Background

The original Crimap is out of date. It was first written when allozymes were the prevalent molecular markers. Consequently, it has some problems dealing with large pedigrees and large numbers of markers that are available for analysis today.

To deal with this we use the Crigen suite of programs, which prepare text files to make Crimap more capable and efficient with handling modern datasets which are larger and more complicated.

An improved version of Crimap was produced by Jill Maddox and Ian Evans (2.504) which is generally better than the old. However the Crigen suite of programs is not compatible with the improved Crimap.

Consequently, analysis starts with Crigen, then the old Crimap, then continues with Crigen (Autogroup) to produce chromosome-specific files, and then moves to the new Crimap which is better for the analysis.

Michael D Grosz and Monsanto have been providing the Crigen code to researchers to serve the community, and have kindly granted permission to distribute the Crigen programs through this site.

Jill Maddox and Ian Evans, who are still developing the improved Crimap, provide it for free through the animal genome website. Downloading requires to fill a short form with information on how it is used. It is important to provide as much information as possible, to help with making sure it works properly with current experimental designs, and to help with funding which is easier if a large user base can be demonstrated.

It is best to compile Crimap on the particular machine or OS used, to make sure it works with the available memory efficiently. For example it is possible fora pre-compiled version to crash when it runs out of memory, but a version compiled on the same machine to work fine. In practice, a pre-compiled version should work well enough, except if it crashes, in which case you should compile your own version which may not crash.

In practice I start the analysis with Crimap 2.4 and Crigen on Ubuntu Linux, running on a virtual machine on MacOSX because I could not compile the old programs in a modern machine. I then continue to the Crimap 2.5 which can be compiled on MacOSX (see Installation Instructions)

All commands have more options with which to run, however the following steps can be used as a backbone of the analysis. The full options are available from the three manuals that come with Crimap.

Workflow

Prepare a .gen file

The general format of a .gen file is shown in the table.

Example `.gen` file.

ANIMAL_ID	SIRE_ID	DAM_ID	Marker1		Marker2		Marker3	
1	0	0	A	C	A	T	A	A
2	0	0	A	C	A	G	A	C
3	1	2	A	C	A	G	A	C

Individuals must be named with numbers.

Sex-linked markers will be homozygotes for one sex (for example in males in *Drosophila*). Edit their genotyping so they are heterozygous, with one allele not present in females. For example a locus which is scored as either A or T and in males is either AA or TT should be changed in males to become AC or TC. Alternatively, the non existent allele can be set to `-1`.

Full instructions on how to prepare a `.gen` file from GenomeStudio are at the my GenomeStudio workflow.

Generate a Twopoint file for Autogroup

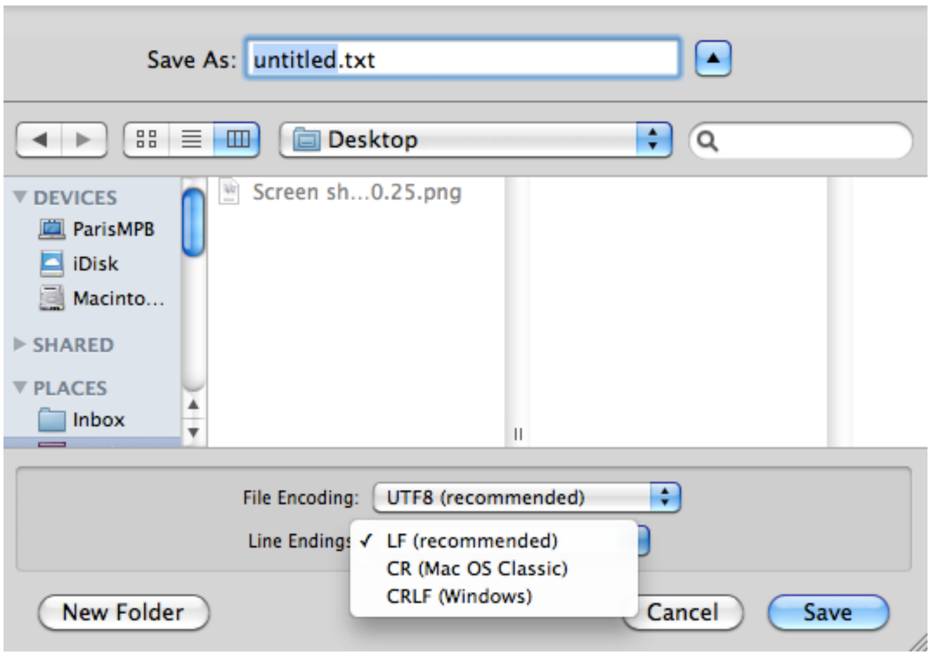
Separate the pedigree into families of about 50 individuals for 3 generations. These files are easier for Crimap to work with.

```
./crigen -g myData.gen -o 10 -size 50 -gen 3
```

```
./crimap 10 prepare
```

