FineSTRUCTURE and ChromoPainter v2 Manual

Daniel John Lawson*
dan.lawson@bristol.ac.uk

August 31, 2015

About

This is the manual for FineSTRUCTURE Version 2.0.6.pre . FineSTRUCTURE is software to perform population assignment using *large numbers of densely sampled* genomes, including both SNP chips and sequence data. This version greatly simplifies its use, removing the complex pipeline and allowing very simple use for small datasets, and reasonably simple integration with High Performance Computing (HPC) machines.

See `www.paintmychromosomes.com` for the most up to date information. The correct reference is:

- Inference of population structure using dense haplotype data, Daniel Lawson, Garrett Hellenthal, Simon Myers, and Daniel Falush, 2012. PLoS Genetics, Vol. 8(1): e1002453,

which contains the motivation and justification behind the method. Similar in concept to STRUCTURE, fineSTRUCTURE assigns individuals to populations using a model for the expected variability. The advantage of our approach is that very large numbers of SNPs (Single Nucleotide Polymorphisms) can be used and linkage disequilibrium can be efficiently exploited. To achieve this the computation is split into a **painting** step and a **population inference** step.

This software is **currently only available for Linux** and Unix compatible operating systems (such as Mac). To use it with Windows you will need cygwin. Version 0 is available for Windows, but its use is strongly discouraged due to the inherent difficulty of running a genomics pipeline via graphical interfaces. The version described in this manual is command-line only; Refer to the Version 0 manual for information about the GUI.

This software is Beta: please report all bugs to the author.

Contents

1 Changes from Version 0 **2**

2 How to use this software **3**

3 Overview of FineSTRUCTURE and the help system **4**
3.1 Information about the processing pipeline 4
3.2 Basic Help for 'automatic mode' 4

*Integrative Epidemiology Unit, School of Social and Community Medicine, and Department of Statistics, University of Bristol, UK

4 Detailed help **5**
4.1 Information on Input formats 5
4.2 Help on how the computation is performed 7
4.3 Help on specific parameters 9
4.4 Accessing FineSTRUCTURE, ChromoCombine and ChromoPainter directly 9
4.5 List of all parameters 9

5 ChromoPainter **11**

6 ChromoCombine **12**

7 FineSTRUCTURE **13**

8 Computational considerations **16**

9 Greedy finestructure **16**

10 Job submission in qsub and related environments **18**

11 Provided scripts **18**
11.1 makeuniformrecfile.pl 18
11.2 convertrecfile.pl 19
11.3 chromopainter2chromopainterv2.pl 19
11.4 phasescreen.pl 20
11.5 phasesubsample.pl 20
11.6 plink2chromopainter.pl (PLINK) 20
11.7 impute2chromopainter.pl (SHAPEIT format) 21
11.8 beagle2chromopainter.pl (BEAGLE format) 22
11.9 msms2cp.pl (MSMS and MS output format) 23

12 Potential pitfalls **23**

13 Additional comments **24**

1 Changes from Version 0

Everything that was possible with Version 0 is still possible now. The main difference is that we now provide a single interface, 'fs', to access all functions and to make managing the computational pipeline much easier. Specifically, this code incorporates:

- **ChromoPainter**: Taking in phased sequence data, this 'paints' each haplotype using the others.
- **ChromoCombine**: This combines the output of ChromoPainter into a few files summarizing the genome-wide sharing of haplotypes between all individuals. The main output of chromocombibe is the 'coancestry matrix'.
- **FineSTRUCTURE**: Working with the coancestry matrix, we identify statistically indistinguishable individuals and cluster them.

We have added diagnostic tests for MCMC convergence to ensure that FineSTRUCTURE has been run long enough.

If you have only a small dataset, then you can **run the entire pipeline**, exploiting multiple processors, **using a single command**. If you have a large dataset and a HPC machine, it can instead provide command lines to be processed.

2 How to use this software

This software attempts to automate as much of the processing pipeline as possible. You need to start with *phased* data as output by either SHAPEIT, BEAGLE, IMPUTE2, etc. There are conversion scripts provided for each of these, described in Section 11. We don't want to make any recommendations, but most people use SHAPEIT. The others may be better depending on your circumstances.

Running FineSTRUCTURE for small datasets is now extremely easy. If you have both a modest number of individuals (less than around 200) and SNPS (100K) you can run the whole pipeline on a single machine (exploiting multiple cores, if you have them). Running the entire pipeline could be as simple as:

Listing 1: Simple example)

```
> fs example.cp -idfile data.ids -phasefiles data.phase -recombfiles data.
    recombfile -go
```

where we have specified 5 things:

- `example.cp`: This is the file where the results and intermediate quantities are stored. A directory called 'example' will be created to store intermediate files.
- `-idfile data.ids`: This defines the names of each individual in the data, one per row.
- `-phasefiles data.phase`: This contains the PHASE format data.
- `-recombfiles data.recombfile`: This contains the linkage information about the genetic distance between the SNPs specified in the phase data.
- `-go`: fs will figure out what needs to be done and in what order. It will then (in this example) run the entire pipeline, including ensuring that the MCMC has been run long enough.

Converting or writing idfiles, phase files and recombination files are described in Section 4.1 with conversion scripts in Section 11.

Running FineSTRUCTURE for larger datasets is more difficult, because we assume that users will want to exploit High Performance Computing (HPC) resources. We therefore split the computation into a number of stages, each of which can be run on a cluster. A text file is generated containing the commands, 1 per line. You will be prompted with the location of this file. The process becomes:

Listing 2: HPC example)

```
> fs example.cp -idfile data.ids -phasefiles data.phase -recombfiles data.
    recombfile -hpc 1 -go
> qsub_run.sh -f example_commandfile1.txt # and wait for it to execute
> fs example.fs -go
> qsub_run.sh -f example_commandfile2.txt # and wait for it to execute
> fs example.fs -go
> qsub_run.sh -f example_commandfile3.txt # and wait for it to execute
> fs example.fs -go
> qsub_run.sh -f example_commandfile4.txt # and wait for it to execute
> fs example.fs -go
```

Because there are things that can go wrong in each processing step, and rerunning has an overhead in this approach, it is more important to get the parameters right in advance in HPC mode. See 'Potential pitfalls' (Section 12) to get these right first time.

You are STRONGLY ENCOURAGED to go through the provided example, to get a feeling for how this works in practice, to see how to set various important parameters, and to cover some basic problems that you might encounter.

3 Overview of FineSTRUCTURE and the help system

The program includes inline help which you can access from the command line. These are reproduced here verbatim from the current version of the program. All of the help in this document can be accessed via the command line.

