is using a pruning option, several of these files are generated for each of the pruning hmm models (<prefix>lamboutp.csv, <prefix>loglike.txt).

# Generating Graphs with Graphviz

The <prefix>graph.dot file is a specially formatted text file containing nodes and edges for a state trasition graph. The free program Graphviz can be used to create graphs in a variety of formats. To generate a jpeg graph from the command line (on Unix-like operating systems), run the following command (assuming the command line version of Graphviz has been installed on your system):

dot –T jpeg –o outfile.jpeg <prefix>graph.dot

Note that for very large graphs, you may need to open up the .dot file with a text editor and modify the size= line. This line sets the bounding box for the graph in inches. The default value hmmSNP uses is 100 by 5. For very large graphs that may scale the graph down so much that it is very distorted. You may need to experiment with this parameter.

GUI versions of Graphviz are available for Windows and Mac OS X at no cost.

We have created several perl scripts that are useful for manipulating the .dot files produced by hmmSNP. See Section 7 for more information.

# Perl Tools

We include some perl scripts that we find helpful. Currently there are two such perl scripts included with hmmSNP, and they both are used to manipulate the .dot file generated when hmmSNP is run with the –graphviz option.

## strain\_trace.pl

This perl script will trace the path of one or more strains through the graph.

Usage: strain\_trace.pl -h <num\_haplotypes> -g <graph.dot> \

-p <path.csv> -o <out.dot> [-n] [-e] <strain nums>

Inputs: -h - number of haplotypes used in hmm

-g - input graphviz file

-p - path file from hmmSNP

-o - path to output file (will be created)

-n - keep original graphs node coloring

-e – keep original edge coloring

<strain nums> space separated list of strain numbers

to trace. Strain numbers start at 1.

All arguments except –n and –e are required. If you intend to color the strain path and subset the graph, the path coloring should be done with strain\_trace.pl before running graph\_subset.pl

## graph\_subset.pl

This perl script will create a new .dot file with a subset of the SNPs from the original .dot file.

Usage: graph\_subset.pl -h <num\_haplotypes> -s <start\_SNP> \

-e <end\_SNP> -I inFile.dot –o outFile.dot

Inputs: -h – number of haplotypes used in hmm

-s – starting SNP #

-e – end SNP #

-I – path to .dot file output from hmmSNP

-o – path to output file (will be created)

All options are required.

# Running on clusters

Although the current version of the program is serial (it does not contain any parallelism), it can still be useful to run it on HPC clusters or SMP systems. This allows a user to run multiple chromosomes through the program at the same time (since each chromosome is and independent job this is said to be “embarrassingly parallel”). For TJL users, one must create a simple “batch script” and then submit that script to the bath system through the “qsub” command.

Note that with large datasets, a hmmSNP run can take many days to complete. Some of the runs used to impute missing genotypes for the CGD SNP Database (<http://cgd.jax.org/cgdsnpdb/>) took over a week to complete.

Here is an example batch script for running hmmSNP on an HPC cluster at TJL:

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=72:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR -h 4 -s c -t 0.000001 perlegen36.csv

This script will have to be modified slightly for your use, but basically this script tells the batch system that the job needs one processor for up to 72 hours. Once the job is executed, the script will cd into the directory that it was submitted from ($PBS\_O\_WORKDIR) and then it will execute. TJL users can find more information on the batch system at hpc.jax.org.

To submit this script, you would run the command “qsub –v CHR=N script\_name” in the directory containing the script and hmmSNP, substituting the appropriate chromosome for N. For example, to use the batch script above to run chromosome 19, you would execute the following qsub command:

qsub –v CHR=19 script\_name

The qsub command will return with a job identifier (e.g. 6033.cluster\_name). You can check the status of this job by running “qstat 6033” (substituting your job number for 6033) on the cluster head node. The command “qstat –f job\_number” will give more detailed information.

A running or queued job can be cancelled by running “qdel job\_number” on the cluster head node.

# Future modifications

We have a number of improvements planned for. This list below highlights the major enhancements we have planned.