choose no no no and then option 7 (twopoint) yes yes

Note, segmentation errors can be reported during the 'prepare' command. I found that they were resolved by choosing Linux encoding for line endings in the `.gen` file. This is the default way text files are saved from BBEdit or Textmate.


File Encoding: UTF8 (recommended)

Line Endings: ✓ LF (recommended)

Save as window in TextMate. Choose 'LF (recommended)' for line encoding.

Alternatively, it is possible to convert line endings to Linux format in the terminal using

```
r '\r' '\n' < in.txt > out.txt
```

next, run twopoint

```
./crimap 10 twopoint > chr10.tpt
```

The output of this command differs between the new and old crimap versions. The old version provides the sex-average lod score while the new version provides the sex-specific lod score. Up to now we should be running the old version because the Autogroup command (below) expects its input in the old format.

`chr10.tpt` shows all two-way linkages between markers. Consult the lod scores (end of each line). They give a feel of how linked the markers are. The file can be consulted to find markers that are not linked, to be assigned to different chromosomes.

Identify obvious errors

Investigate the individuals that cause many inheritance errors by cross-referencing to their scoring in GenomeStudio. If the errors cannot be reduced, mark the individuals as un-scored for the error-prone markers (manually in the genotyping file). If the same individual causes errors with many markers consider removing the individual from the analysis.

An easy way to identify pedigree errors from a genotyping file is to use GenotypeChecker by the Roslin institute. The `.gen` file that Crimap uses needs to be converted to work with GenotypeChecker.

Prepare files for Genotype Checker

The following instructions generate files for Genotype Checker using R. They are adapted from a script provided by Jon Slate. The paths assume you are working on a Mac.

The two input files and the associated R script are available here, but also described in the text below.

Input files

We need two input files per chromosome:

A genotype file which looks like

```
Progeny Sire    Dam Sex MarkerA.1   MarkerA.2   MarkerB.1   MarkerB.2
98   0    0    M    A    G    A    G
168  98   1    F    G    G    G    G
169  57   100  F    A    G    A    G
170  57   100  F    G    G    A    G
99   0    0    F    A    G    A    G
```

and a map file which looks like

```
marker  position
MarkerA 0.000
MarkerB 43.987
```

In R

Read in the genotype file

```
genotype_file<-read.table("/Users/username/Desktop/genotype_file.txt", header=T)
```

Retain just the pedigree information

```
GenotypeCheckerPedFile <- genotype_file[,1:4]
```

Import map data and add a marker order column to them

```
example.map <- read.table('/Users/username/Desktop/exampleMap.txt', header=T, sep="\t")
maporder <- example.map
```

Store the number of markers in a variable

```
m <- (ncol(genotype_file)-4) * 0.5
```