3.1 Information about the processing pipeline

Listing 3: Overview help

```
> fs -h info
```

```
***** FineSTRUCTURE and ChromoPainter *****
USAGE: "fs <projectname>.cp <options> <actions>"
SUMMARY:
    * Creates <projectname>.cp and a directory <projectname>.
    * Performs inference using ChromoPainter and FineSTRUCTURE.
    * Can create commands to be run on an external HPC machine.
    * Supports parallel processing in a single machine, obtain finestructure
       results from a single command.
    * This is helper software to easily run chromopainter and finestructure, which
       are built in to this program.
OVERVIEW OF USE:
    * Provide SNP data with -phasefiles, recombination data with -recombfiles and
       individual data with -idfile
    * Configure any advanced parameters
    * And -go: fs will figure out the rest!
    * If using -hpc mode, run "fs" on the head node, which will not run the model.
       Instead four "stages" of
    processing are created for running on a external HPC system. Then rerun this
       program once they have completed
    to generate the next stage.
USE "fs -h" for help on the automatic mode.
```

3.2 Basic Help for 'automatic mode'

Listing 4: Basic Help

```
> fs -h
```

```
***** Help for fs - running the whole chromopainter/finestructure inference
   pipeline in 'automatic' mode *****
USAGE: "fs <projectname>.cp <options> <actions>"
GENERAL OPTIONS FOR "project" tool:
    -h/-help:    Show this help.
    -help info: Show 'overview' help explaining how this software works.
    -help actions: Show help for all actions.
    -help parameters: Show help for all parameters.
    -help <list of commands or parameters>: Show help on any specific commands or
       parameters.
    -help input: Show examples and give details of the input file formats.
    -help stages: Detailed description of what happens in, and between, each stage
       of the computation.
    -help tools: Show help on how to access the chromopainter/chromocombine/
       finestructure tools directly.
```

```
    -help example: Create and show help for a simple example.
    <tool> -h: Show help on a particular tool: one of fs,cp,combine. IMPORTANT NOTE
       : These have simplified interfaces
    with different names when running in automatic mode. The help for automatic
       mode parameters explains which
    parameters it changes.
    -v      :    Verbose mode
    -n      :    New settings file, overwriting any previous file
    -<parameter>:<value> : Sets the internal parameter, exactly as if they were
       read in from -import.
The colon is optional, unless <value> starts with a '-' symbol. Escape spaces and
   don't use quotes;
e.g. '-s1args:-in\ -iM'.

IMPORTANT PARAMETERS:
idfile : IDfile location, containing the labels of each individual. REQUIRED, no
   default (unless -createids is used).
phasefiles : Comma or space separated list of all 'phase' files containing the (
   phased) SNP details for each
haplotype. Required.
recombfiles : Comma or space separated list of all recombination map files
   containing the recombination distance
between SNPs. If provided, a linked analysis is performed. Otherwise an 'unlinked'
   analysis is performed. Note
that linkage is very important for dense markers!
IMPORTANT ACTIONS:
   -go : Do the next things that are necessary to get a complete set of
      finestructure runs.
   -import <file> : Import some settings from an external file. If you need to set
      any non-trivial settings,
   this is the way to do it. See "fs -hh" for more details.
   -createid <filename> : Create an ID file from a PROVIDED phase file. Individuals
       are labelled IND1-IND<N>.
```

4 Detailed help

4.1 Information on Input formats

See also the conversion scripts in Section 11.

Listing 5: Input help)

```
> fs -h input
```

```
#########################
HELP ON INPUT FORMATS.

This help, combined with looking at the example and the use of the provided scripts
    to convert your
data, should be enough for most users to get started.
NOTE: You can specify multiple phase and recombination files, one for each
   chromosome (at least, they are assumed
unlinked.) Specify via:
-phasefiles <list>.phase <of>.phase <files>.phase
with corresponding:
-recombfiles <list>.rec <of>.rec <files>.rec
```

```
##########################
IDFILE FORMAT:
This specifies the names of the individuals in the data, as well as (optionally)
   which population they are from
and whether they are included.
Format: N lines, one per individual, containing the following columns:
<NAME> <POPULATION> <INCLUSION>
    Where <NAME> and <POPULATION> are strings and <INCLUSION> is 1 to include an
       individual and 0 to exclude
    them. The second and third columns can be omitted (but the second must be
       present if the third is). Currently
    <POPULATION> is not used by this version of fs.
EXAMPLE IDFILE:
Ind1 Pop1 1
Ind2 Pop1 1
Ind3 Pop2 0
Ind4 Pop2 1
Ind5 Pop2 1

##########################
CHROMOPAINTER'S 'PHASE' FORMAT:
This is heavily based on 'FastPhase' output.
* The first line contains the number of *haplotypes* (i.e. for diploids, 2* the
   number of individuals).
* The second line contains the number of SNPs.
* The third line contains the letter P, followed by the basepair location of each
   SNP (space separated). These
must match the recombination file. Within each chromosome, basepairs must be in
   order.
* Each additional line contains a haplotype, in the order specified in the IDFILE.
   Diploids have two contiguous
rows. Each character (allowing no spaces!) represents a *biallelic* SNP. Accepted
   characters are 0,1,A,C,G,T,
with NO missing values!
EXAMPLE PHASEFILE:
10
6
P 100 200 300 400 500 600
010101
011101
111101
001101
011000
001100
001001
001011
001001
001111

##########################
CHROMOPAINTERS RECOMBINATION FILE FORMAT:
Required only if running in unlinked mode.
This specifies the distance between SNPs in 'recombination rate' units. There
   should be a header line followed by
```

```
one line for each SNP in haplotype infile. Each line should contain two columns,
    with the first column denoting
the basepair position values given in haplotype infile, in the same order. The
    second column should give the
genetic distance per basepair between the SNP at the position in the first column
    of the same row and the SNP
at the position in the first column of the subsequent row. The last row should have
     a '0' in the second column
(though this is not required     this value is simply ignored by the program).
    Genetic distance should be given
in Morgans, or at least the relevant output files assume this value is in Morgans.
If you are including genetic information from multiple chromosomes, put a '-9' (or
    any value < 0) next to the last
basepair position of the preceeding chromosome.
EXAMPLE RECOMBFILE:
start.pos recom.rate.perbp
100 0.01
200 0.02
300 -9
400 0.02
500 0.05
600 0

##########################

See the dedicated ChromoPainter v1 manual for more details.
```

