

# Crosslink - polyploid-aware genetic mapping for outbreeders

---

## Synopsis

Builds genetic maps using codominant biallelic markers derived from the F1 progeny of an outcross between two unrelated non-inbred parents, with additional features suitable for allopolyploid species.

Crosslink was written and tested on the Linux operating system, and designed to allow a high degree of automation by the use of scripting. Cross platform deployment is currently supported by the use of virtual machine and docker images. The source code is also freely available.

Crosslink consists of a number of programs, written in C, and helper scripts written in bash and python. The main programs are:

**crosslink\_group** performs grouping, phasing and approximate marker ordering, and also redundant marker removal, missing value imputation and polyploid-related marker-typing error correction

**crosslink\_map** performs final marker ordering and assigns final map positions, and also, for allopolyploids, detects markers which falsely join homeologous linkage groups

**crosslink\_pos** performs a subset of crosslink\_map's functionality, assigning final map positions

**crosslink\_viewer** produces a two dimensional colour-coded plot showing an all-versus-all marker comparison indicating phasing, rf, LOD and map distances for one or more linkage groups at a time

**crosslink\_graphical** shows a graphical representation of genotype values along one or more linkage groups

**create\_map** and **sample\_map** allow for creation of simulated marker data

## File Formats

### genotype file

A file containing marker data in a format similar to the loc file format of JoinMap<sup>®</sup>, but lacking any header information. Each marker occupies one line as follows:

```
<name> <type> <phase> <genotype1> [<genotype2>...]
```

There must be no blank lines or comments. All columns must be separated by a single space. A marker name can be any string, but should not contain any white space characters (space, tab etc.). Marker type must be one of the following three: <lmxll> indicating a biallelic codominant marker heterozygous in the maternal and homozygous in the paternal parent, <nnxnp> a biallelic codominant marker heterozygous in the paternal and homozygous in the maternal parent, or <hkxhk> a biallelic codominant marker heterozygous in both parents. Marker phase must always be specified but can be set arbitrarily to zero where phase is not yet known and will be ignored. For lmxll markers the phase must be either {0-} or {1-}, for nnxnp markers either {-0} or {-1} and for hkxhk markers one of {00}, {01}, {10} or {11} where the first and second digits indicate the maternal and paternal phase respectively. Each marker in the same file must have the same number of genotypes, each of which must consist of exactly two characters. For all marker types missing values must be specified as --, for lmxll markers genotypes must be either lm or ll, for nnxnp either nn or np and for hkxhk one of hh, kk, hk or kh. If the hk genotypes have not been imputed then hk and kh will be treated the same, whereas for imputed genotypes hk is taken to mean allele h was from the maternal parent and kh to mean allele h was from the paternal parent. Example markers:

51

52 MK-1234 <lmxl1> {0-} lm lm ll lm ll -- ll lm

53 4567.890 <nnxnp> {-1} nn nn -- np -- -- nn np

54 abcd <hkxhk> {01} hh kk hh hk hk -- hk kh

55

56 There is no way to specify groupings of markers into linkage groups within a single file.

57 Initially all markers should be together in the same file. When linkage groups are formed

58 each is put into a separate file, and different versions of the map can be organised into their

59 own subdirectory where required.

## 60 **map file**

61 crosslink\_map outputs one map file per linkage group using a four column format giving the

62 marker name and its centimorgan position in the maternal, paternal and combined map.

63 Where a marker is not present or its position cannot be calculated in one of the maps, its

64 position is listed as NA. The first line of the file contains the linkage group name and the

65 number of markers as follows:

66

67 group <linkage\_group\_name> ; markers <number\_of\_markers>

68

69 Each marker is listed using four whitespace (i.e. tab and space characters) separated

70 columns:

71

72 <marker\_name> <maternal\_map\_position> <paternal\_map\_position> <combined\_map\_position>

73

74 **Example map file:**

75

76 group 000 ; markers 126

77 PHR10-89849276 0.0000 NA 0.0000

78 NMH21-89792123 NA 0.0000 0.0000

79 PHR11-89793667 0.0000 0.0000 0.0000

80 NMH12-89832883 0.6211 NA 0.7819

81	PHR10-89793765	1.2423	NA	1.5637
82	NMH01-89892406	NA	2.5322	2.1004
83	PHR21-89835283	NA	3.7823	3.1373
84	PHR11-89793784	2.4923	3.7823	3.1373
85	NMH12-89793368	3.1134	NA	3.6021

86

## 87 **Installation**

### 88 **Using the Prebuilt VirtualBox Image**

89 Cross-platform installation of Crosslink is only currently supported through the use of virtual  
90 machine and docker images. To use the graphical visualisation tools it is recommended to  
91 use the virtual machine rather than docker image, as everything should work without any  
92 further setup required. The VirtualBox software is currently available from Oracle for  
93 Windows, OS X, Linux and Solaris from: <https://www.virtualbox.org> and may also be  
94 available for your Linux distribution through its built-in package management system, for  
95 example on Ubuntu it can usually be installed by using the following command:

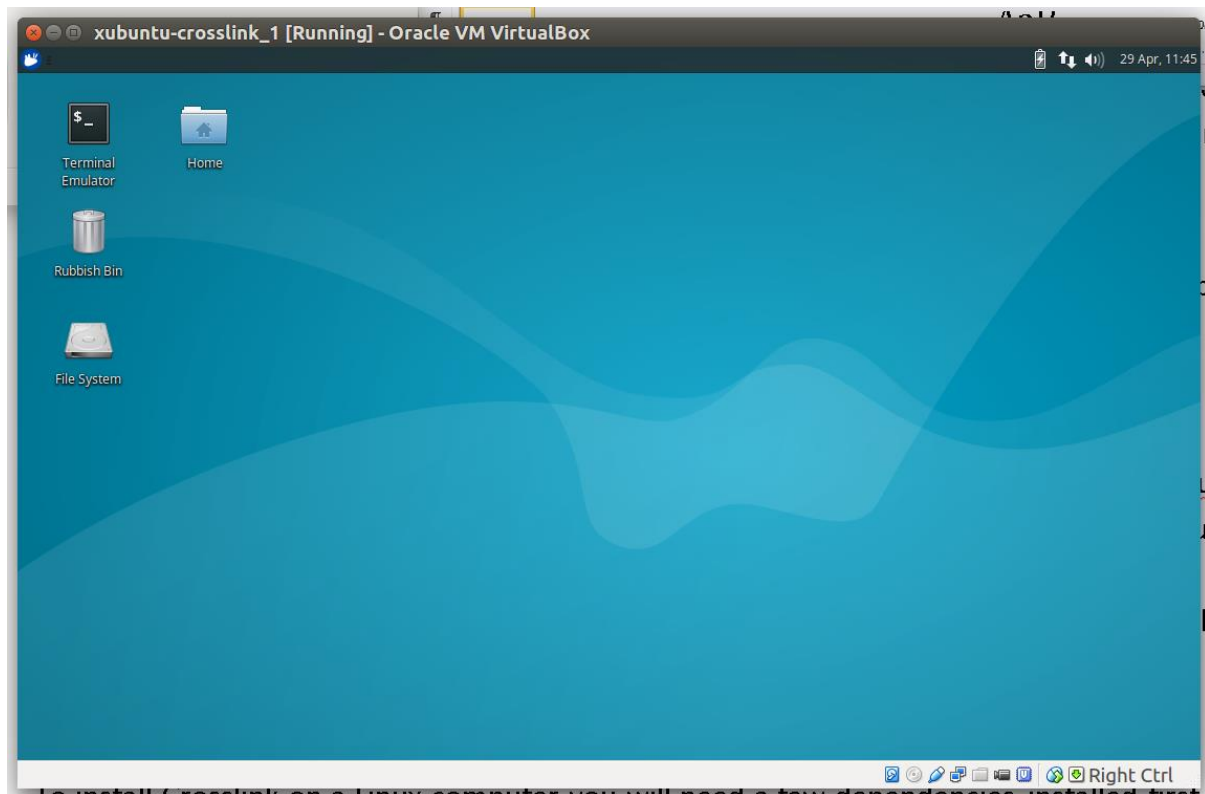
96

```
97 sudo apt-get install virtualbox
```

98

99 Once VirtualBox is installed, download the virtual machine file xubuntu-crosslink.ova  
100 (approximately 2GB) from the latest Crosslink github release:  
101 <https://github.com/eastmallresearch/crosslink/releases>

102 Launch VirtualBox and import the image using File->Import Appliance, then select the  
103 xubuntu-crosslink.ova file you just downloaded. When the Appliance Settings dialog appears  
104 adjust the amount of RAM and the number of CPUs you wish to allocate to the virtual  
105 machine if required, then click Import. Once imported select the virtual machine you have  
106 just created and click the green Start arrow. The image contains Xubuntu 16.04, a minimalist  
107 version of Ubuntu using the Xfce desktop environment, which should appear like this once  
108 loaded:



109

110 You may wish to enable copy-pasting between the host and guest operating system, if so

111 Devices -> Shared Clipboard -> BidirectionalCrosslink in the VirtualBox menu system.

112 Crosslink is already installed. To test it is working open a terminal by clicking on the black

113 Terminal Emulator icon and run the following commands:

114

115 `mkdir test`

116 `cd test`

117 `~/crosslink/example_pipelines/simple_simulated.sh`

118

119 After the command finishes type the 'ls' command and you should see 28 .loc and .map files

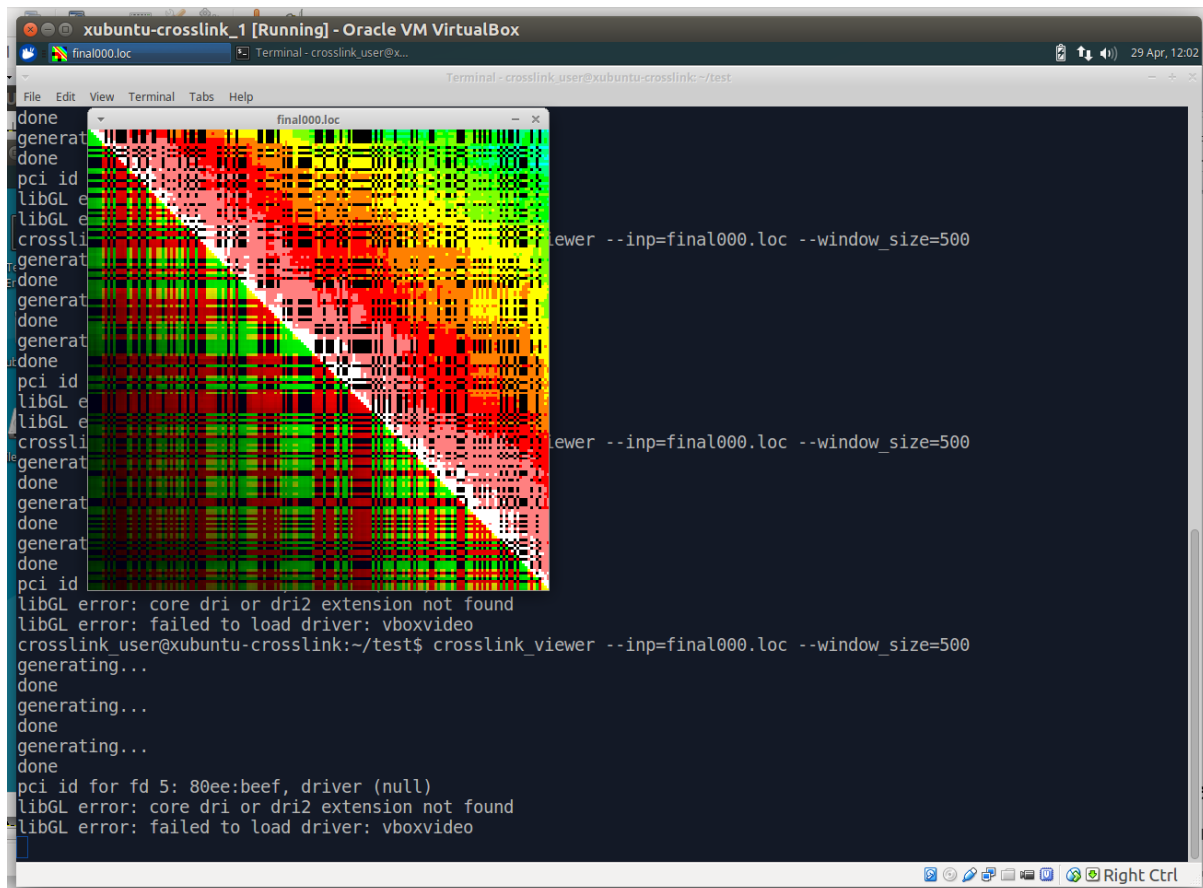
120 called group and final. To test a data visualisation command run the following:

121

122 `crosslink_viewer --inp=final000.loc --window_size=500`

123

124 You should see something like this:



125

126 Press 'q' or close the graphical window to quit. Type 'exit' or close the window to exit the  
 127 terminal. To shut down the virtual machine click on the mouse icon in the top left and click on  
 128 the logout button at the bottom right of the menu, then select Shutdown. The password of  
 129 the crosslink\_user account is crosslink, you may want to change this to something more  
 130 secure using the passwd command.

## 131 Installing Directly on Linux

132 To install Crosslink on a Linux computer begin by downloading the latest release of the  
 133 source code from github at <https://github.com/eastmallresearch/crosslink/releases>  
 134 (choose either Source code (zip) or Source code (tar.gz)) and extract the files somewhere.  
 135 You may wish to move the folder to a more convenient location such as ~/crosslink. To  
 136 compile the main Crosslink programs you will need gcc installed. Open a terminal. Type the  
 137 following to see if you have it already installed:

138

139 `gcc --version`

140

141 If not, on Ubuntu or Debian based systems gcc can usually be installed using:

142

```
143 sudo apt-get update
```

```
144 sudo apt-get install gcc
```

145

146 If you do not have the ability to install packages on your system ask your system  
147 administrator to help you. If you use a different Linux distribution use the appropriate  
148 package management tool for your system. Now compile the main Crosslink programs using  
149 (assuming the crosslink source code is now in a directory called ~/crosslink):

150

```
151 cd ~/crosslink/src
```

```
152 ./make.sh
```

153

154 If no error messages appear the compilation was probably successful, and the programs  
155 crosslink\_group, crosslink\_map and crosslink\_pos should have appear in the bin directory.  
156 Test you can run crosslink\_group as follows:

157

```
158 ../bin/crosslink_group --help
```

159

160 You should see a help message starting "Crosslink Copyright (C) 2016 NIAB EMR...". To  
161 compile the data visualisation tools you will need SDL2 installed, including the development  
162 files. On Ubuntu or Debian based system you should be able to install these using:

163

```
164 sudo apt-get install libsdl2-2.0-0 libsdl2-dev
```

```
165 sudo apt-get install libsdl2-image-2.0.0 libsdl2-image-dev
```

166

167 Now compile the data visualisation programs (crosslink\_viewer and crosslink\_graphical)  
168 using:

169

170 `./viewer_make.sh`

171

172 Check you can run `crosslink_viewer` using:

173

174 `../bin/crosslink_viewer --help`

175

176 You should see a similar help message appear. Next, to allow running the programs from  
177 any directory on your system add the path to Crosslink's programs to your `.bashrc` file. Here  
178 I will assume that the editor `nano` will be used (install first using `sudo apt-get install nano` if  
179 required). Launch the `nano` editor using:

180

181 `nano ~/.bashrc`

182

183 Add the following lines to the bottom of the file (again assuming Crosslink files were installed  
184 to `~/crosslink`):

185

186 `export CROSSLINK_PATH=~/crosslink`

187 `export PATH=${CROSSLINK_PATH}/bin:${CROSSLINK_PATH}/scripts:${PATH}`

188

189 Make sure not to omit the final `${PATH}` from the second line! Press `CTRL-O` then `ENTER` to  
190 save and `CTRL-X` to close the editor. Then close the terminal and open a new one (or log  
191 out and then back in again if you are logged in remotely). You should now be able to run  
192 Crosslink's programs from any directory:

193

194 `cd ~`

195 `mkdir test_crosslink`

196 `cd test_crosslink`

197 `crosslink_group --help`

198



199 You should see the same help message as before. Now try running one of the example  
200 pipelines:

201

```
202 ${CROSSLINK_PATH}/example_pipelines/simple_simulated.sh
```

203

204 The pipeline should run and produce new output files in your current directory, including one  
205 called final000.loc. Test you can view this file now using crosslink\_viewer:

206

```
207 crosslink_viewer --inp=final000.loc
```

208

209 which should display a coloured plot similar to the one shown for the virtual machine  
210 installation section above. Some of the python helper scripts require one or more of the  
211 following python modules: scipy, numpy, matplotlib. Install them using:

212

```
213 sudo apt-get install python-scipy python-numpy python-matplotlib
```

214

215 Or, if you have pip available you may wish to try installing them only in your user account  
216 rather than system-wide, using:

217

```
218 pip install --user numpy scipy matplotlib
```

219

## 220 **Using the Docker Image**

221 For expert users a Docker image is available called rjvickerstaff/crosslink. This is not the  
222 recommended way for non-Linux users to run Crosslink as it is more difficult to setup and  
223 use than the VirtualBox method and problems were encountered during testing when trying  
224 to produce graphical output. It is included here in case it proves useful for advanced users.  
225 To use the image first install Docker on your machine following the installation instructions

226 for your platform on the <https://www.docker.com> website. To launch a container use the  
227 following:

228

```
229 docker run -it -u crosslink_user rjvickerstaff/crosslink:latest
```

230

231 Depending on how Docker is setup on your machine you may need to run this command as  
232 root. If running on a Linux host you may be able to get graphical output using:

233

```
234 docker run -it -u crosslink_user -v /tmp/.X11-unix:/tmp/.X11-unix\  
235     -e DISPLAY=$DISPLAY rjvickerstaff/crosslink:latest
```

236

237 The following command, or similar, may also be needed to allow the container permission to  
238 create graphical output:

239

```
240 xhost +local:
```

241

242 The Dockerfile used to create the container can be found at  
243 \${CROSSLINK\_PATH}/docker/Dockerfile.

244

## 245 Quick Start Guide

### 246 Simulated Data Without Polyploid-Related Errors

247 An example of a simple pipeline using Crosslink to generate and map simulated outcross  
248 data is included in the file example\_pipelines/simple\_simulated.sh, which you may already  
249 have run to test your install. If not it can be run using the following:

250

```
251 cd ~
```

```
252 mkdir test_crosslink_simple
```

```
253 cd test_crosslink_simple
```

```
254 ${CROSSLINK_PATH}/example_pipelines/simple_simulated.sh
```

255

256 We will now step through the simple\_simulated.sh pipeline one line at a time, by typing (or  
257 copy-pasting) the commands into the shell. Begin in a new, empty directory:

258

259 `cd ~`

260 `mkdir test_crosslink_simple2`

261 `cd test_crosslink_simple2`

262

263 Now invoke the crosslink\_group command with the help option to verify that the program  
264 runs:

265

266 `crosslink_group --help`

267

268 You should see :

269

270 `Crosslink Copyright (C) 2016 NIAB EMR see included NOTICE file for details`

271 `remove redundant markers, form linkage groups, correct marker typing errors,`

272 `phase, impute missing values, perform approximate ordering`

273

274 `--inp=STRING input genotype file (required)... etc`

275

276

277 Invoking any of the core Crosslink programs (create\_map, sample\_map, crosslink\_group,  
278 crosslink\_map, crosslink\_pos, crosslink\_viewer, crosslink\_graphical) with the --help option  
279 will print a summary of command line options to the terminal and exit. Now create a map  
280 specification file with the following command, which can be entered as a single line without  
281 the backslash characters (\), or across multiple lines retaining the backslashes:

282

283 `create_map --output-file=sample.spec --numb-lgs=28\`

284 `--map-size=2000 --marker-density=2.0\`

```
285         --prob-both=0.28 --prob-maternal=0.36
```

286

287 A new file called sample.spec should have appeared. Take a look at its contents using:

288

```
289 head sample.spec
```

290

291 It should begin with something similar to:

292

```
293 #markers 4000 lgs 28
```

```
294 M000000a98 <nnxnp> {-1}    0 0.465766
```

```
295 M000000029 <lmxl1> {0-}    0 2.513004
```

296

297 where each line specifies the marker name, type, phase, linkage group and centimorgan  
298 map position. The options given to create\_map define the number of linkage groups (28), the  
299 total map length (2000 centimorgans), the average marker density (2.0 markers per  
300 centimorgan), the approximate proportion of markers heterozygous in both parents (0.28)  
301 and heterozygous in the maternal parent only (0.36) with the proportion heterozygous in the  
302 paternal parent being defined implicitly as  $1.0 - 0.28 - 0.36 = 0.36$ . Use create\_map --help for  
303 more help with the options. Next we will simulate some genotype data obtained from an F1  
304 cross:

305

```
306 sample_map --input-file=sample.spec --output-file=sample.loc\
```

```
307         --samples=162 --prob-missing=0.007 --prob-error=0.01
```

308

309 where we have told sample\_map to simulate data from 162 progeny with a missing data rate  
310 of 0.007 and a genotyping error rate of 0.01. This program does not generate any  
311 allopolyploid-related errors, which are created using the separate create\_type\_errors.py  
312 script. A new file sample.loc should have appeared, which is a genotype file as described in  
313 a previous section, and can be viewed using:

314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341

```
less -S sample.loc
```