Store marker order in a column

```
maporder$maporder <- seq (1,m,1)
```

Make the genotype file for genotype checker

```
GenotypeCheckerMarkerFile <- genotype_file[,c(1,5,6)]
GenotypeCheckerMarkerFile[,4] <- maporder[1,1]
GenotypeCheckerMarkerFile <- GenotypeCheckerMarkerFile [,c(1,4,2,3)]
colnames(GenotypeCheckerMarkerFile) <- c("ID","Locus","Allele1","Allele2")

for (i in 2:m) {
    GenotypeCheckerTemp <- genotype_file[,c(1,(2*i+3),(2*i+4))]
    GenotypeCheckerTemp[,4] <- maporder[i,1]
    GenotypeCheckerTemp <- GenotypeCheckerTemp[,c(1,4,2,3)]
    colnames(GenotypeCheckerTemp) <- c("ID","Locus","Allele1","Allele2")
    GenotypeCheckerMarkerFile <- rbind (GenotypeCheckerMarkerFile,GenotypeCheckerTemp)
}
```

Find and replace unknown genotypes with ? required by genotype checker

```
GenotypeCheckerMarkerFile$Allele1 <- gsub ("^0","?", GenotypeCheckerMarkerFile$Allele1)
GenotypeCheckerMarkerFile$Allele2 <- gsub ("^0","?", GenotypeCheckerMarkerFile$Allele2)
```

Export data

```
write.table (GenotypeCheckerMarkerFile, '/Users/username/Desktop/GenotypeCheckerMarkerFile', col.na
mes=F, sep="\t",quote=F,row.names=F)
write.table (GenotypeCheckerPedFile, '/Users/username/Desktop/GenotypeCheckerPedFile', col.names=F,
 sep="\t",quote=F,row.names=F)
```

Crimap pedigree error reporting

After using GenotypeChecker, this section is not as useful, but I have kept it in case someone finds it useful.

The crimap command above outputs inheritance errors in the terminal window. If they are too many, they do not fit in the terminal window even if scrolling up. The following steps recover them and use them to make a manageable text file which can be used to identify particularly error-prone individuals, which may reflect pedigree errors or bad DNA preparation for those individuals.

Run the `script` command in the Terminal. This outputs all Terminal text to a `typescript.txt` file. Run the commands as normal and then run `exit`. For example:

```
./crigen -g myData.gen -o 10 -size 50 -gen 3

script

./crimap 10 prepare

exit
```

The `typescript.txt` file contains information on inheritance errors. Sometimes it is in duplicate lines which can be combined to single lines indicating the number of duplicates each represented with the uniq command. Their text is then stripped off and they are converted to tab-delimited files for R.

```
echo "Repetition,Family,Individual,Locus" > temp.txt

perl -pe 's/,/\t/g' temp.txt > nonInheritanceData.txt
```

```
sort typescript.txt | uniq -c | grep "NONINHERITANCE" | perl -pe 's/ NONINHERITANCE:
family FAMILY_/\t/g ; s/, locus /\t/g ; s/, individ /\t/g ; s/ +//g' >>
nonInheritanceData.txt
```

All individuals are renamed by crimap. The correspondence between the old and new names is shown the file `CRIMAPID_G3S50.txt` which is generated when running Crimap.

It should be possible to use the `CRIMAPID_G3S50.txt` and `nonInheritanceData.txt` files to identify the individuals causing errors. I have not managed to see how they correspond, so I use GenotypeChecker to identify any errors, instead.

Autogroup

Once scoring and pedigree errors are zeroed, we can continue with mapping. The purpose of the Autogroup command is to spread the markers across separate unlinked groups, ideally chromosomes.

Start by running

```
./autogroup -n 10 -para autogroup.txt
```

`autogroup10.txt`[1] gives thresholds for confidence to be used by the Autogroup command, for deciding how to group the markers into linkage groups. The command can be run with multiple values until the optimal number of recombination groups is obtained.

It is possible that it will work and many markers go to the to-be-mapped list. Those markers appear in separate text files in the `MRKS_CHROM` folder.

It is possible to help chromosome assignment by providing a `locations.txt` file, with two columns, a marker name and a linkage group. Choose markers with many meioses, as shown in the `MARKER_BY_GROUP.TXT` file, made by autogroup. In my experience providing the `locations.txt` file was necessary to get any markers assigned to groups.

```
./autogroup -n 10 -old locations.txt -para autogroup.txt
```

The aim is to get as many markers as possible in the to-be-mapped list.