* Checkpointing
* Parallel Processing

###### Program Flow

This is the basic flow of main():

parse command line options

open input file and read in data

clean data

if user passed parameter file

read in transition/emission matrixes

else

initialize transition/emission matrixes

write lamboutp\_init.csv to file

if (!sort\_only AND !no\_em)

initialize data structures for hmm\_em function

if (pruning)

set tolerance to tolerance\_relaxed

run hmm and prune until #pruned == 0

output lambda/outp

if (graphviz)

output graphviz file

reset tolerance

run hmm\_em

output transition/emission matrixes (lamboutp.csv)

if (graphviz)

output graphviz file

compute path

output path

impute missing genotypes using path

write filled data

if (sort)

sort

if (grapviz)

output graphviz file

compute sorted path

output sorted path

fill in using sorted path

writes sorted imputed data

###### Use Cases

The HMM method draws statistical power from both number of strains and number of SNP markers. Better performance is achieved when more information is utilized, e.g., more strains in each subpopulation, more informative markers, more observed genotypes. In general, imputation is accurate given ~100K SNP genotypes per genome. The choice of *H* is most critical which should reflect users’ assumption of maximum haplotype groups present in the dataset. The method is robust against the choice of transition and emission priors. Dirichlet pseudocount should always be used as regularizer.

In order to achieve the best predictive accuracy of the HMM, parameters can be optimized by sensitivity analysis. . If haplotype inference is of interests, experiments can be carried out by varying *H* and testing the performance of the subsequent HAM analysis. If the primary interest is imputation, experiments can be carried out by randomly masking portions of the genotype data and assessing accuracy of imputed genotypes. Based on these studies, we chose the following parameters for the imputation of each CGD SNP dataset. Note that questions regarding creation of CGD SNP datasets should be addressed directly to Yueming Ding (yueming.ding@jax.org).

**SNP\_v5.5\_70K dataset: 57 classical and wild-derived strains**

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=8:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR -h $5 -l 0.6 --pseudo\_option 0.01 --pseudo\_mod 1 \

-o r --maxmiss 1 --warn\_snp\_removed --random\_starts 10 \

—-annotation\_columns 7 \

/home/jinpengs/CGDsnps/datainw/chr${CHR}\_v5.5\_build37\_inw.csv

**SNP\_v5.5\_70K dataset : A, B6, AXB RILs**

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=2:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR --parents 2 --maxmiss 1 --warn\_snp\_removed\

/home/jinpengs/CGDsnps/dataAXB/chr${CHR}\_v5.5\_build37\_AXB.csv

**SNP\_v5.5\_70K dataset : B6, DBA2, BXD RILs**

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=2:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR --parents 2 --maxmiss 1 --warn\_snp\_removed\

/home/jinpengs/CGDsnps/dataBXD/chr${CHR}\_v5.5\_build37\_BXD.csv

**Broad SNPs [H=6, l=0.9]**

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=8:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR -h 6 -l 0.90 --pseudo\_option 0.01 --pseudo\_mod 1 \

-o r --maxmiss 1 --warn\_snp\_removed --random\_starts 10 \

~/Broad36\_SNP/data/chr${CHR}\_broad36.csv

**MuDiv 166 strain genotype data [H=6, l=0.9]**

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=150:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR –h 6 -l 0.90 --pseudo\_option 0.01 --pseudo\_mod 1 \

-o r --maxmiss 1 --warn\_snp\_removed --random\_starts 10 \

--annotation\_columns 5 \

/home/jinpengs/CGD\_MusDiv\_filtered/data/chr${CHR}.csv

**NIEHS-Broad merged data [H=6, l=0.99]**

#!/bin/bash

#PBS -l nodes=1:ppn=1,walltime=1000:00:00

cd $PBS\_O\_WORKDIR

hmmSNP -c $CHR -h 6 -l 0.99 --pseudo\_option 0.01 --pseudo\_mod 1 \

-o r --maxmiss 1 --warn\_snp\_removed --random\_starts 10 \

~/Perlegen\_Broad\_updated\_v2.1/data/chr${CHR}\_perlegen\_broad\_updated\_v2.1.csv

###### References

Szatkiewicz JP, Beane GL, Ding Y, Pardo-Manuel de Villena F, Churchill GA. (2008) An imputed genotype resource for the laboratory mouse. Mamm Genome. 19(3):199-208

Use Hidden Markov Model to Infer Haplotype Structure and Impute Missing Genotypes from Mouse SNPs. [Poster]

An imputed genotype resource for the laboratory mouse [Talk]