You will notice that all phase values are set to zero and a few genotypes are set to -- indicating they are missing. The markers are also in a random order with respect to their true linkage group and map position. The marker names do not indicate the true marker order in any way, so that it is not possible for a mapping program undergoing testing to inadvertently cheat by alphabetically sorting the markers. For hkxhk markers, all heterozygous genotype calls are given as hk, so that it is unspecified whether the maternally inherited allele was h or k. Press 'q' to quit the less command. Next we will split the markers into a separate file for each linkage group:

```
crosslink_group --inp=sample.loc --outbase=group --min_lod=10.0 --knn=3
```

where we have told the program to split markers based on a linkage LOD threshold of 10, and to impute missing values using the 3 nearest neighbouring markers. The separate linkage group files should have appeared with names starting with group, and be numbered sequentially from 000 to 027. If you wish to rerun the grouping at a different LOD threshold, be aware that crosslink\_group does not automatically delete any existing group files, so it is important to first remove all the old files, by a command such as (be sure you are familiar with the rm command before deleting any files from the command prompt in this way as it is easy to delete the wrong files by mistake):

```
rm group???.loc
```

With the present data a LOD of 10 ought to split the markers into the correct 28 groups. View the first linkage group using:

342 `less -S group000.loc`

343

344 You will notice the phases are now set and there are no longer any missing values  
345 (crosslink\_map does not support missing values). The markers have also been  
346 approximately ordered, so that you may notice consecutive genotypes have similar patterns.  
347 To see how many markers were assigned to each group:

348

349 `wc --lines group???.loc`

350

351 The last step is to produce the final marker order and map positions for each linkage group:

352

353 `for fname in group???.loc`

354 `do`

355  `echo ${fname}...`

356  `outname=${fname/group/final}`

357  `crosslink_map --inp=${fname} --out=${outname} --map=${outname/loc/map}`

358 `done`

359

360 Here we have told crosslink\_map to output the final genotype files with names beginning  
361 with final, and to produce associated map files for each one. View the first final genotype file  
362 using:

363

364 `less -S final000.loc`

365

366 You will notice that hkxhk markers now have some heterozygous genotypes given as kh as  
367 well as hk, indicating that crosslink\_map has imputed which allele is likely to be from the  
368 maternal parent and which from the paternal (the maternal is the first of the two letters).  
369 View the first map file using:

370

371 less final000.map

372

373 You should see something like:

374

375 group group000 ; markers 143

376 M000001da 0.0000 0.0000 0.0000

377 M00000700 NA 1.8870 2.2589

378 M00000055 NA 3.7740 4.5178

379 M00000542 NA 5.0241 6.0142

380 M000007ae 8.7471 6.2742 7.5106

381 M00000f0e 13.2663 NA 10.7209

382 M00000d96 14.5163 NA 11.6090

383 etc

384

385 where the four columns are the marker name followed by its positions on the maternal,  
386 paternal and combined maps respectively. lmxll and nnxnp markers are only present on the  
387 maternal and paternal maps respectively so the other position is listed as NA. To see the  
388 combined length of each linkage group:

389

390 for x in final???.map

391 do

392 echo \${x} \$(tail -n 1 \${x} | cut -f 4)

393 done

394

395 You should see something like:

396

397 final000.map 253.2675

398 final001.map 234.6211

399 final002.map 234.7441

400 final003.map 278.4530

401 final004.map 265.6859

402 final005.map 272.5785  
403 final006.map 268.4784  
404 final007.map 288.5128  
405 final008.map 280.2432  
406 final009.map 256.5174 etc

407

## 408 Real Data From A Polyploid Outcross

409 The file sample\_data/rgxha.loc.gz contains a real dataset from the allo-octoploid cultivated  
410 strawberry *Fragaria x ananassa* from an outcross between Redgauntlet (maternal parent)  
411 and Hapil (paternal parent) cultivars. There are 11587 markers genotyped on 162 progeny  
412 using the Affymetrix Axiom® iStraw90K microarray platform. The markers were called as part  
413 of a batch of 6x96 samples (including data from other, unrelated plants) using the Affymetrix  
414 Power Tools (APT) and SNPolisher pipeline and further filtered to remove extreme  
415 segregation distortion. The marker names indicate the classification assigned to each  
416 marker by APT: PHR (polymorphic, high resolution) indicates three clusters were detected  
417 across all the samples and assigned as the AA, AB and BB genotypes; NMH (no minor  
418 homozygote) indicates only two clusters which were assigned as AA and AB genotypes. The  
419 two numbers after the three letter code indicate the genotype code assigned by APT to the  
420 maternal and paternal parents using the convention 0=AA, 1=AB, 2=BB. For example,  
421 PHR12 indicates a PHR marker where Redgauntlet was called as AB and Hapil as BB. The  
422 rest of the marker name is the iStraw90K probesetid, for example PHR12-89894503  
423 indicates that the original marker probesetid was AX-89894503.

424

425 The script example\_pipelines/build\_rgxha.sh demonstrates a pipeline which builds a map  
426 from the sample data, making use of Crosslink's helper scripts. The helper scripts assist in  
427 calling the appropriate Crosslink program (crosslink\_group, crosslink\_map or crosslink\_pos)  
428 with the appropriate options when only a subset of their full functionality is required, for  
429 example, outputting just a list of the names of redundant markers. Most of the scripts require



430 a configuration file specifying Crosslink parameters, which are provided in  
431 sample\_data/rgxha\_conf and which can be copied and modified as needed. The pipeline  
432 script contains comments explaining each step. To begin open a terminal, create a new  
433 directory and change into it:

434

```
435 mkdir test_build_rgxha
```

```
436 cd test_build_rgxha
```

437

438 To simply run the entire pipeline from start to end, assuming Crosslink has been properly  
439 installed, enter the following:

440

```
441 ${CROSSLINK_PATH}/example_pipelines/build_rgxha.sh
```

442

443 Otherwise, to perform each step of the pipeline manually follow all the steps listed below.  
444 Begin by making new working copies of the helper script configuration files and genotype  
445 data:

446

```
447 cp -r ${CROSSLINK_PATH}/sample_data/rgxha_conf ./conf
```

```
448 zcat ${CROSSLINK_PATH}/sample_data/rgxha.loc.gz > all.loc
```

449

450 The next step is an initial exploratory grouping, outputting one file per provisional linkage  
451 group into a new subdirectory. The helper scripts generally process one genotype file at a  
452 time or else deal with all the genotype files within a named subdirectory (specifically all the  
453 files with .loc filename extension), these being assumed to represent the linkage groups. In  
454 most cases passing the name of a subdirectory to a script will cause it to create the directory  
455 if required and remove existing files ending with .loc to prevent old files from getting mixed  
456 up with new ones. The following takes the all.loc genotype file and splits it into linkage  
457 groups in a new subdirectory called initgrps using a LOD threshold of 7:

458

```
459 cl_group.sh all.loc initgrps 7
```

460

461 This command can be rerun with different LOD values to see how many linkage groups  
462 result. If you have graphical output available from your Crosslink install, you can use  
463 crosslink\_viewer to view the linkage groups. This will be explained in more detail in the next  
464 section, but for now we will ensure that we retain all the intermediate results of the pipeline in  
465 different subdirectories for later viewing. Polyploidy appears to cause a few markers to be  
466 misclassified. Most commonly this is a NMH marker where the classification of the two  
467 clusters as homozygote and heterozygote has been reversed, so that an lmxll marker is  
468 called as an nnxnp or vice versa. This error is much less common for PHR markers.  
469 Crosslink fixes these errors by detecting unexpected linkage between maternal and paternal  
470 markers. Where a group of markers are thus found to be linked above a significant LOD  
471 threshold but contain a mixture of maternal and paternal markers, the rarer kind of marker is  
472 converted to the more common kind (allowing for some markers to be weighted more  
473 strongly than others). This is performed using:

474

```
475 cl_fixtypes.sh all.loc all.loc conf/fixtypes.000
```

476

477 where conf/fixtypes.000 is the configuration file which contains:

478

```
479 CL_GROUP_MINLOD=${CL_GROUP_MINLOD:-6.0}
```

```
480 CL_GROUP_MATPATLOD=${CL_GROUP_MATPATLOD:-10.0}
```

```
481 CL_MATPAT_WEIGHTS=${CL_MATPAT_WEIGHTS:-"01P03"}
```

```
482 CL_GROUP_LOGFILE=${CL_GROUP_LOGFILE:-fixtypes.log}
```

483

484 The helper script configuration files, written in bash, contain default values which can be  
485 overridden in the calling script. For example MYVARIABLE=\${MYVARIABLE:-10} indicates  
486 that if no value is already set, MYVARIABLE should be set to 10 (*not* -10), otherwise the

existing value is used. In this case CL\_GROUP\_MINLOD specifies markers should be grouped at a LOD threshold of 6, CL\_GROUP\_MATPATLOD that anomalous linkage between maternal and paternal markers in the same group should be considered at a LOD threshold of 10, CL\_MATPAT\_WEIGHTS indicates that by default markers should be weighted as 01 (i.e. 1), but markers with names starting with the letter P should be weighted as 03 (i.e. considered as 3 times more reliable) when deciding the true marker type. For example if two lmxll PHR-type markers and five nnxnp NMR-type markers are found to be linked together, this will count as  $2 \times 3 = 6$  votes for lmxll and  $5 \times 1 = 5$  for nnxnp and all the markers will be set as lmxll type (because PHR markers are misclassified more rarely than NMHs). CL\_GROUP\_LOGFILE names the log file that crosslink\_group will output containing the names of the markers whose types were switched. cl\_fixtypes.sh reads markers in from all.loc, processes them using the configuration provided and then outputs to all.loc again, *overwriting* the existing data. Alternatively a different output filename could have been provided to allow retention of the input data. The linkage groups formed internally are not retained and all markers are outputted into the same output file together. This allows different LOD threshold to be used in different steps of the pipeline. The next step forms linkage groups again, outputting to a new subdirectory. This step is simply to allow the effect of the type error correction to be assessed:

505

```
506 cl_group.sh    all.loc    fixgrps    7
```

507

Next the pipeline identifies which markers are redundant, containing only genotype calls all exactly the same as another marker(s). Missing values are not counted as being new information:

511

```
512 cl_findredun.sh    all.loc    all.redun    conf/findredun.000
```

513

514 Input is read from all.loc (which is not modified) and a list of redundant markers output to  
515 all.redun in a two column format. Column one gives the name of the redundant marker,  
516 column two gives the name of the non-redundant marker that represents it.  
517 conf/findredun.000 is the configuration file:

518

```
519 CL_GROUP_MINLOD=${CL_GROUP_MINLOD:-6.0}  
520 CL_GROUP_REDUNLOD=${CL_GROUP_REDUNLOD:-20.0}
```

521

522 where a CL\_GROUP\_MINLOD of 6.0 indicates that linkage groups be formed above a LOD  
523 threshold of 6, and a CL\_GROUP\_REDUNLOD of 20 indicates that markers linked above  
524 this LOD are explicitly checked to see if one can be flagged as redundant with respect to the  
525 other. Only the marker list is output, the grouping is not retained. The next step is to impute  
526 missing genotype calls:

527

```
528 cl_knnimpute.sh all.loc all.loc conf/knnimpute.000
```

529

530 The configuration file contains:

531

```
532 CL_GROUP_MINLOD=${CL_GROUP_MINLOD:-6.0}  
533 CL_GROUP_KNN=${CL_GROUP_KNN:-3}
```

534

535 Genotypes are read in from all.loc and once imputed are output to all.loc overwriting the  
536 original data. A CL\_GROUP\_MINLOD of 6 indicates that linkage groups should be formed  
537 down to a LOD threshold of 6, a CL\_GROUP\_KNN of 3 indicates that each missing  
538 genotype is imputed from the mean value of the 3 most similar markers in the same linkage  
539 group which do not lack the corresponding genotype. The imputed data are output into a  
540 single file, the grouping is not retained. Next just the redundant markers (listed in all.redun)  
541 are excluded from the fully-imputed genotypes (all.loc) and saved as a new file (all.uniq).  
542 The redundant markers will be used again later and are therefore not overwritten:

543

```
544 cl_extract.sh all.loc all.redun all.uniq
```

545

546 Next linkage groups are formed from the unique markers and put in a new subdirectory  
547 called `uniqgrps`:

548

```
549 cl_group.sh all.uniq uniqgrps 7
```

550

551 Because polyploid related errors are causing some markers to be linked strongly to two  
552 different (but homeologous) linkage groups, the data do not split into the expected 28  
553 linkage groups. This causes `crosslink_group` to give up phasing the affected groups, as  
554 phasing information will not propagate to all the markers in those groups, therefore we use  
555 an additional step to force phasing to complete down to a LOD threshold of zero for each  
556 linkage group individually. The phasing between unlinked markers will be meaningless, but  
557 at least all genuinely linked markers should have a usable phase. Here we retain the data  
558 before and after forced phasing (which just runs `crosslink_group` with a LOD threshold of  
559 zero) for later visualisation if desired:

560

```
561 cl_phase.sh uniqgrps phasegrps
```

562

563 Next, we try to automatically detect all the markers which are incorrectly joining two linkage  
564 groups. This step *must* be run on fully phased markers:

565

```
566 cl_detect_crosslg.sh phasegrps crosslg_markers conf/detectcrosslg.000
```

567

568 The configuration file contains:

569

```
570 CL_PARALLEL_JOBS=${CL_PARALLEL_JOBS:-1}
```

```
571 CL_MAP_RANDOMISE=${CL_MAP_RANDOMISE:-0}
```

```

572 CL_GA_SKIPORDER1=${CL_GA_SKIPORDER1:-1}
573 CL_MAP_CYCLES=${CL_MAP_CYCLES:-5}
574 CL_GA_ITERS=${CL_GA_ITERS:-150000}
575 CL_GA_OPTIMISEMETH=${CL_GA_OPTIMISEMETH:-0}
576 CL_GA_USEMST=${CL_GA_USEMST:-5}
577 CL_GA_MINLOD=${CL_GA_MINLOD:-3.0}
578 CL_GA_MSTNONHK=${CL_GA_MSTNONHK:-0}
579 CL_GA_PROBHOP=${CL_GA_PROBHOP:-0.3333}
580 CL_GA_MAXHOP=${CL_GA_MAXHOP:-1.0}
581 CL_GA_PROBMOVE=${CL_GA_PROBMOVE:-0.3333}
582 CL_GA_MAXMOVESEG=${CL_GA_MAXMOVESEG:-1.0}
583 CL_GA_MAXMOVEDIST=${CL_GA_MAXMOVEDIST:-1.0}
584 CL_GA_PROBINV=${CL_GA_PROBINV:-0.5}
585 CL_GA_MAXSEG=${CL_GA_MAXSEG:-1.0}
586 CL_GIBBS_SAMPLES=${CL_GIBBS_SAMPLES:-200}
587 CL_GIBBS_BURNIN=${CL_GIBBS_BURNIN:-5}
588 CL_GIBBS_PERIOD=${CL_GIBBS_PERIOD:-1}
589 CL_GIBBS_PROBSEQUEN=${CL_GIBBS_PROBSEQUEN:-0.0}
590 CL_GIBBS_PROBUNIDIR=${CL_GIBBS_PROBUNIDIR:-1.0}
591 CL_GIBBS_MINPROB1=${CL_GIBBS_MINPROB1:-0.1}
592 CL_GIBBS_MINPROB2=${CL_GIBBS_MINPROB2:-0.0}
593 CL_GIBBS_TWOPT1=${CL_GIBBS_TWOPT1:-1.0}
594 CL_GIBBS_TWOPT2=${CL_GIBBS_TWOPT2:-1.0}
595 CL_HOME_O_MINCOUNT=${CL_HOME_O_MINCOUNT:-2}
596 CL_HOME_O_MINLOD=${CL_HOME_O_MINLOD:-1}
597 CL_HOME_O_MAXLOD=${CL_HOME_O_MAXLOD:-25}

```

598

599 This step uses `crosslink_map` internally to impute the missing information in the `hkxhk`  
600 markers before detecting those joining two linkage groups. `CL_PARALLEL_JOBS=1`  
601 indicates to only process one linkage group at a time. Increase the value to utilise more than  
602 one CPU core. The many options relating to marker ordering and `hk` genotype imputation will  
603 be described in detail in a later section. The three final options relate directly to detection of  
604 cross linkage group markers. The problem appears to be the result of a few `hkxhk` markers

605 whose maternal information comes from one linkage group while the paternal information  
606 comes from one of the homeologous linkage groups. That is to say that the marker's probes  
607 seem to be hybridising to two or more homeologous loci, one of which is heterozygous in the  
608 maternal genome and another in the paternal genome. These cases are detected by  
609 comparing pairs of hxxhk markers, where we expect the markers to be truly linked according  
610 to one parent but unlinked according to the other. For a pair of hxxhk markers if one parent  
611 indicates a LOD below CL\_HOME0\_MINLOD and the other a LOD above  
612 CL\_HOME0\_MAXLOD then both markers are given one point. Markers accumulate points  
613 for each qualifying hxxhk marker pair they are involved in. Those that receive a total number  
614 of points greater than or equal to CL\_HOME0\_MINCOUNT are flagged as being cross  
615 linkage group markers, and listed in the output file (here we have called it crosslg\_markers).  
616 The next three steps remove the flagged markers then regroup and rephase the data:

617

```
618 cl_removemarkers.sh all.uniq filt.uniq crosslg_markers
619 cl_group.sh filt.uniq filtgrps 7
620 cl_phase.sh filtgrps filtgrps
```

621

622 There are still 27 rather than the expected 28 linkage groups. If you have graphical output  
623 available, crosslink\_viewer can be used to view each linkage group in turn (explained in  
624 detail in the next section), which should make it fairly obvious which group needs to be split.  
625 There should be a single hxxhk marker holding the two true linkage groups together. Zoom  
626 in using the cursor keys and plus/minus keys to find the marker. Clicking on the marker  
627 should print its name to the terminal. We can then manually add it to the list of cross linkage  
628 group markers and repeat the filtering and regrouping steps. If you don't have  
629 crosslink\_viewer available just follow these steps below anyway for now:

630

```
631 echo 'PHR11-89834490' >> crosslg_markers
632 cl_removemarkers.sh filt.uniq filt.uniq crosslg_markers
633 cl_group.sh filt.uniq filtgrps 7
```

634 `cl_phase.sh filtgrps filtgrps`

635

636 There should now be 28 linkage groups. To produce the final map ordering use the  
637 following:

638

639 `cl_order_hkimpute.sh filtgrps finalgrps conf/orderhkimpute.000`

640

641 which will output the final markers orderings with imputed hk genotypes into the new  
642 subdirectory finalgrps. The configuration file contains:

643

644 `CL_PARALLEL_JOBS=${CL_PARALLEL_JOBS:-1}`

645 `CL_MAP_RANDOMISE=${CL_MAP_RANDOMISE:-0}`

646 `CL_GA_SKIPORDER1=${CL_GA_SKIPORDER1:-1}`

647 `CL_MAP_CYCLES=${CL_MAP_CYCLES:-5}`

648 `CL_GA_ITERS=${CL_GA_ITERS:-150000}`

649 `CL_GA_OPTIMISEMETH=${CL_GA_OPTIMISEMETH:-0}`

650 `CL_GA_USEMST=${CL_GA_USEMST:-5}`

651 `CL_GA_MINLOD=${CL_GA_MINLOD:-3.0}`

652 `CL_GA_MSTNONHK=${CL_GA_MSTNONHK:-0}`

653 `CL_GA_PROBHOP=${CL_GA_PROBHOP:-0.3333}`

654 `CL_GA_MAXHOP=${CL_GA_MAXHOP:-1.0}`

655 `CL_GA_PROBMOVE=${CL_GA_PROBMOVE:-0.3333}`

656 `CL_GA_MAXMOVESEG=${CL_GA_MAXMOVESEG:-1.0}`

657 `CL_GA_MAXMOVEDIST=${CL_GA_MAXMOVEDIST:-1.0}`

658 `CL_GA_PROBINV=${CL_GA_PROBINV:-0.5}`

659 `CL_GA_MAXSEG=${CL_GA_MAXSEG:-1.0}`

660 `CL_GIBBS_SAMPLES=${CL_GIBBS_SAMPLES:-200}`

661 `CL_GIBBS_BURNIN=${CL_GIBBS_BURNIN:-5}`

662 `CL_GIBBS_PERIOD=${CL_GIBBS_PERIOD:-1}`

663 `CL_GIBBS_PROBSEQUEN=${CL_GIBBS_PROBSEQUEN:-0.0}`

664 `CL_GIBBS_PROBUNIDIR=${CL_GIBBS_PROBUNIDIR:-1.0}`

665 `CL_GIBBS_MINPROB1=${CL_GIBBS_MINPROB1:-0.1}`



666 CL\_GIBBS\_MINPROB2=\${CL\_GIBBS\_MINPROB2:-0.0}

667 CL\_GIBBS\_TWOPT1=\${CL\_GIBBS\_TWOPT1:-1.0}

668 CL\_GIBBS\_TWOPT2=\${CL\_GIBBS\_TWOPT2:-1.0}

669

670 Again CL\_PARALLEL\_JOBS controls how many linkage groups will be processed in parallel  
671 using multiple CPU cores if available. CL\_MAP\_CYCLES=5 means to run 5 cycles of marker  
672 ordering and hk imputation. CL\_MAP\_RANDOMISE=0 and CL\_GA\_SKIPORDER1=1  
673 means to skip the very first round of marker ordering, retaining the approximate ordering of  
674 the input data produced previously by crosslink\_group and proceed straight to hk imputation.  
675 For subsequent rounds marker ordering is performed as normal. CL\_MAP\_RANDOMISE=1  
676 would cause the program to ignore the order of the markers in the input file and start the first  
677 round of ordering from a random initial order (this option cannot be used with  
678 CL\_GA\_SKIPORDER1=1, otherwise randomly ordered markers would be passed to the hk  
679 imputation step). The remaining options will be explained in detail in the reference section of  
680 this document. In summary, this step produces a final marker ordering for each linkage  
681 group and also imputes the likely parental origin of the h and k allele for all hk genotypes.  
682 The output genotype files contain the markers in map order with hk genotypes listed  
683 explicitly as either hk or kh. The final three commands produce the final map positions of the  
684 markers, and produce a redundant version of the genotype and map file for each linkage  
685 group where the redundant markers have been reinserted at the position of their  
686 representative marker:

687

688 cl\_mappos.sh finalgrps finalgrps

689 cl\_reinsert\_loc.sh finalgrps all.loc all.redun finalredun conf/reinsert.000

690 cl\_reinsert\_map.sh finalgrps all.redun finalredun

691

692 The configuration file contains:

693

694 CL\_GIBBS\_SAMPLES=\${CL\_GIBBS\_SAMPLES:-200}

```

695 CL_GIBBS_BURNIN=${CL_GIBBS_BURNIN:-5}
696 CL_GIBBS_PERIOD=${CL_GIBBS_PERIOD:-1}
697 CL_GIBBS_PROBSEQUEN=${CL_GIBBS_PROBSEQUEN:-0.0}
698 CL_GIBBS_PROBUNIDIR=${CL_GIBBS_PROBUNIDIR:-1.0}
699 CL_GIBBS_MINPROB1=${CL_GIBBS_MINPROB1:-0.1}
700 CL_GIBBS_MINPROB2=${CL_GIBBS_MINPROB2:-0.0}
701 CL_GIBBS_TWOPT1=${CL_GIBBS_TWOPT1:-1.0}
702 CL_GIBBS_TWOPT2=${CL_GIBBS_TWOPT2:-1.0}

```

703

704 and controls the process of imputing the hk genotypes of the reinserted markers. The  
 705 version of the map containing only unique markers is now in the finalgrps subdirectory, with  
 706 the redundant version in finalredun.

## 707 **Data Visualisation**

708 crosslink\_viewer is the main visualisation tool, which creates coloured plots showing linkage  
 709 LOD and map distances between all the markers. To use it you must have either  
 710 successfully compiled Crosslink's data visualisation programs using the viewer\_make.sh  
 711 script during the installation procedure, or be using the Crosslink virtual machine inside  
 712 VirtualBox or have set up graphical output from the Crosslink docker image.  
 713 crosslink\_viewer can display a single linkage group, a subset of a linkage group or multiple  
 714 linkage groups at once. It can be set to begin centred on a named marker of interest, and  
 715 clicking on the plot will output information about the markers to the terminal. The plot itself  
 716 conveys information only through the colour and brightness of the pixels. For this reason it  
 717 makes full use of the red, green and blue colour channels. An alternative colour scheme is  
 718 available by editing the src/viewer\_make.sh script and uncommenting (i.e. deleting the initial  
 719 # from) the line AUX="-DALTCOLSCHEME", changing into the src directory and running the  
 720 script as ./viewer\_make.sh to recompile. This scheme may be more useful for people with  
 721 red-green colour blindness. The file src/crosslink\_viewer.h can also be edited to change the  
 722 definition of the function setpixelrgb by simply rearranging the position of the variables r, g  
 723 and b. The descriptions below assume the default colour scheme.

724

725 The visualisation programs can work your computer's graphical system quite hard,  
726 especially for large datasets, and in my experience are more likely to cause crashes or other  
727 problem than the rest of Crosslink. For this reason make sure you have saved any other  
728 work you are doing before running them.

729

730 crosslink\_viewer works by loading the marker data, interpreting it as either unphased,  
731 phased or imputed depending on the --datatype option, then rendering three versions of the  
732 plot: maternal, paternal and combined information. The plots are fully rendered in memory  
733 before being displayed on screen, which can cause a considerable delay before anything  
734 appears on screen for large datasets. An all-versus-all marker comparison is performed,  
735 meaning that for  $n$  markers on the order of  $n^2$  memory and time will be required. The plots  
736 are created at full resolution, using the graphics card for scaling, panning and zooming. For  
737 this reason you may wish to limit the number of markers you load at once, at least to begin  
738 with, or use a computer with more video memory.

739

740 To illustrate data visualisation we will go through most of the steps of the  
741 example\_pipelines/build\_rgxha.sh pipeline again, but using only one of the linkage group  
742 files as an example. Recall that the first step of the Redgauntlet x Hapil example pipeline  
743 formed linkage groups using a simple LOD threshold, and that 21 instead of the expected 28  
744 linkage groups were formed. This was caused by some hkxhk markers which had strong  
745 linkage to two different linkage groups, resulting in two or more true linkage groups  
746 remaining joined together. The example file we will use here is sample\_data/000.loc.gz  
747 which was created by running cl\_group.sh on the sample dataset sample\_data/rgxha.loc.gz  
748 with a LOD threshold of 7. The file 000.loc grouped correctly at the first try. After running  
749 through all three of the quick start guide sections you will have the knowledge to visualise  
750 the groups which did not group correctly to begin with and follow them through the steps of  
751 the Real Data example pipeline example\_pipelines/build\_rgxha.sh. We will begin by

752 processing 000.loc through all the necessary steps. Starting from a clean working directory  
753 as follows:

754

```
755 cd ~  
756 mkdir test_crosslink_vis  
757 cd test_crosslink_vis
```

758

759 Then run the data through the required pipeline steps (note that because the grouping is  
760 already correct we do not need to run cl\_detect\_crosslg.sh or cl\_removemarkers.sh at any  
761 point) by entering the following commands:

762

```
763 cp -r ${CROSSLINK_PATH}/sample_data/rgxha_conf ./conf  
764 zcat ${CROSSLINK_PATH}/sample_data/000.loc.gz > 000.loc  
765 cl_fixtypes.sh 000.loc 000.fix conf/fixtypes.000  
766 cl_findredun.sh 000.fix 000.redun conf/findredun.000  
767 cl_knnimpute.sh 000.fix 000.imp conf/knnimpute.000  
768 cl_extract.sh 000.imp 000.redun 000.uniq  
769 mkdir -p 000.dir  
770 cp 000.uniq 000.dir/000.loc  
771 cl_order_hkimpute.sh 000.dir 000.final conf/orderhkimpute.000  
772 cl_mappos.sh 000.final 000.final  
773 cl_reinsert_loc.sh 000.final 000.imp 000.redun 000.finalredun  
774 conf/reinsert.000  
775 cl_reinsert_map.sh 000.final 000.redun 000.finalredun
```

776

777 Or run the same commands using the provided script:

778

```
779 ${CROSSLINK_PATH}/example_pipelines/build_one_group.sh
```

780

781 The initial file 000.loc contains the first linkage group output by crosslink\_group when run on  
782 the sample dataset rgxha.loc. The markers are already phased and approximately ordered.  
783 To view only the first 100 markers of this linkage group use the following:

784

```
785 crosslink_viewer --inp=000.loc --datatype=phased --total=100
```

786

787 This plot should display at the default window size of 1000x1000 pixels. If the window is too  
788 large or small for your monitor press 'q' to quit and restart using the --window\_size option to  
789 adjust the window size, for example to use an 800x800 pixel window:

790

```
791 crosslink_viewer --inp=000.loc --datatype=phased --total=100 --window_size=800
```

792

793 The cursor keys pan the plot, plus and minus zoom in and out, and ENTER re-centres the  
794 plot. The combined plot is shown first, press 'm' or 'p' to switch to the maternal or paternal  
795 plots and 'c' to return to the combined view. To view the next 100 markers in the linkage  
796 group we would use:

797

```
798 crosslink_viewer --inp=000.loc --datatype=phased --skip=100 --total=100
```

799

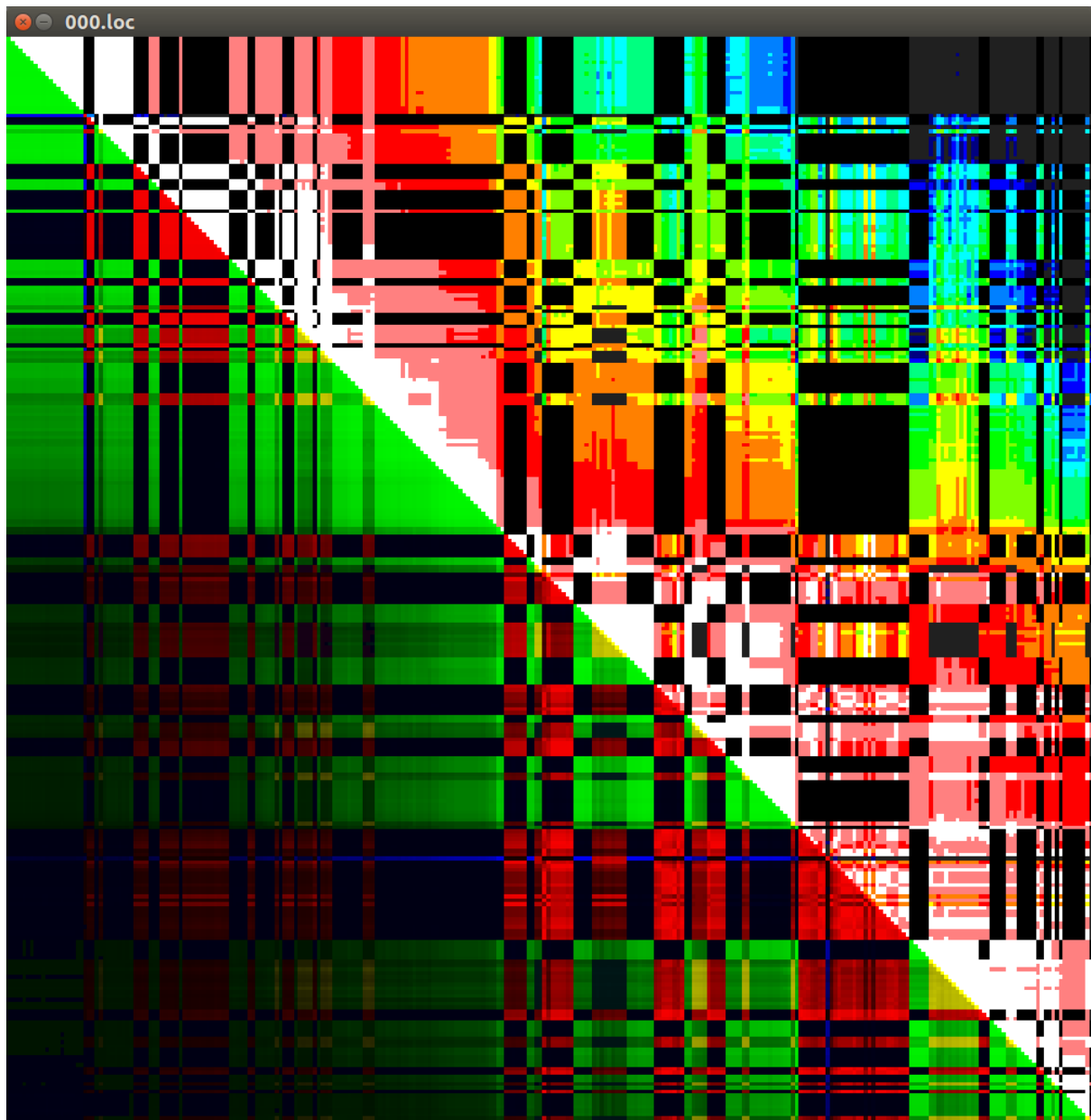
800 To view all the markers at once:

801

```
802 crosslink_viewer --inp=000.loc --datatype=phased
```

803

804 which should look like this:



805

806 The markers are represented as coloured squares along the main diagonal with the first at  
 807 the top left and the last at bottom right. Maternal (lmxll) markers are red, paternal (nnxnp)  
 808 green and shared (hxxhk) yellow. These squares are always shown at full brightness. Click  
 809 on a marker, making sure to click only on one of the squares along the main diagonal (you  
 810 may need to zoom in to be sure you have clicked on the marker you intend) and information  
 811 about the marker will appear in the terminal, for example:

812

813 Xaxis: (0)NMH01-89891327 nnxnp {-0} Yaxis: (0)NMH01-89891327 nnxnp {-0}\

814 Comp: pat:N=162 rf=0.00000 lod=48.77

815

816 This tells us the linkage group (0), the marker name (NMH01-89891327), type (nnxnp) and  
817 phase ({-0}). In this case we get the same information twice, because, as we clicked on the  
818 main diagonal, both the row and column we clicked on correspond to the same marker. We  
819 therefore get a comparison of the marker with itself, telling us the paternal information (pat)  
820 has N=162 non missing genotype values with an rf of zero (of course) and a LOD of 48.77.  
821 There is no maternal information as this is an nnxnp marker. Now click away from the main  
822 diagonal. We get information comparing the markers corresponding to the row and column  
823 that we clicked on, for example:

824

825 Xaxis: (0)NMH01-89891327 nnxnp {-0}...

826 Yaxis: (0)PHR11-89818133 hkxhk {00}...

827 Comp: pat:N=76 rf=0.03947 lod=17.39

828

829 This indicates that there were N=76 genotypes available for the comparison, (before hk  
830 genotypes have been imputed they are not available and must be treated as missing),  
831 showing an rf of 0.03947 corresponding to a LOD of 17.39. Below the main diagonal the  
832 square colour shows information about the rf, LOD and phasing between the two markers. In  
833 general, to find the square showing the rf/LOD/phasing information between two markers,  
834 find the first (higher) of the two markers and move down its column until you reach the row  
835 corresponding to the second of the two markers. This square's red channel shows the  
836 maternal information and the green channel the paternal. The LOD is represented by the  
837 brightness, so a brighter square means a larger LOD. Squares corresponding to a pair of  
838 Imxll markers therefore indicate the LOD between them as the red brightness, squares  
839 corresponding to a pair of nnxnp markers as the green brightness and squares  
840 corresponding to a pair of hkxhk markers as the red and green brightness, resulting in a  
841 yellow square. Comparing an Imxll with an hkxhk allows only the maternal information of the  
842 hkxhk to be used, therefore the square will be red, likewise an nnxnp and an hkxhk

843 comparison can only use the paternal information and is represented as a green square.  
844 Comparing an Imxll with an nnxnp marker does not normally make sense, as the markers  
845 should not show any linkage. However, due to the polyploid nature of the strawberry, some  
846 markers are misclassified as Imxll when they are actually nnxnp and vice versa. To visualise  
847 this phenomenon crosslink\_viewer *does* perform a linkage analysis between the maternal  
848 information of all Imxll and the paternal information of all nnxnp markers. These squares  
849 show the linkage LOD with the blue brightness, making it easier to pick out these likely  
850 errors. Blue is also used to indicate likely phasing errors (for comparisons other than Imxll  
851 versus nnxnp). When treating the data as phased by passing option --datatype=phased, if  
852 the rf value between two markers is in disagreement with their declared phases, this may  
853 indicate a phasing error. For example, if two linked Imxll markers have been assigned  
854 different phases (i.e. one is {0-} and the other is {1-}) then we expect the rf to be greater than  
855 0.5, conversely if they have the same phase we expect the rf to be less than 0.5. If the rf is  
856 found to be in agreement with the phase of the two markers then the usual colour channel is  
857 used, whereas if a disagreement is found the colour channel is switched to blue. Therefore  
858 an Imxll to Imxll square would be blue instead of red and an nnxnp to nnxnp blue rather than  
859 green. For an hkxhk to hkxhk comparison there are two independent phases. If both checks  
860 fail the square changes from yellow to blue, if only the maternal phase check fails the red  
861 channel changes to blue but the green channel stays green resulting in a cyan colour, if only  
862 the paternal check fails the green channel changes to blue resulting in a purple colour.

863

864 If you zoom in a little and pan along the main diagonal you should notice that three of the  
865 markers are showing bright blue squares against some of the other markers. Click on some  
866 of the blue squares below the main diagonal and see if you can work out what is going on  
867 from the rf, LOD and marker type information displayed to the terminal. Three Imxll markers  
868 are showing a strong linkage to many of the nnxnp markers, causing the corresponding  
869 squares to appear bright blue, but lack any strong linkage to other Imxll markers, causing the  
870 corresponding squares to appear almost black, all of which indicates they are likely



871 misclassified nnxnp markers. There are no other bright blue squares, suggesting that the  
872 remaining markers do not have any obvious phasing errors.

873

874 **Summary of crosslink\_viewer colour scheme (if compiled using default colour mode)**  
875 **when viewing phased data, for the bottom left half of the plot which shows**  
876 **rf/LOD/phasing information. This applies to combined mode (press 'c' to switch to**  
877 **this mode). Increasing brightness indicates increasing linkage LOD.**

Marker 1	Marker 2	Correct Phasing	Incorrect Phasing
lmxll	lmxll	red	blue
nnxnp	nnxnp	green	blue
hkxhk	hkxhk	both correct => yellow	maternal error => cyan paternal error => purple both in error => blue
lmxll	hkxhk	red	blue
nnxnp	hkxhk	green	blue
If <i>one</i> of the two markers has been incorrectly typed:			
lmxll	nnxnp	blue	blue

878

879 If we are viewing fully hk imputed data using the --datatype=imputed option the colour  
880 scheme is the same as the above. If we are viewing unphased data using  
881 the --datatype=unphased option then the colour scheme is interpreted slightly differently.  
882 Because there are no declared phases to be check now the blue channel is simply used to  
883 represent where the rf value is greater than 0.5 indicating a likely repulsion phase. So in the  
884 table above we simply change the meaning of “correct phasing” to “coupling phase” and  
885 “incorrect phasing” to “repulsion phase”. So for an hkxhk to hkxhk comparison a yellow  
886 square would indicate coupling phase in both maternal and paternal genomes for example. If

887 viewing in maternal or paternal mode (by pressing 'm' or 'p') then only information about the  
888 respective parent will be shown so, for example, in maternal mode an hxxhk to hxxhk square  
889 will appear as red, showing only information about the maternal part of the genotypes.

890  
891 The bottom left half of the plot therefore shows information about rf, LOD and phasing, and  
892 is useful for detecting grouping, phasing and marker typing errors. However the implied map  
893 distance between two markers is not linearly related to the LOD, making subtle problems  
894 with marker ordering difficult to spot. Above the diagonal a colour scheme is used to  
895 represent the centimorgan distance between the markers which is designed to make  
896 problems with the marker ordering easier to see. The rf value is converted into a  
897 centimorgan distance using the Haldane mapping function and displayed on a heatmap  
898 scale where white represents the shortest distances and grey the longest, with black  
899 indicating no information (between lmxll and nxxnp markers).

900  
901 **Colour scheme used in the top right half of the crosslink\_viewer plot, which displays**  
902 **map distance information, calculated using the Haldane mapping function.**

Colour	Distance Range (cM)
white	< 10
pink	10 - 20
red	20 - 30
orange	30 - 40
yellow	40 - 50
yellow-green	50 - 60
green	60 - 70
green-cyan	70 - 80
cyan	80 - 90
cyan-blue	90 - 100

blue	100 - 110
dark blue	110 - 120
grey	> 120
black	no information

903

904 000.fix contains the same markers after typing errors were fixed using cl\_fixtype.sh, view it  
905 now using:

906

907 `crosslink_viewer --inp=000.fix --datatype=phased`

908

909 which should look the same as the previous plot except that the erroneously typed markers  
910 have now been recoded as the correct type. To check this view the uncorrected plot, click on  
911 an affected marker to get its name, then view the corrected file using the above command  
912 but add the --marker option to specify that the affected marker should appear in the centre of  
913 the view, e.g.:

914

915 `crosslink_viewer --inp=000.fix --datatype=phased --marker=NMH10-89914030`

916

917 The next steps identified redundant markers (cl\_findredun.sh), imputed missing genotype  
918 calls (cl\_knnimpute.sh) and extracted the imputed versions of the non-redundant markers  
919 (cl\_extract.sh) producing 000.uniq which contains 126 instead of 284 markers, and contains  
920 no missing calls. View it using:

921

922 `crosslink_viewer --inp=000.uniq --datatype=phased`

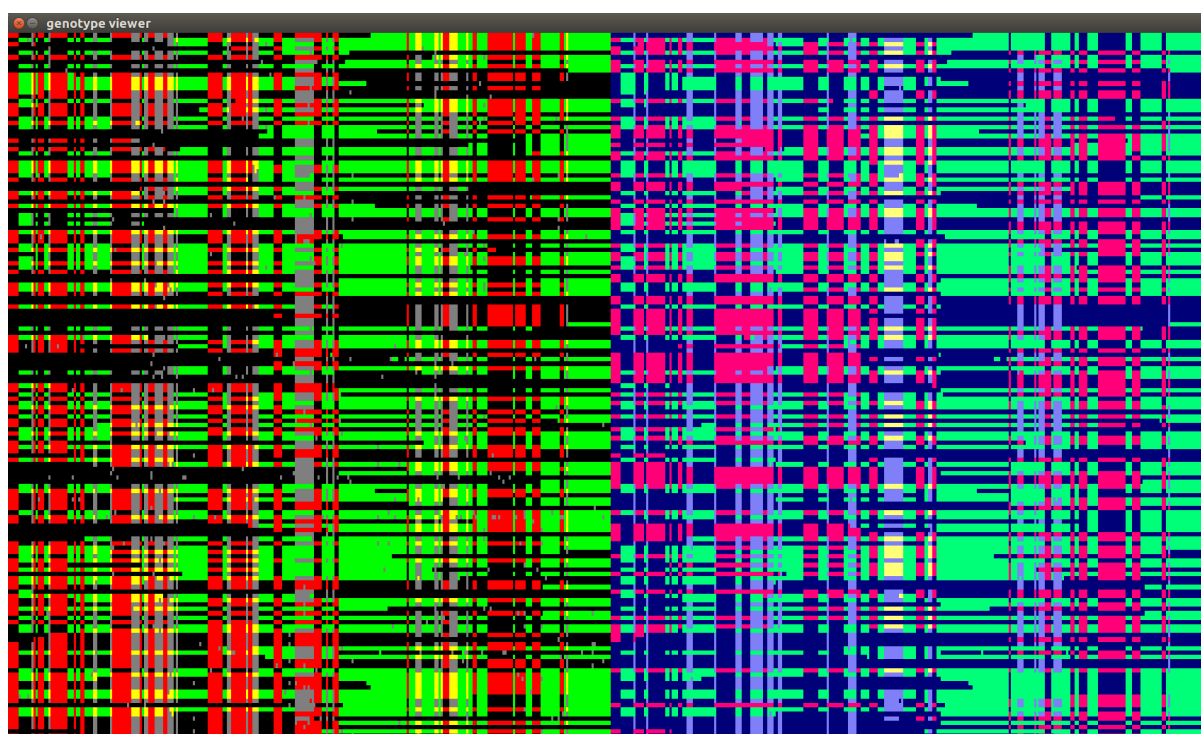
923

924 To compare the data before and after missing calls have been imputed we can use the  
925 crosslink\_graphical command, which displays a simple colour coded representation of the  
926 genotype values. The following command will open 000.fix and 000.imp (the data before and

after missing value imputation) side-by-side in a window 1700x1000 pixels. You may wish to adjust the size to fit your monitor:

```
crosslink_graphical --inp="000.fix 000.imp" --datatype=phased\  
--size=1700 --sizey=1000
```

which should look like this (this figure shows the default combined-phased mode):