- Add one linkage group, run autogroup, and add one more marker from the `MRKS_CULLED.TXT` file to the `locations.txt` file, in a new linkage group. The top marker is the best, because it has the largest number of meioses.
- If the marker added is suggested to go to a preexisting chromosome, by the autogroup output, remove it from the `locations.txt` file and add another one. Too many markers to the same linkage group in the `locations.txt` file may result in erroneously assigning markers from different chromosomes to the same linkage group.
- Markers with low linkage to others, judged from the lod score of the `.tpt` file can also be tried on new linkage groups.
- The best markers to go to a map are the informative ones, as judged from the number of informative meioses given in the `.loc` file.

Run Upig

Upig makes a .gen file using only the markers of a linkage group, previously identified by Autogroup. It should be run for every `CROM_XX.TXT` file produced by the auto group command (XX is a number). `CROM_XX.TXT` files show the markers that are linked together in groups, and their recombination distance. Ideally they represent chromosomes. Move the files to the folder the Upig command is run from.

```
./upig -g myData.gen -o 1 -subm CHROM_01.TXT
```

```
./crigen -g 1 -o 1 -size 50 -gen 3
```

outputs `chr1.gen`

Once there is a `.gen` file for all chromosomes, move over to the new Crimap version.

The following commands should be run for each linkage group (`chr1.gen`, `chr2.gen`, etc).

Run Twopoint

Visually check that the markers are linked and the lod score tolerance to use. The `1` used in the command below is a shortcut for `chr1.gen`. Answer the prepare commands as above.

```
./crimap 1 prepare
```

```
./crimap 1 twopoint
```

Run Build

```
./crimap 1 prepare
```

option 1

This makes the `chr1.par` file which has options for the next command.

```
./crimap 1 build > chr1build.txt
```

This command takes some time to run. Open the resulting file. Towards the bottom there should be a sex-averaged map and a sex-specific map. Just after these there will be the order or markers it came up with. Copy these to clipboard to compare when running build again.

Run build **again** without the prepare command. Run until no more markers are added in `chr1build.txt`. Then manually reduce PUK_LIKE_TOL and PK_LIKE_TOL values in the `chr1.par` file (for example from 3 to 2, then 1 and then 0.01) and run again until no more markers are added.

Copy the marker order from the final `chr1build.txt` file to the clipboard.

Run Flips

Flips switches the order of markers locally and reports whether they fit the data better or worse. The workflow is to run flips, manually make changes in the marker order `chr1.par` file, and run flips again until no better order can be found. The `chr1.par` file is made once with the prepare command, and then manually edited in a text editor.

```
./crimap 1 prepare
```

option 5 for flips

Answer questions as above. When asked, input the order copied in the clipboard.

```
./crimap 1 flips
```

See which order is preferred and run with new order. Define the new order directly in `chr1.par` file in a text editor. Any negative numbers in the output in the end of rows indicate preferable marker orders.

If PUK_LIKE_TOL is set to 0 in the par file, only equally likely or better orders are shown in the output.

Bad markers are not linked with anything and will be put in the end of a chromosome. This is a problem for the flips command because it cannot compare the two ends of chromosomes. Manually add those markers to either end of a chromosome, and if they are too distant from any other marker (close to 50 cM) consider removing them.

Flips can be run for every 2, every 3, every 4 etc markers. Start with low numbers which run fast, and run more as the order improves. Running more than 5 takes a long time but it should be possible to run up to 6 in a modern computer.

```
./crimap 1 flips3
```

```
./crimap 1 flips4
```

```
etc
```

Smart options here will allow to flip in only part of the map and speed up processing. Define a start and end locus number (from the flips output, i.e. what you see on the screen).