4.2 Help on how the computation is performed

Listing 6: Help on what happens during each processing stage

```
> fs -h stages
```

```
***** Help on computational stages *****
The computation for finestructure is split into 4 main stages.
These are breakpoints at which we can export computation to a HPC machine. Before
    and after each, automatic mode
will do the work necessary to construct the next stage. This includes the
    construction of the command lines to be
executed; the command lines themselves are all that is run externally.
pre-stage<x>: performed by fs. A -reset <x> command will result in this being
    redone.
post-stage: performed by fs A -reset <x> command will use the output of the post-
    stage<x-1> processing. This means,
for example, that we can avoid needing to duplicate the chromopainter (stage2) runs
     in order create a duplicated
finestructure (stage3) run.
stage: Either '-dop<x>' meaning that the previously generated commands are run
    internally (in parallel) or
'-writep'<x>' meaning that they are written to file to be performed externally in
    HPC mode.
DETAILS:

==== pre-stage0 ====
#### stage0 ####
```

```
Data conversion. Currently not implemented!
==== post-stage0 ====
Action    -countdata : Ends stage0. Performs checks on the data and confirms that
   we have valid data.
==== pre-stage1 ====
Important note: stage1 is skipped when running in unlinked mode (no recombination
   file provided)
Action    -makes1 : Make the stage1 commands.
#### stage1 #### Chromopainter parameter inference
Action    -dos1 : Do the stage1 commands. This we should only be doing in single
   machine mode; we use -writes1 in
HPC mode.
Action    -writes1 <optional filename> : Write the stage1 commands to file, which
   we only need in HPC mode. In
single machine mode we can instead use -dos1.
==== post-stage1 ====
Action    -combines1 : Ends stage1 by combining the output of the stage1 commands.
   This means estimating the
parameters mu and Ne from the output of stage1.
==== pre-stage2 ====
Action    -makes2 : Make the stage2 commands.
#### stage2 #### Chromopainter painting
Action    -dos2 : Do the stage2 commands. This we should only be doing in single
   machine mode; we use -writes2 in
HPC mode.
Action    -writes2 <optional filename> : Write the stage2 commands to file, which
   we only need in HPC mode. In
single machine mode we can instead use -dos2.
==== post-stage2 ====
Action    -combines2 : Ends stage2 by combining the output of the stage2 commands.
   This means estimating 'c'
and creating the genome-wide chromopainter output for all individuals.
==== pre-stage3 ====
Action    -makes3 : Make the stage3 commands.
#### stage3 #### FineSTRUCTURE MCMC inference
Action    -dos3 : Do the stage3 commands. This we should only be doing in single
   machine mode; we use -writes3 in
HPC mode.
Action    -writes3 <optional filename> : Write the stage3 commands to file, which
   we only need in HPC mode. In
single machine mode we can instead use -dos3.
==== post-stage3 ====
Action    -combines3 : Ends stage3 by checking the output of the stage3 commands.
==== pre-stage4 ====
Action    -makes4 : Make the stage4 commands.
#### stage4 #### FineSTRUCTURE tree inference
Action    -dos4 : Do the stage4 commands. This we should only be doing in single
   machine mode; we use -writes4 in
HPC mode.
Action    -writes4 <optional filename> : Write the stage4 commands to file, which
   we only need in HPC mode. In
single machine mode we can instead use -dos4.
==== post-stage4 ====
Action    -combines4 : Ends stage4 by checking the output of the stage4 commands.
Not a command, but if -go gets here, we will provide the GUI command line for
   visualising and exploring the results.
```

4.3 Help on specific parameters

Help on specific commands or parameters is obtained by invoking help with the name as the argument. See Section 4.5 for obtaining a list of all parameters.

Listing 7: Example for accessing the help about a parameter

```
> fs -h s1indfrac
```

```
Help for Parameter s1indfrac : fraction of individuals to use for EM estimation. (
    default: 1.0)
```

4.4 Accessing FineSTRUCTURE, ChromoCombine and ChromoPainter directly

Listing 8: Help on accessing the tools

```
> fs -h tools
```

```
***** Help for tool mode *****
USAGE: fs [tool] [OPTIONS]
Using this interface you can access any of the advanced functionality of
    chromopainter and finestructure.
     [tool] can be any of:
     <projectname>.cp: automatic mode - creates and runs the commands below for you,
         organising what to do in a
'project'. This should be the default unless you know what you are doing.
     cp: chromopainter mode (commands exactly as chromopainter.) This can be used to
         perform more sophisticated
     analyses, such as admixture modelling via GLOBETROTTER.
     combine: chromocombine mode (commands exactly as chromocombine)
     fs: finestucture mode (commands exactly as finestructure)

     Run "fs [tool] -h" to obtain more detailed help on cp, combine, or fs tools.
     Run "fs -h" to get the automatic mode help.
```

4.5 List of all parameters

Listing 9: Help on all parameters that can be set

```
> fs -h parameters
```

```
Help for Parameter validatedoutput : Derived. Whether we have validated output from
     each stage of the analysis (0-4)
Help for Parameter exec : Finestructure command line. Set this to be able to use a
    specific version of this
software. (default: fs)
Help for Parameter hpc : THIS IS IMPORTANT FOR BIG DATASETS! Set hpc mode. 0:
    Commands are run 'inline' (see
'numthreads' to control how many CPU's to use). 1: Stop computation for an external
     batch process, creating a file
containing commands to generate the results of each stage. (default: 0.)
Help for Parameter numthreads : Maximum parallel threads in 'hpc=0' mode. Default:
    0, meaning all available CPUs.
Help for Parameter ploidy : Haplotypes per individual. =1 if haploid, 2 if diploid.
     (default: 2)
```

```
Help for Parameter linkagemode : unlinked/linked. Whether we use the linked model.
    default: unlinked / linked if
recombination files provided.
Help for Parameter indsperproc : Desired number of individuals per process (default
    : 0, meaning autocalculate: use
1 In HPC mode, ceiling(N/numthreads) otherwise. Try to choose it such that you get
    a sensible number of commands
compared to the number of cores you have available.
Help for Parameter outputlogfiles : 1=Commands are written to file with redirection
     to log files. 0: no
redirection. (default:1)
Help for Parameter s12inputtype : What type of data input (currently only "phase"
    supported)
Help for Parameter idfile : IDfile location, containing the labels of each
    individual. REQUIRED, no default
(unless -createids is used).
Help for Parameter s12args : arguments to be passed to Chromopainter (default:
    empty)
Help for Parameter ninds : Derived. number of individuals observed in the idfile
Help for Parameter nindsUsed : Derived. number of individuals retained for
    processing from the idfile
Help for Parameter nsnps : Derived. number of SNPs in total, over all files
Help for Parameter s1args : Arguments passed to stage1 (default:-in -iM --
    emfilesonly)
Help for Parameter s1emits : Number of EM iterations (chromopainter -i <n>, default
    : 10)
Help for Parameter s1minsnps : Minimum number of SNPs for EM estimation (for
    chromopainter -e, default: 10000)
Help for Parameter s1snpfrac : fraction of genome to use for EM estimation. (
    default: 0.1)
Help for Parameter s1indfrac : fraction of individuals to use for EM estimation. (
    default: 1.0)
Help for Parameter s1outputroot : output file for stage 1 (default is
    autoconstructed from filename)
Help for Parameter Neinf : Derived. Inferred 'Effective population size Ne' (
    chromopainter -n).
Help for Parameter muinf : Derived. Inferred Mutation rate mu (chromopainter -M)
Help for Parameter s2chunksperregion : number of chunks in a "region" (-ve: use
    default of 100 for linked, nsnps/100
for unlinked)
Help for Parameter s2args : Additional arguments for stage 2 (default: none, "")
Help for Parameter s2outputroot : Output file name for stage 2 (default:
    autoconstructed).
Help for Parameter cval : Derived. 'c' as inferred using chromopainter. This is
    only used for sanity checking. See
s34 args for setting it manually.
Help for Parameter cproot : The name of the final chromopainter output. (Default: <
    filename>, the project file name)
Help for Parameter cpchunkcounts : the finestructure input file, derived name of
    the chunkcounts file from cproot.
Help for Parameter fsroot : The name of the finestructure output (Default: <
    filename>, the project file name).
Help for Parameter s34args : Additional arguments to both finestructure mcmc and
    tree steps. Add "-c <val>" to
manually override 'c'.
```

```
Help for Parameter s3iters : Number of TOTAL iterations to use for MCMC. By default
    we assign half to burnin and
half to sampling. (default: 100000)
Help for Parameter s3iterssample : Number of iterations to use for MCMC (default: -
    ve, meaning derive from s3iters)
Help for Parameter s3itersburnin : Number of iterations to use for MCMC burnin (
    default: -ve, meaning derive
from s3iters)
Help for Parameter numskip : Number of mcmc iterations per retained sample; (
    default: -ve, meaning derive from
maxretained)
Help for Parameter maxretained : Maximum number of samples to retain when numskip -
    ve. (default: 500)
Help for Parameter nummcmcruns : Number of *independent* mcmc runs. (default: 2)
Help for Parameter fsmcmcoutput : Filename to use for mcmc output (default:
    autogenerated)
Help for Parameter mcmcGR : Derived. Gelman-Rubin diagnostics obtained from
    combining MCMC runs, for log-posterior,
K,log-beta,delta,f respectively
Help for Parameter threshGR : Threshold for the Gelman-Rubin statistic to allow
    moving on to the tree building
stage. (Default: 1.3)
Help for Parameter s4iters : Number of maximization steps when finding the best
    state from which the tree is
built. (default: 100000)
Help for Parameter fstreeoutput : Filename to use for finestructure tree output. (
    default: autogenerated)
Help for Parameter phasefiles : Comma or space separated list of all 'phase' files
    containing the (phased) SNP
details for each haplotype. Required.
Help for Parameter recombfiles : Comma or space separated list of all recombination
    map files containing the
recombination distance between SNPs. If provided, a linked analysis is performed.
    Otherwise an 'unlinked' analysis
is performed. Note that linkage is very important for dense markers!
Help for Parameter nsnpsvec : Derived. Comma separated list of the number of SNPs
    in each phase file.
Help for Parameter s1outputrootvec : Derived. Comma separated list of the stage 1
    output files names.
Help for Parameter s2outputrootvec : Derived. Comma separated list of the stage 2
    output files names.
Help for Parameter fsmcmcoutputvec : Derived. Comma separated list of the stage 3
    output files names.
Help for Parameter old_fsmcmcoutputvec : Derived. Comma separated list of the stage
    3 output files names, if we
need to continue a too-short MCMC run.
Help for Parameter fstreeoutputvec : Derived. Comma separated list of the stage 4
    output files names.
Help for Parameter stage : Derived. Don't mess with this! The internal measure of
    which stage of processing we've
reached. Change it via -reset or -duplicate.
```

5 ChromoPainter

IMPORTANT NOTE: this version of ChromPainter temporarily does not support donor files!

Listing 10: ChromoPainter help

```
> fs -h cp
```

6 ChromoCombine

Listing 11: ChromoCombine help

```
> fs -h combine
```

```
Usage: chromocombine [OPTIONS] -o <outputfileroot> <listofinputfiles>
    Remember to set <outputfileroot>.
    Additionally, you must supply at least one inputfile.
    Inputfiles can be specified in the following forms:
      a) ChromoPainter root file name
      b) ChromoPainter *full* file names, with any combination of endings.
         Repeated roots will be kept only once.
    Note that you must not have changed the file name endings.
Options:
-o <outputfileroot> Set the output file root name (default: output)
-d                  Specify that the input is a directory, and that *all*
                    valid file roots ending in <ending> are of interest.
-l                  Specify that length matrices should be IGNORED.
-m                  Specify that mutation matrices should be IGNORED.
-f <forcefile>      Process a continental forcing file.  "c" will be
                    calculated for this file separately, and stored in it.
                    The other output files will not be changed by this.
                    Therefore, fineSTRUCTURE will use the correct "c" when
                    run on the dataset without a forcefile, as well as when
                    one is provided. You must however use the force file
                    on the dataset for which "c" was calculated.
-F <forceoutput>    By default, the force file is *updated* with the correct
                    value of c.  This instead writes a *new* force file that
                    contains the important sections (in case you want to use
                       the.
                    same force file on many datasets).
-e <ending>         Set the inputfile ending (default: ".out")
                    Remember the dot!
-E <ending>         Set the outputfile ending (default: ".out")
-C                  Instead of smallish regions, use complete genomic segments
   for
                    calculation of c.  (Currently only valid if you provide a
                    number of input files and that each has approximately the
                       same recombination distance)
-i <file>           The id file from ChromoPainter. This is needed to provide
   the correct column headers if
donorfiles have been used to exclude some individuals, but the individuals were
    themselves the populations.
-t                  Test: print the files that will be processed.
                    and additionally test that they exist.
-u                  Unsafe mode; when summing individuals from different files,
                    the default is to stop if some are seen more often than
                       others.
                    Use this option to override.
```

```
    -v                  Verbose mode.
    -h                  This help message.
```