For example if we are only interested on the end of

```
21 25 0 12 13 30 26 18 11 8 15 16 35 20 24 1 2 32 10 28 3 29 5 17 31 23 4 7 23 9 14
19 22 6
```

Just type:

```
./crimap chrNumber flips6 -s 31 -e 6
```

Run Chrompic

Chrompic is used to make a map with markers and recombination frequencies. It also reports errors, and can show sex-specific maps. To calculate the sex-specific map, change `SEX_EQ` from 1 to 0 in the `.par` file, after running prepare.

```
./crimap 1 prepare
```

```
./crimap 1 chrompic > chr1chrompic.txt
```

Visualise the map in the end of the output file. Look for sensible distances between markers. Maps with too long distances are unrealistic and not useful.

Example of a sex-specific map:

Sex-specific map (recomb. frac., Kosambi cM – female, male):

1 Marker1 0.000 0.000
0.157 16.220 0.000 0.000
2 Marker2 16.220 0.000
0.102 10.397 0.000 0.000
3 Marker3 26.617 0.000
0.276 31.123 0.000 0.000
4 Marker4 57.740 0.000
0.203 21.599 0.000 0.000
5 Marker5 79.339 0.000

The text above the map in the chrompic file represents the grand-paternal chromosomes of each individual. `i`, `1`, `o`, `0` are grand-paternal and grand-maternal inheritance markers. The two versions of the letters/numbers reflect confidence and can be ignored. – is missing information.

Example of the text above the map:

Family FAMILY_5 phase likelihood = 0.016, 2d best = 0.016

17 -i–1–1–11 i—i-i–1 –1i-i 0
———- ———1 –1—- 0

18 iii-iio-ii i—i-i–i –ii-i 2
7 Repeat.12563_SNPunique2
-i—–– i–ii-i— i—ii 0

28 -oo—oo– ——i— -io-io 4
23 Repeat.13065_99_SNPcommon1 25 Repeat.31926_SNPunique2 26
Repeat.14058_72_SNPcommon2
-ii-i–ii- —-i—– –i-ii 0

The number on the right indicates the number of crossovers. It is atypical to have more than 2. If particular individuals have many recombination events (>3) they are problematic. Single markers showing opposite inheritance than others across a chromosome arm are obvious errors. They either have bad call rates, were low quality samples and may need to be replaced with NA if their calling cannot be changed.

The bottom line on each individual represents the chromosome coming from the father. In *Drosophila* this should have no recombination because there is no recombination in males. There may still be legitimate cases of recombination, if phase shifts from different grandparents are supported by multiple markers.

In practice, first look at the number on the right to identify individuals with multiple recombinations. If these repeat across chromosomes (i.e. the same individual comes up in multiple chrompic files), these individuals are dodgy and should be checked and probably removed. Once such individuals are removed, look at each family for columns (representing markers) with multiple recombination errors. These are bad markers and should be removed.

Finally, there will be individuals markers at particular individuals that cause too much recombination. These need to be zeroed in the `.gen` file.

This is an iterative process, i.e. consult the Chrompic files to identify badly scored markers, zero them, and go through the procedure again.

A bash script that filters the chrompic output to only show lines with a particular number of recombination, is available at the crimap users forum.

The `chrn.double` file, is a new file that is automatically output in Crimap2.504 when running chrompic, and contains double recombinant information.

Drawing the map in MapMaker

The free software MapMaker can be used to make maps.

The input file is assembled manually from the chrompic output.

An example of the input file format for MapMaker is:

```
; comments can be left under ‘;’

chrom “Chromosome 1” S=0 E=24.872
MarkerName1 0
MarkerName2 8.19
MarkerName3 24.872

chrom “Chromosome 2” S=0 E=19.305
MarkerName14 0
MarkerName12 12.238
MarkerName25 19.305
```

`S` and `E` stand for the start and end positions of the map.