7 FineSTRUCTURE

Listing 12: FineSTRUCTURE help

```
> fs -h fs
```

```
    Usage: finestructure [OPTIONS] datafile <initialpopfile> outputfile
        Datafile is a matrix of copy counts.
        initialpopfile (optional) is a population state e.g. an outputfile.
        outputfile is the destination.
    -m <method>         Method to use.  Default: oMCMC.
                        <method> is either MCMCwithTree, oMCMC (MCMC without tree),
                        SplitMerge, Tree, or a contraction of any.
    -I <x>              Initial number of populations.  <x> is either a number
                        or "n" for the number of individuals, or "l" for label
                           detected
                        populations.  Default is 1.
    -s <s>              Sets the RNG seed to s (>0)
    -i <i>              Ignores the first i lines of the input file
    -x <num>            Number of burn in iterations for MCMC method.
    -y <num>            Number of sample iterations for MCMC method.
    -z <num>            Thin interval in the output file, for MCMC method.
    -t <num>            Maximum number of tree comparisons for splitting/merging.
    -K                  Fix the number of populations to whatever you started with.
                        This would be set by '-I' or by an initial state file.
    -l <filename>       Specify the average copy length datafile.  -i,-X,-Y options
                        *preciding* this file will affect this read; you can set
                           different
                        options for the copy rate datafile by specifying these -i,-
                           X,-Y again
                        after the -l option.
    -u <datatype>       Use a data inference method; one of :
                        counts: use only the copy counts data. (default if -l not
                           specified)
                        lengths: use only the copy length data (still needs valid
                           counts data!)
                        totallengths: use the mean length of chunk sizes
                        all: use all data (careful: this may not be statistically
                           valid).
                        default: use counts and totallengths (default with -l
                           specified).
    -a <num>            Set alpha, the prior of the number of parameters
                        (default: 1.0).
    -c <num>            Set the likelihood correction factor: L_{used}=L^{1/<
       corfactor>}.
                        (default: 1.0)
    -B <model>          Choose a model for beta:
                        1/e/E:  Equipartition model of Pella and Masuda.
                        2/c/C:  Constant model.
                        4/o/O:  F model of Falush et al 2003 with a single
                           parameter
```

```
                              for all populations (default).
-b <num>(,<num>,..) Hyperparameters for ALL models, in the order COUNTS,LENGTHS
   ,MEANS.
                    COUNTS: *must* be included, even if count matrix not used!
                    For model 1, there are no parameters.
                    For model 2, set the prior of the distribution of
                    population sizes (each population has beta_i=<num>).
                    (default: 1.0).
                    For model 4, set the hyperprior of the distribution of
                    delta and F. Parameters are
                    (k_f,k_delta,theta_f,theta_delta) for the parameters of the
                    gamma distribution F~Gamma(k_f,theta_f),
                    and delta~Gamma(k_delta,theta_delta)
                    (default: -b 2,2,0.01,0.01).
                    LENGTHS: 8 parameters:
                    (k_alpha0,k_beta0,k_alpha,k_beta,beta_alpha0,beta_beta0,
                       beta_alpha,beta_beta)
                    MEANS: 6 parameters:
                    (k_betamu, k_alphamu, k_kappa, beta_alphamu,beta_betamu,
                       beta_kappa)
                    Set K parameters negative for fixed =|k|
                    e.g. when finding a tree given the mean parameters.
-M <modeltype>      Specify the type of inference model for chunk counts.
                    <modeltype> accept contractions and lower case, and can be:
                      1 or Finestructure: standard finestructure model (default
                         ).
                      2 or Normalised: Normalise data row and columns within a
                         population.
                      3 or MergeOnly: As 2, but only compare populations being
                         merged or split.
                      4 or Individual: Prior is placed on individual rows
                         instead of
                                      population rows. (slowest model).
-e <name>           Extract details from a state; can be (a unique contraction
   of):
                    beta: the parameter matrix
                    X: the copying data matrix for populations
                    X2: the normalised copying matrix
                    maxstate: maximum observed posterior probability state
                    meancoincidence: the mean coincidence matrix
                    merge<:value><:split>: create a merge(or split)
                      population from the mean coincidence.
                    range:<from>:<to> gets the iterations in the specified
                       range.
                    thin:<step>: thins the output by step.
                    probability: get the posterior probability of the data
                    given the conditions of the outputfile.
                    likelihood: samples the likelihood of the data given the
                       conditions
                    in the outputfile.
                    tree: extract the tree in newick format and print it to a
                       FOURTH file
-F <filename>       Fix the populations specified in the file.  They should be
   specified as
                    population format, i.e. PopA(ind1,ind2) would fix the data
                       rows ind1 and ind2
```

```
                    to always be in the same population (they form a 'super
                       individual')
                    called PopA. Continents are specified with a * before the
                       name, and are treated
                    specially in the tree building phase,  i.e. *ContA(ind1,
                       ind2).  Continents
                    are not merged with the rest of the tree.
-T <type>           When using a merge tree, initialisation can be set to the
   following:
                    1:      Use the initial state "as is".
                    2:      Perform merging to get to best posterior state.
                    3:      Perform full range of moves to to get to best
                       posterior state.
                            This is the default.  Set number of attempts with -
                               x <num>.
                    4:      As 1, but don't flatten maximum copy rates for the
                       main tree.
                    5:      As 2, but don't flatten maximum copy rates for the
                       main tree.
                    6:      As 3, but don't flatten maximum copy rates for the
                       main tree.
                    7:      As 1, but maximise hyperparameters between merges.
                    8:      As 2, but maximise hyperparameters between merges.
                    9:      As 3, but maximise hyperparameters between merges.
-o <name>           File containing a state to use for ordering, if not the
   main file.
-k <num>            Change the tree building algorithm.
                    0:      Discard all ordering and likelihood information (
                       default).
                    1:      Maintain ordering.
                    2:      Maintain ordering and likelihood.
-X                  Specifies that there are row names in the data (not
   necessary for
                    ChromoPainter or ChromoCombine style files.)
-Y                  Specifies that there are column names in the data file (as
   -X, not necessary.)
-v                  Verbose mode
-V                  Print Version info
-h                  This help message

Examples:
finestructure -X -Y -m omcmc -i 2 -B 4 -b 2,2,0.01,0.01 -s 1 -x 100000 -y
   100000
    -z 1000 datafile.csv resfile.xml
                    Infers population structure (-m omcmc) from datafile.csv
                       which
                    contains 2 irrelevent lines (-i 2) with row (-X) and column
                        (-Y)
                    names, using the F model with a global F and Delta (-B 4)
                       using
                    Gamma(2,0.01) distributions. 100000 burn in steps are used
                        (-x)
                    and 100000 further iterations are sampled (-y) keeping
                       every
                    1000th sample (-z).
finestructure -X -Y -i 2 -e min datafile.csv resfile.xml resmsfile.xml
```

```
                    Create a minimum state file from the MCMC output.
finestructure -X -Y -i 2 -m T -t 20000 -B 4 -b -0.003,-0.94,-1,-1
    datafile.csv resmsfile.xml restree.xml
                    Create a tree (-m T) from the minimum state using the
                        inferred
                    values of F (0.003) and Delta (0.94), allowing 20000 (-t
                        20000).
                    different trees to be examined per merge attempt (slow!).
                    Perform admixture (-m admixiture) MCMC using the minimum
                        state
                    and parameters found as above.
finestructure -X -Y -i 2 -e admixture datafile.csv resmsfile.xml admixstate.csv
                    Extract the admixture matrix Q for a state in csv format.
                        This
                    is useful for making comparisons to the observed admixture
                        matrix.
```

8 Computational considerations

The ChromoPainter step has computational cost proportional to N^2L where N is the number of individuals and L is the number of SNPs. This is parallelized automatically, so if you have a large enough compute cluster, the cost is NL. For guidance, $L = 88K$ and $N = 100$ takes 3130 seconds (50 minutes) on a 2010 laptop using a single CPU. $L = 88K$ and $N = 500$ takes 264000 seconds (3 days). $L = 800K$ and $N = 1000$ (HGDP scale dataset) required a week on a moderate scale cluster. $L = 10M$ (sequence data) for $N = 500$ requires a similar amount of compute. $N = 1500$ on sequence data is about as high as is reasonably manageable; up to $N = 3000$ is manageable for SNP chip data.

The big barrier to computation for sequence data is memory. The cost per parallel run is proportional to NL, which can run to several Gigabytes, preventing easy parallelization. We are addressing this, but for the meantime you may need to customize the provided qsub script to request an appropriate amount of memory per chromosome.

If you are attempting to work with a dataset at or above this scale, we do have methodology in development for this. PBWT painting (an approximate algorithm) is orders of magnitude faster, and we are developing low-memory, efficient versions of the ChromoPainter algorithm too. Contact us if you might be interested in joining the development of these algorithms.

Running FineSTRUCTURE is also a problem at this scale. It has run time independent of L, and has been run successfully (taking approx two weeks) for $N = 2000$. For larger runs we provide an optimization script (See scripts/finestructuregreedy.sh) which greedily searches for the maximum a-posteriori state. This typically gets stuck in a local mode but multiple independent runs find similar enough best states to be useful. Expect serious problems above $N = 10000$.

9 Greedy finestructure

We have created a simple bash script that uses the pre-existing finestructure commands to compute the MAP (maximum aposteriori) state estimation using greedy optimisation. This can be many times faster than performing full MCMC, and is suitable for very large datasets (it is probably your only option for 10000+ samples). ChromoPainter will have become a very significant cost by this point.

To use greedy optimization, you should:

1. Run ChromoPainter to obtain the *combined* coancestry matrix using `fs <filename>.cp <options> -comb`
2. Run `finestructuregreedy.sh <filename>.chunkcounts.out outputfile.xml` This uses the "tree building" step of finestructure by repeatedly:

- Attempting MCMC moves, accepting only if they increase the posterior probability.
- Checking after a certain amount of iterations whether any progress has been made.

IMPORTANT NOTE: The -x option controls how many steps are taken between each step. The default of 50000 may be moderately slow, but it does try hard to find a better state. If this is too low, the algorithm will terminate prematurely.

There is a danger of getting stuck in a local optima, and of failing to find a possible move that would increase the Posterior. However, empirically the approach does perform well enough for exploratory data analysis. Convergence is assessed simply by counting the number of populations (as it unlikely that adding then removing populations is possible).

Listing 13: finestructuregreedy

```
> finestructuregreedy.pl
```

```
ERROR: Require .xml file name ending for outfile ("" is invalid)
Usage: /home/dan/code/trunk/fs1/scripts/finestructuregreedy.sh: [-r] [-R] [-d] [-m
    value] [-x value] [-t value]
[-a value] [-f value] datafile outputfile
Essentials: datafile and outputfile
Important flags are -m and -x
  -m value: sets the number of repeated FineSTRUCTURE runs to perform before giving
      in. (default: 20)
  -x value: sets the number of FineSTRUCTURE iterations to perform per step (
     finestructure -x flag). (default: 50000)
  -t value: (finestructure -t flag). (default: t=100000000, i.e. effectively
     infinite. careful, this may be slow)
  -a value: finestructure flags to be passed to all runs, e.g. "-X -Y". Quotes
     essential! Usually not needed. (default:
  "")
  -f value: set the location of the finestructure executable (default:
     finestructure)
  -r: when set, temporary files are replaced. without this you can run more
     iterations by changing -m and -x
  -R: when set, the final tree file is deleted if present. Default is to not run.
  -d: perform a dry run but don't actually do anything. Useful to see the
     fineSTRUCTURE arguments sued in each step.
EXAMPLE: /home/dan/code/trunk/fs1/scripts/finestructuregreedy.sh -a "-X -Y\ -c 0.2"
     -m 4 -t 1000 -x 50000
test.chunkcounts.out testgreedy.xml
.. continued with: /home/dan/code/trunk/fs1/scripts/finestructuregreedy.sh -a "-X -
    Y\ -c 0.2" -m 10 -R -t 1000 -x
50000 test.chunkcounts.out testgreedy.xml

FineSTRUCTURE is in theory run until convergence; i.e. until successive greedy tree
     runs have the same tree.
You must therefore set "-x" large enough to find differences at each step.
With "-x" too small, early stopping is likely and a lower K will be found.
The tree is computed only once, at the end; intermediate trees are present but
    highly stochastic.
Set "-t" to some smaller value if you are worried you may find very many
    populations; you can always rerun the
final step.

You may ignore the two warnings:
WARNING!  NOT TESTING ALL <c> COMBINATIONS! (max 1)
```

```
WARNING! Cannot confirm data file is the same as the MCMC was run on!
The first is generated by each iteration, the second by all but the initial run.
```

10 Job submission in qsub and related environments

It is your own responsibility to correctly submit jobs to your HPC infrastructure. Because of the wide variety of configurations available, we cannot write a script that will be able to work with all or even a high fraction of such systems. If you have a way of doing this for every line in a text file, then use that.

However, we have provided a script that works on our institutional HPC machine using qsub. It may require minor or major modification to work with other systems - use it cautiously! It can be found in the scripts directory.

Listing 14: Qsub script for job submission

```
qsub_run.sh  <commandlist.txt>
```

This creates a

For other systems, you might want to consider the unix command 'split'. This can split the command list into e.g. parallel processing units. If you can run commands in batches of 8, then:

Listing 15: Qsub script for job submission

```
> split -l 8 example_commandfile1.txt example_commandfile1_split
```

generates files with names like `example_commandfile1_split<aa-...>` each containing 8 lines from the command file.

11 Provided scripts

These are provided in the 'scripts' directory. You will need to add this directory to your path, copy them to somewhere in your path, or specify absolute file locations.

The usual caveats should be followed; we try to make these scripts work, but if they don't then we aren't responsible! Try to fix the problem yourself and let the author know of the issue.

11.1 makeuniformrecfile.pl

Creates a recombination rate map for use with the linkage model of chromopainter. This is *essential* if you do not have a provided recombination map for your species! The map assumes a constant rate of recombination *per base*, not per SNP.