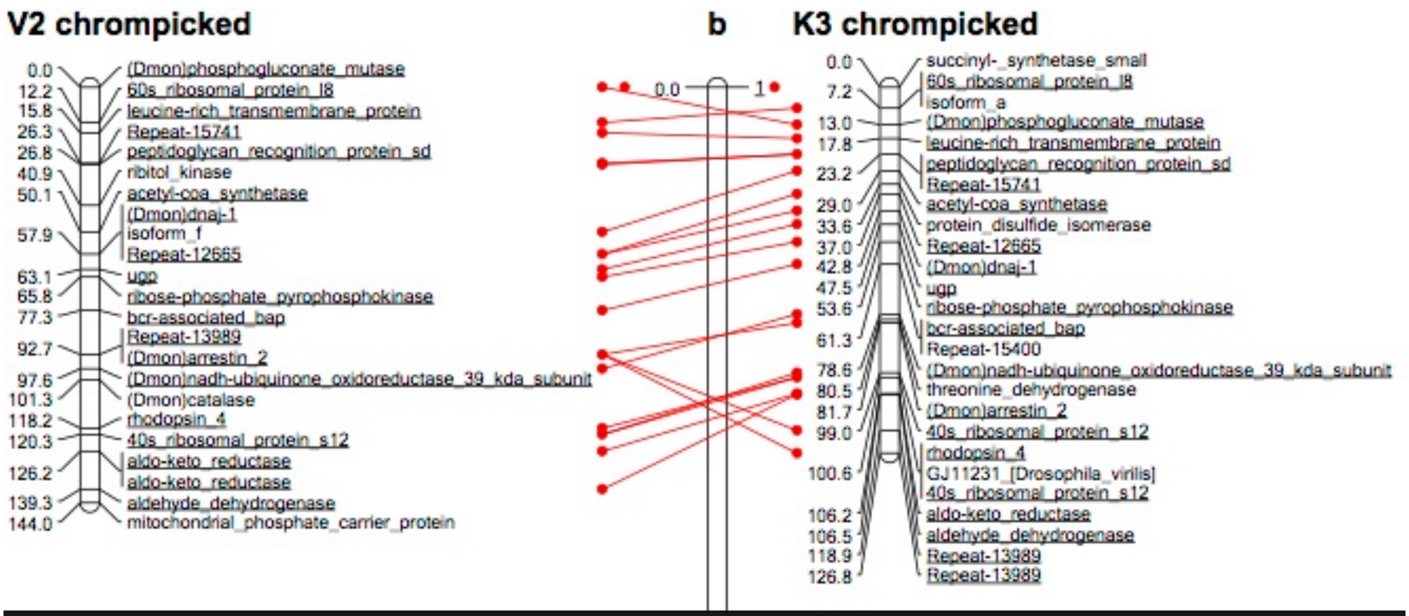
If the same chromosome has been reconstructed from different populations, the files can be combined within MapMaker and common markers can be indicated based on their names.

Adding `p` after the chromosome name, will draw it in a new page. This is useful when all chromosomes are drawn at once.

When comparing chromosomes between multiple populations or iterations of Crimap, it is possible that the whole chromosome is inverted. If so, manually invert the values from the text file, and draw again.

To compare the same chromosome it is useful to add an empty chromosome between them, so that it is easy to tell whether the lines connecting the same marker between the chromosomes are parallel or cross, indicating inversions.

Once a picture is assembled in MapMaker, choose “Export pages”. The resulting files are in .emf format. The easiest way to open them is in Word, and save them as pdf to retain maximum quality.