Listing 16: makeuniformrecfile

```
> makeuniformrecfile.pl
```

```
Usage ./makeuniformrecfile.pl <phasefile> <outputfile>
     <phasefile> is a valid chromopainter inputfile ending in .phase (in
        ChromoPainter v1 or v2 format)
     <outputfile> will be a recombination file usable with <phasefile> in
        ChromoPainter, nominally in Morgans/base.
The recombination rate is scaled to be approximately that in humans (0.1 Morgans/Mb
   ). Because of this, it will
NOT be usable directly and should only ever be used in conjunction with EM
   parameter estimation, which corrects
```

```
for the global amount of recombination. If you are working on non-humans or
   simulated data, you may experience
problems with EM estimation. The parameter may get stuck at a local mode where
   there is effectively infinite (or no)
recombination. In this case, you should specify the initial conditions of
   ChromoPainter to have a much smaller or
larger Ne (-n) value.
```

11.2 convertrecfile.pl

Converts between recombination map files, and can take a wide varienty of map formats and convert them into a suitable format for ChromoPainter. For example, the HapMap B37 data obtained from nih can be processed with "convertrecfile.pl -M hapmap", but any CDF or PDF style format is supported.

Listing 17: convertrecfile

```
> convertrecfile.pl
```

```
-----convertrecfile.pl, create recombination maps for phase files from other maps.
Copyright (C) 2014 Daniel Lawson (dan.lawson@bristol.ac.uk) licenced under GPLv3
This is free software with NO WARRANTY, you are free to distribute and modify; see
   http://www.gnu.org/licenses

Usage: ./convertrecfile.pl <MAJOR MODE> <options> phasefile inrecfile outputrecfile
phasefile is a valid chromopainter or chromopainter v2 inputfile ending in .phase
inrecfile is a recombination file specified in one of the formats specified in <
   mode>
outputrecfile will be a valid recombination file for use with ChromoPainter.
MAJOR MODES: specified with -M. (Shortest unambiguous mode option will work)
 -M: <val>:      Specify the major mode. <val> can be:
         hapmap: The hapmap format is specified as 4 columns: chromosome, Position(
            BP) Rate(cM/Mb) Map(cM)
                 This uses columns 2 and 4 to reconstruct the map.
         plain:  (default) Assumes that the data are specified in 2 columns,
            Position(BP) Rate(M/b)
                 This is the mode assumed chromopainter (note: the rate is Morgans
                    per base).
Other important options:
 -v:            Verbose mode.
 -h:             This help.
 -H:             Detailed help on the wide variety of different options, including
    different column
                 separators, different units, reading of Culmulative vs non-
                    culmulative distributions,
                 and handling maps that do not cover the full range of the SNPs.
EXAMPLE: ./convertrecfile.pl -M hap my_chr1.phase genetic_map_GRCh37_chr1.txt
   my_chr1.recombfile
```

11.3 chromopainter2chromopainterv2.pl

Convert old phase format datasets to the new format. Not that this is not strictly necessary, because fineSTRUCTURE can use either type, but the new format is much more sensible.

Listing 18: chromopainter2chromopainterv2

```
> chromopainter2chromopainterv2.pl
```

```
CONVERTS FROM CHROMOPAINTER v1 FORMAT TO v2
usage: perl chromopainter2chromopainterv2.pl <phasefile> <outputphasefile>
with:
<phasefile>:            ChromoPainter/PHASE style SNP file
<outputphasefile>:       Output phase file

<options>:
-p <val> : Ploidy
-v: Verbose mode
```

11.4 phasescreen.pl

Remove non-varying SNPs and singletons from a PHASE file. This speeds execution of ChromoPainter and does not change the output.

Listing 19: phasescreen

```
> phasescreen.pl
```

```
REMOVE SINGLETONS OR NON-SNPS FROM PHASE DATA
usage:   perl phasesscreen.pl <phasefile> <outputphasefile>
```

11.5 phasesubsample.pl

Subsamples phase-style data in a contiguous block. This is useful for pipeline generation and testing, although ChromoPainter now provides this facility with the `-l <from> <to>` format which you can specify in -s12args.

Listing 20: phasesubsample

```
> phasesubsample.pl
```

```
EXTRACTS SNP RANGE FROM PHASE (CHROMOPAINTER) FORMAT
usage:   perl phasesubsample.pl <options> <from> <to> <phasefile> <outputphasefile>
where:
<from>:         First SNP to retain (1 is the first snp)
<to>:           Final SNP to retain (L is the last snp, the number on the 3rd line
   of the phase file)
<phasefile>:            ChromoPainter/PHASE style
(http://www.hapmap.org/downloads/phasing/2007-08_rel22/phased/00README.txt) SNP
   file
<outputphasefile>:       Output phase file

<options>:
-v: Verbose mode
NB Compatible with chromopainter and chromopainterv2 phase formats
```

11.6 plink2chromopainter.pl (PLINK)

Conversion script for going from PLINK (`pngu.mgh.harvard.edu/~purcell/plink/`) style PED and MAP files to ChromoPainter's PHASE and RECOMBFILES files. *IMPORTANT NOTE: Use plink -recode12 to get output in an appropriate format for this script!*. Note that many plink commands can be used without losing phasing information, despite PLINK being phasing unaware.

Listing 21: plink2chromopainter

```
> plink2chromopainter.pl
```

```
Usage: ./plink2chromopainter.pl -p=pedfile -m=mapfile -o=phasefile
                [-d=donorfile] [-r=recfile] [-g=chromosomegap] [--quiet] [--asis]

pedfile is a valid PLINK ped inputfile
mapfile is a valid PLINK map file
phasefile will be a valid chromopainter phase file (ChromoPainter's -g switch)
        (i.e. a fastphase file with an additional header line)
donorfile is OPTIONAL and simply stores the list of individual names (NOT
   ChromoPainter's -f switch!)
recfile is OPTIONAL and will be a valid chromopainter recombination file (
   ChromoPainter's -r switch)
chromosomegap (=10e6 by default) is the gap in BP placed between different
   chromosomes
-a or --asis assume the SNPs are stored as 0/1 rather than 1/2 (default plink
   behaviour)
-q or --quiet reduces the amount of screen output
EXAMPLE: ./plink2chromopainter.pl myped.ped mymap.map mydata.phase donor=mydonor.
   donor rec=myrec.map
IMPORTANT: You should use the --recode12 option in plink
MORE HELP ON FILE FORMATS: ./plink2chromopainter.pl -h
```

11.7 impute2chromopainter.pl (SHAPEIT format)

Conversion script for going from IMPUTE2 format, which includes SHAPEIT (`www.shapeit.fr`) output, to ChromoPainter's PHASE and RECOMBFILES files.

Listing 22: impute2chromopainter

```
> impute2chromopainter.pl
```

```
CONVERTS PHASED SHAPEIT/IMPUTE2 OUTPUT TO CHROMOPAINTER-STYLE INPUT FILES
usage:   perl impute2chromopainter.pl <options> <-r recommap_filein>
   impute_output_file.haps output_filename_prefix
where:
        (i) impute_output_file.haps = filename of IMPUTE2 output file with suffix
           ".haps" that contains phased
        haplotypes
        (ii) output_filename_prefix = filename prefix for chromopainter input file(
           s). The suffix ".phase" is added

The output, by default, is in CHROMOPAINTER v2 input format.
<options>:
-J:                Jitter (add 1) snp locations if snps are not strictly ascending
   . Otherwise an error is produced.
-r recommap_filein: Filename of genetic map used to phase haplotypes in IMPUTE2/
   shapeit. If provided, a ".recomrates"
file is also produced.
<further options>   NOTE: YOU ONLY NEED THESE OPTIONS FOR BACKWARDS COMPATABILITY!
-v1:               Produce output compatible with CHROMOPAINTER v1, i.e. include
   the line of "S" for each SNP.
-f:                By default, this script produces PHASE-style output, which
   differs from
```

```
                          ChromoPainter input which requires an additional first
                             line.  This option creates
                          the correct
                          first line for standard fineSTRUCTURE usage (i.e. the
                             first line is "0", all other
                          lines are appended)

NOTE: TO USE IN CHROMOPAINTER: You also need a recombination map. Create this with
   the "-r" option, or use the
"convertrecfile.pl" or "makeuniformrecfile.pl" scripts provided.

 !!! WARNING:  THIS PROGRAM DOES NOT SUFFICIENTLY CHECK FOR MISSPECIFIED FILES. WE
    ARE NOT ACCOUNTABLE FOR THIS
 RUNNING INCORRECTLY !!!
```

11.8 beagle2chromopainter.pl (BEAGLE format)

Conversion script for going from BEAGLE format to Chromopainter PHASE format.

Listing 23: beagle2chromopainter

```
> beagle2chromopainter.pl
```

```
CONVERTS PHASED BEAGLE OUTPUT TO CHROMOPAINTER-STYLE INPUT FILES
usage:   perl beagle2chromopainter.pl <options> beagle_phased_output_file
   output_filename_prefix
where:
        (i) beagle_phased_output_file = filename of BEAGLE phased file (unzipped)
           that contains phased haplotypes
        (ii) output_filename_prefix = filename prefix for chromopainter input file(
           s). The suffixes ".phase" amd
        ".ids" are added

The output, by default, is in CHROMOPAINTER v2 input format. NOTE THAT ONLY
   BIALLELIC SNPS ARE RETAINED, i.e. we
omit triallelic and non-polymorphic sites.
<options>:
-J:                Jitter (add 1) to snp locations if snps are not strictly
   ascending. Otherwise an error is produced.
<further options>   NOTE: YOU ONLY NEED THESE OPTIONS FOR BACKWARDS COMPATABILITY!
-v1:               Produce output compatible with CHROMOPAINTER v1, i.e. include
   the line of "S" for each SNP.
-f:                By default, this script produces PHASE-style output, which
   differs from
                          ChromoPainter input which requires an additional first
                             line.  This option creates
                          the correct
                          first line for standard fineSTRUCTURE usage (i.e. the
                             first line is "0", all other
                          lines are appended)

 !!! WARNING:  THIS PROGRAM DOES NOT SUFFICIENTLY CHECK FOR MISSPECIFIED FILES. WE
    ARE NOT ACCOUNTABLE FOR THIS
 RUNNING INCORRECTLY !!!
NOTE: TO USE IN CHROMOPAINTER: You also need a recombination map. Create this with
   the "convertrecfile.pl" or
```

```
"makeuniformrecfile.pl" scripts provided.
```

11.9 msms2cp.pl (MSMS and MS output format)

Conversion script for going from data simulated by MS (`home.uchicago.edu/rhudson1/source/mksamples.html`) or its variants including MSMS (`www.mabs.at/ewing/msms`), to ChromoPainter's PHASE and RECOMBFILES files.

Listing 24: msms2cp

```
> msms2cp.pl
```

```
CONVERTS MSMS OUTPUT TO CHROMOPAINTER-STYLE INPUT FILES
usage:   perl msms2cp.pl <options> msmsoutput.txt output_filename_prefix

OPTIONS
-c1     : Output chromopainter version1 format
-n <x> : Multiplier for the SNP locations (default: 1000000)
-p <x> : Specify the ploidy (default:2 for diploid; needed only for CP version 1)
-ms <x>: Specify ms mode, and give the number of *haplotypes* in it (because ms
   doesn't include this in the header)
-v      : Verbose mode
```

12 Potential pitfalls

If your data is not correctly in the format we expect, then anything can go wrong. We try to detect this but we don't test everything. Check that your data are valid first!

The main pitfalls that can happen with valid data are:

1. ChromoPainter parameter estimation fails. This happens when the default parameters are too far from the true parameters, and therefore the parameter estimation converges to a suboptimal solution (usually with effectively infinite or zero recombination rate).

   - *Symptoms*: getting a silly value of 'c' (tiny), getting very many or very few chunks: row sums of the chunk count matrix being close to the number of SNPs or being about 1. The *EMprobs.out files probably aren't converged. When running -combines2 you may get a warning about 'c' being out of the expected range.
   - *Happens when*: using data with too large or too small genetic distance between SNPs. Happens with simulated data and with non-humans, particularly when using makeuniformrecfile.pl to make a recombination map, which assumes human-like SNP density.
   - *Solutions*: Rerun stage1 with a different starting location for Ne. Try either very much larger or very much smaller than the default, in the opposite direction to the inferred values. The default is 400000/number of donor haplotypes. Obtain the estimate using `grep Neinf <file>.cp`. Set the parameter via `-s1args:-in\ -iM\ --emfilesonly\ -n <value>` where you replace ¡value¿ with a number, e.g. 10 or 100000. (The other arguments are defaults that only experts should change.)

2. ChromoPainter 'c' estimation fails.

   - *Symptoms*: Usually you will get a 'ChromoCombine' error and be told that no regions were found. You should rerun stage2.

- *Happens when*: The parameters are badly inferred. There isn't very much data. The recombination rate is very low, resulting in high LD.
- *Solutions*: Try setting `-reset 2 -s2chunksperregion <value>` (chromopainter's -k) to a lower ¡value¿, less than the rowsums of each chromosome of each individual. If that is lower than about 20, see below.

3. ChromoPainter 'c' estimation went wrong, but passed tests.

   - *Symptoms*: MCMC results are over-split.
   - *Happens when*: The parameters are badly inferred. There isn't very much data. The recombination rate is very low, resulting in high LD.
   - *Solutions*: As above. If that isn't possible, you may have to resort to setting 'c' manually. `-duplicate 3 <newroot>.cp -s34args:-c\ 1.0` will create a new MCMC run with c=1. This is typically conservative and will be a good baseline for deciding if splits are clear or not.

13 Additional comments

Please report all bugs to Dan Lawson, dan.lawson@bristol.ac.uk.

Contributions:

- ChromoPainter was written by Garrett Hellenthal: ghellenthal@gmail.com.
- This manual and software was written by Dan Lawson: dan.lawson@bristol.ac.uk.