V2 chrompicked
b
K3 chrompicked
0.0 (Dmon)phosphogluconate_mutase
12.2 60s_ribosomal_protein_l8
15.8 leucine-rich_transmembrane_protein
26.3 Repeat-15741
26.8 peptidoglycan_recognition_protein_sd
40.9 ribitol_kinase
50.1 acetyl-coa_synthetase
57.9 (Dmon)dnaj-1 isoform_f Repeat-12665
63.1 ugp
65.8 ribose-phosphate_pyrophosphokinase
77.3 bcr-associated_bap
92.7 Repeat-13989 (Dmon)arrestin_2
97.6 (Dmon)nadh-ubiquinone_oxidoreductase_39_kda_subunit
101.3 (Dmon)catalase
118.2 rhodopsin_4
120.3 40s_ribosomal_protein_s12
126.2 aldo-keto_reductase aldo-keto_reductase
139.3 aldehyde_dehydrogenase
144.0 mitochondrial_phosphate_carrier_protein
0.0
1
0.0 succinyl-_synthetase_small
7.2 60s_ribosomal_protein_l8 isoform_a
13.0 (Dmon)phosphogluconate_mutase
17.8 leucine-rich_transmembrane_protein
23.2 peptidoglycan_recognition_protein_sd Repeat-15741
29.0 acetyl-coa_synthetase
33.6 protein_disulfide_isomerase
37.0 Repeat-12665
42.8 (Dmon)dnaj-1
47.5 ugp
53.6 ribose-phosphate_pyrophosphokinase
61.3 bcr-associated_bap Repeat-15400
78.6 (Dmon)nadh-ubiquinone_oxidoreductase_39_kda_subunit
80.5 threonine_dehydrogenase
81.7 (Dmon)arrestin_2
99.0 40s_ribosomal_protein_s12
100.6 rhodopsin_4 GJ11231_[Drosophila_virilis] 40s_ribosomal_protein_s12
106.2 aldo-keto_reductase
106.5 aldehyde_dehydrogenase
118.9 Repeat-13989
126.8 Repeat-13989

Example of MapMaker output. An empty chromosome in the middle allows to easily visualise inversions.

General notes

Low quality data inflate recombination distances and disproportionately affect the data. Hence the Golden rule:

> *‘No data is better than bad data’.*

Removing pedigree errors is not expected to allow more markers to be added on the maps.

Chromosome ends

The edges of chromosomes sometimes pick up markers that are not linked to the rest of the chromosome. If those markers are put on the other edge of the chromosome with similar likelihood, they do not belong to the map. There is the complication that chromosome ends tend to have low recombination rates, so markers close to centromeres might truly have a large recombination distance with other markers.

Male typing

In *Drosophila* no recombination is expected in males. However the male maps do have small recombination distances. Most of these are a reflection of genotyping errors, caused, for example, by bad DNA quality. Some may be due to single individuals that were female and wrongly called males, but these will be picked up in Chrompic.

Installation Instructions

Crigen

The programs, along with the source code of the compatible Crimap (v 2.4), are available here. The accompanying user guide is here.

I could not get Crimap 2.4 to compile on MacOSX, so I used a pre-compiled version on Ubuntu Linux.

Download, go to folder and unzip

```
tar -xzvf *.tgz
```

Crimap 2.5

Mountain Lion requires Xcode installed before you can compile anything. You may need to get a developer ID, then the downloads section should link to the ‘command line tools’, which include the `make` command. Once Xcode is installed, get the Crimap files.

Before you can download, there is a questionnaire. It is important to provide information on your particular study, because Crimap is in active development and effort is being made for new versions to take into account modern datasets and experimental designs.

Unzip and locate the file `defs.h`. Change the following according to your system (information on them is provided in the manual which comes along with the source files).

- HIGH_MEMORY_SIZE
- MAX_PRODUCT_CALCS
- HIGH_MEMORY_BITS
- MAX_PRODUCT_CACHE_MEMORY

Compile using,

```
make crimap
```

On MacOSX, there may be an error message involving `values.h`, during compilation. I managed to compile by replacing

`values.h` with `float.h` in the following source files, as suggested here.

- e_chrom.c
- e_ped.c

- get_like.c
- screen_o.c
- Map.cc

Problems?

The Crimap users group provides a mailing list and an archive of previous questions, where you may be able to find solutions to problems you have been having.

---

1. An example autogroup file (used in the *D. montana* project) is below. It is project-specific and depends on parameters such as pedigree structure, number of individuals and number of markers. ↩

   (30, 1.5, 2, 0.9)

   (15, 1.2, 3, 0.7)

   (5, 0.8, 5, 0.6)

   (3, 0.2, 10, 0.3)