Each row represents a progeny and each column a marker. The slight blue colouration on the right indicates that these columns are the data from the second genotype file (000.imp). Use 'm', 'p' and 'c' to change between maternal, paternal and combined views, and 'z' to which between phased and unphased views. In *unphased* view mode a red or green colour represents an m or p allele for an lmxll or nnxnp marker respectively, whereas a black or dark blue rectangle represents the complementary allele (l or n); and for hkxhk markers yellow, red, green and black/dark blue represent the genotypes kk, kh, hk and hh respectively. For data lacking imputed hk information, as in the present example, grey is used to represent hk/kh genotypes. In *phased* mode red and green represent a phase 0 allele m or p or a phase 1 allele l or n. For hkxhk markers the red and green channels

945 represent the maternal and paternal information, so the red channel is activated if there is a  
946 maternal phase 0 k or phase 1 h, and the green channel activated if there is a paternal  
947 phase 0 k or phase 1 h. If red and green are active a yellow colour results, if neither are  
948 active a black / dark blue colours results. Grey also represents any missing genotype calls.

949

950 As with `crosslink_viewer` plus/minus perform zooming, cursor keys pan and ENTER  
951 recentres the view. Shift combined with a cursor key can be used to shrink or grow the plot  
952 in one direction only. You should be able to see a few grey squares in the left plot (000.fix),  
953 these represent the missing values which have all been imputed in the right plot. Identifying  
954 markers/progeny by clicking on them is not yet implemented in this program.

955 As a quirk of the way that the helper script pipeline works by calling `crosslink_group`  
956 internally, the process of imputing missing values actually also repeats the process of  
957 phasing and approximate ordering. (The advantage of breaking the `crosslink_group`  
958 functionality down into separate steps is that the full dataset can be used to impute missing  
959 values at the same time as identifying redundant markers in the non-imputed data, so that  
960 markers with less missing data are retained preferentially). You will therefore be able to see  
961 in the above figure that the maternal information in 000.fix is phased using the opposite  
962 convention to those in 000.imp (i.e. phase 0 in 000.fix is called phase 1 in 000.imp), whereas  
963 the paternal markers have been assigned the same phasing convention in both files (you  
964 may have experienced a different outcome when you ran the pipeline, as a small degree of  
965 randomness can arise in imputing missing values). The markers are also in a slightly  
966 different order. To get around this problem press 'm' to switch to maternal-only view, then  
967 press 'z' to switch to unphased view. Now both datasets should look almost the same,  
968 except for the slight difference in ordering.

969

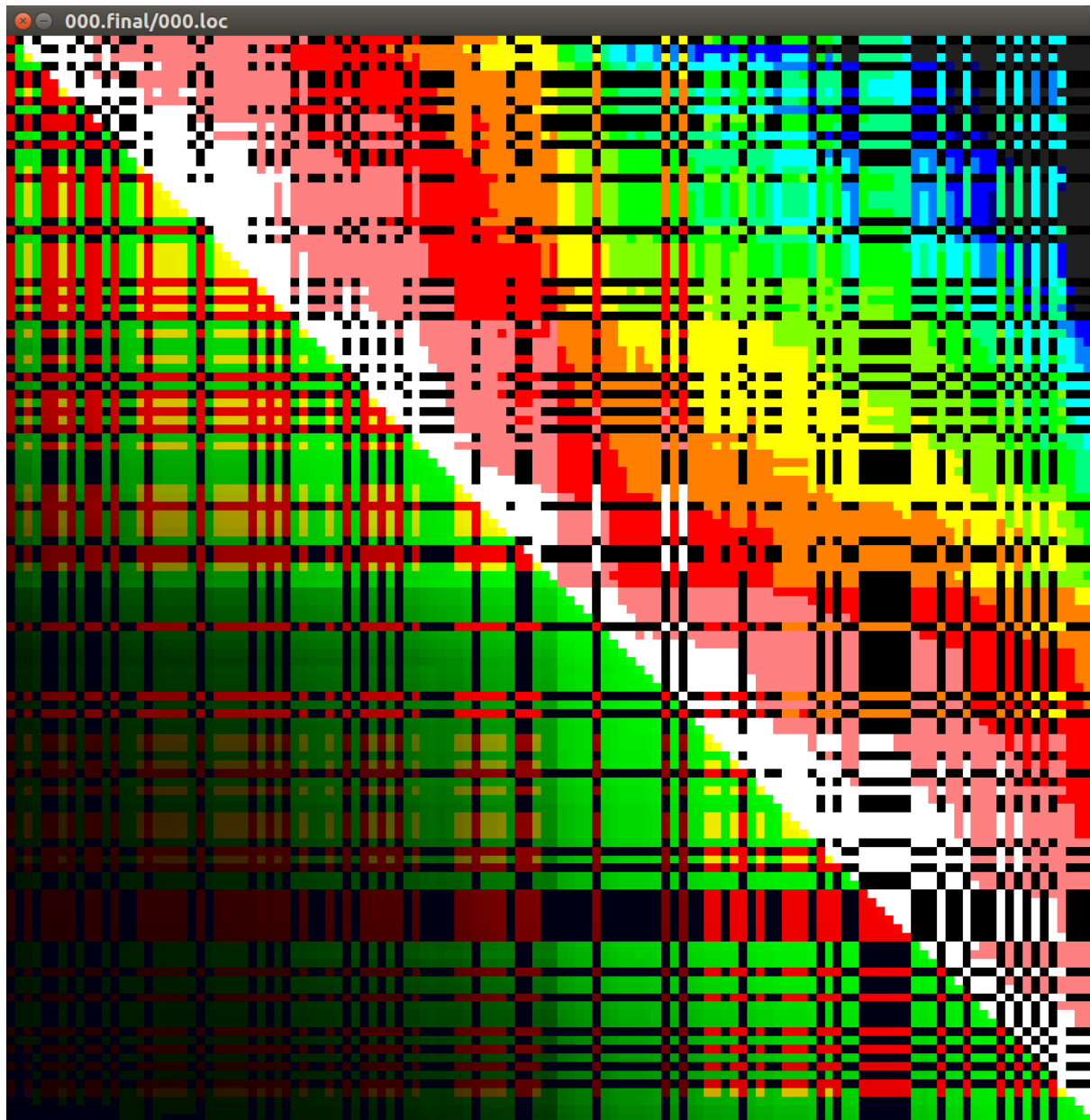
970 The final version of the non-redundant markers was produced using `cl_order_hkimpute.sh` to  
971 refine the map order and impute the hk genotype missing information. View this version of  
972 the file using:

973

974 `crosslink_viewer --inp=000.final/000.loc --datatype=imputed`

975

976 which should look like this:



977

978 To check the ordering switch to maternal-only view by pressing 'm' and check that the map  
979 distances increase monotonically away from the main diagonal, then repeat this process for  
980 the paternal map by pressing 'p'.

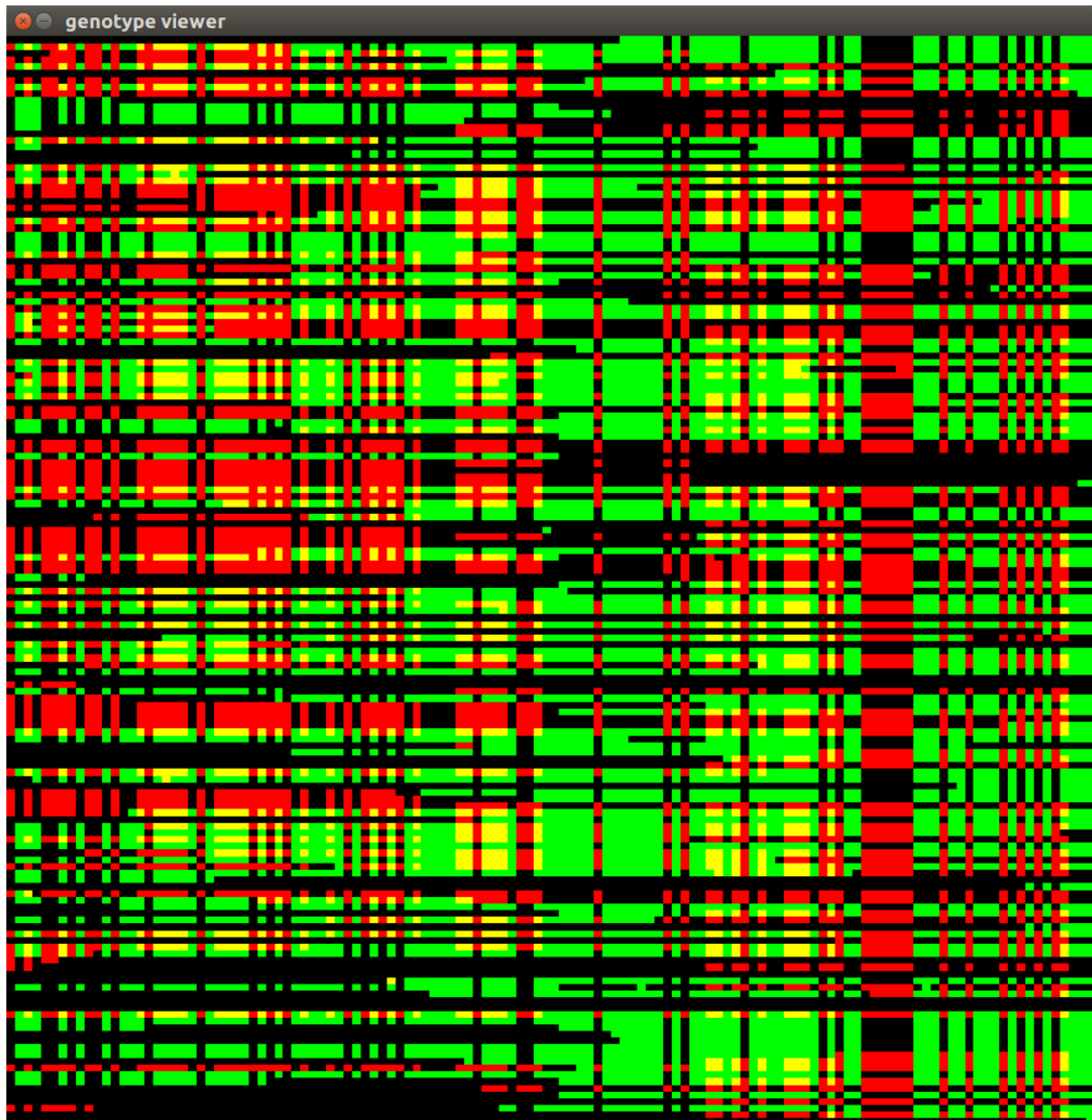
981 View the same data in `crosslink_graphical` using:

982

983 `crosslink_graphical --inp=000.final/000.loc --datatype=imputed`

984

985 which should look like this:



986

987 Switch between combined, maternal and paternal views (by pressing 'c', 'm' and 'p') to check  
988 that the ordering and hk imputation seem correct (there will be a few remaining errors as  
989 Crosslink does not currently support error correction). Finally the redundant markers were  
990 reinserted into the ordering producing the redundant version of the final ordering, viewable  
991 using:

992

```
993 crosslink_viewer --inp=000.finalredun/000.loc --datatype=imputed
```

994

995 As well as viewing single files, crosslink\_viewer can display multiple genotype files at the  
996 same time. To view all the linkage groups from the build\_rgxha.sh pipeline at the same time  
997 (be aware this will use a lot more video memory than the previous single linkage group  
998 examples) we would use a command like this executed in the directory where you ran the  
999 full build\_rgxha.sh pipeline:

1000

```
1001 crosslink_viewer --inp="$(echo initgrps/*.loc)" --datatype=phased
```

1002

1003 To view each linkage group one at a time:

1004

```
1005 for x in initgrps/*.loc
```

```
1006 do
```

```
1007     crosslink_viewer --inp=${x} --datatype=phased || break
```

```
1008 done
```

1009

1010 In this example pressing 'q' quits the current linkage group and moves on to the next one  
1011 whereas pressing 'escape' breaks out of the for loop without viewing anymore. This works  
1012 because crosslink\_viewer returns an exit status (accessible in bash as the \$? variable) of 0 if  
1013 'q' is pressed, but a status of 100 if 'escape' is pressed. If you have a large batch of file to  
1014 view and want to automatically process each one differently depending on a decision made  
1015 by a user viewing the files, crosslink\_viewer also exits when any of the number keys 0-9 are  
1016 pressed with an exit status of ten plus the number of the key pressed.

1017

## 1018 **Export To JoinMap® Compatible Files**

1019 Given a subdirectory containing the final genotype (\*.loc) and map (\*.map) files, the following  
1020 commands will export the data into JoinMap compatible loc and map files:



1021

1022 `cl_loc2joinmap.sh`    `final_lgs`    `jm.loc`

1023 `cl_map2joinmap.sh`    `final_lgs`    `jm.map`

1024

## 1025 **Commands and Options**

### 1026 **crosslink\_group**

1027     `--inp=STRING` input genotype file (required)

1028     `--outbase=STRING` basename for output genotype files (optional)

1029     `--mapbase=STRING` basename for output map files (optional)

1030     `--redun=STRING` filename for outputting marker redundancy information (optional)

1031     `--log=STRING` filename for outputting logging information (optional)

1032     `--seed=UNSIGNED` random number generator seed, 0=use system time (default: 1)

1033     `--map_func=UNSIGNED` mapping func, 1=Haldane,2=Kosambi (default: 1)

1034     `--randomise_order=UNSIGNED` start from a random initial marker ordering

1035         (default: 0)

1036     `--bitstrings=UNSIGNED` use bitstring data representation internally (default: 1)

1037     `--matpat_lod=FLOAT` minimum LOD used to identify spurious linkage between

1038         maternal and paternal markers, 0.0=disable (default: 0.0)

1039     `--matpat_weights=STRING` conditional weightings to give markers when correcting

1040         marker typing errors

1041         eg 01P03L05 gives default weight of 1 but 3 to markers starting with P and 5

1042         for those starting with L (default: 01)

1043     `--min_lod=FLOAT` minimum linkage LOD to use when forming linkage groups

1044         (default: 3.0)

1045     `--em_tol=FLOAT` for 2 point rf calculations, convergence tolerance for EM

1046         algorithm (default: 1e-5)

1047     `--em_maxit=UNSIGNED` for 2 point rf calculations, max EM iterations (default:

1048         100)

1049     `--knn=UNSIGNED` how many nearest neighbours to use for kNN missing data

1050         imputation, 0=disable imputation (default: 0)

1051     `--ignore_cxr=UNSIGNED` 1=ignore cxr and rxc linkage during grouping (default: 0)

1052     `--redundancy_lod=FLOAT` minimum linkage LOD to use when identifying redundant

1053         markers, 0.0=disable (default: 0.0)

1054

1055 This program, implemented in C, performs the first part of the mapping pipeline, consisting  
1056 of: Two-point rf and LOD Calculation, Form Linkage Groups, Fix Marker-Typing Errors,  
1057 Phasing, Impute Missing Values, Approximate Marker Ordering and Combined Map  
1058 Positions.

1059

1060 Data is read from the genotype file named by required option --inp. Option --seed is used to  
1061 seed the integer pseudo-random number generator using the C function call srand. The  
1062 floating point pseudo-random number generator is then seeded using srand48(rand()). A  
1063 seed value of zero causes the microsecond resolution system time to be used instead via a  
1064 gettimeofday function call, giving different random values for each invocation. The default  
1065 seed is 1, causing the same behaviour each invocation, provided the genotype data is  
1066 identical and no other options are changed. Option --bitstrings=1 causes the genotype calls  
1067 to be encoded in memory using a compact bitstring representation which may execute faster  
1068 on some machines (enabled by default, using --bitstrings=0 to disable).  
1069 Option --randomise\_order=1 causes the order of the genotypes to be randomised as soon  
1070 as they are loaded (disabled by default). This may occasionally leading to a better  
1071 approximate ordering of the markers for some datasets. Note this will be the same “random”  
1072 ordering every time if the --seed option is left at the default. If provided logging messages  
1073 are output for the file named by option --log, otherwise no log messages are generated.

1074

## 1075 (1) Two-point rf and LOD Calculation

1076 This step estimates rf and linkage LOD between all pairs of markers using the so-called two-  
1077 point rf method, by treating all hk genotypes of hkxhk markers as missing. Estimating rf  
1078 values between pairs of hkxhk markers involves using an iterative expectation-maximisation  
1079 algorithm. Options --em\_tol and --em\_maxit control the tolerance and maximum iterations  
1080 respectively. Any LOD found to be less than the value of option --min\_lod is ignored and not

stored in memory. `hkxhk` to `hkxhk` `rf`/LOD estimation sometimes fails to provide complete phasing information as `cxr` and `rxr` phases cannot be distinguished, but given sufficient marker density, phasing can likely be worked out indirectly via other markers. An option, `--ignore_cxr=1`, is provided which causes `cxr` and `rxr` `hkxhk` to `hkxhk` linkage to be ignored. In a dense map nearby markers will often have identical genotype calls, or differ only in the pattern of missing calls. An option, `--redundancy_lod`, specifies a LOD threshold above which two markers are checked to see if one can be deleted without losing any information, which can speed up `rf`/LOD calculations for some datasets. A list of redundant markers will be output to the file named by option `--redun` if set, this file will have two columns contains the names of redundant markers followed by the names of the remaining non-redundant markers representing them. To deal with `lmxll` markers incorrectly classified as `nnxnp` and vice versa an option, `--matpat_lod`, is provided. If enabled, `rf` and LOD values are calculated between the maternal information of `lmxll` markers and the paternal information of `nnxnp` markers, and linkage is retained if the LOD meets the specified threshold value (which should be greater than `--min_lod`), and will be used in steps (2) and (3) below.

## (2) Form Linkage Groups

This step forms linkage groups consisting of all markers connected (directly or indirectly) by edges from step (1) (i.e. via LOD values above `--min_lod` and, if linkage between `lmxll` and `nnxnp` markers was enabled, above `--matpat_lod` between these markers). If the `--ignore_cxr` option is disabled then `cxr` and `rxr` linkage between pairs of `hkxhk` markers contributes to linkage group formation even though it will not directly provide phasing information. Markers are split into their respective linkage groups which are processed separately for the remaining steps.

### 1107 (3) Fix Marker-Typing Errors

1108 The --matpat\_lod option when set greater than zero activates an error correction method to  
1109 resolve lmxll markers incorrectly typed as nnxnp and vice versa, the --matpat\_weights option  
1110 allows markers to receive a different reliability weighting based on the first letter of the  
1111 marker name. Using the marker linkages identified during step (1), but excluding all linkage  
1112 involving hkxhk markers, markers are formed into subgroups. For a well-covered linkage  
1113 group with no marker typing errors one subgroup of maternal markers and one of paternal  
1114 would be expected. The correct marker type of each subgroup is assumed to be that of the  
1115 most common type it contains, after taking account of the weighting system. The  
1116 option --matpat\_weights must be set to a string starting with two digits defining the default  
1117 weight followed by any number of weight categories consisting of a single character  
1118 matching the first character of the marker's name followed by two digits. For example to set  
1119 the default weight to 10, and a category for markers starting with X with a weight of 20 and  
1120 one for markers starting with Y with a weight of 95 the option would  
1121 be --matpat\_weights=10X20Y95. Weights are therefore limited to integers in the range 0 to  
1122 99 inclusive. In the event of a tied score markers are arbitrarily set to maternal type. If  
1123 option --matpat\_weights is omitted all markers have a weight of 1.

1124

### 1125 (4) Phasing

1126 lmxll and nnxnp markers require one phase each whereas hkxhk markers require a maternal  
1127 and paternal phase. The phases calculated here are purely relative, so that the same marker  
1128 phase value could be assigned as 0 or 1, but should always be assigned the same phase  
1129 relative to the other markers in its linkage group. Phasing can be estimated between any pair  
1130 of markers that show linkage (except for cxr and rxc linkage between hkxhk markers) based  
1131 on the rf value being greater than or less than 0.5, but is more reliable between markers with

1132 a high LOD value. Maternal and paternal phasing is performed separately, so that the same  
1133 algorithm is run twice for each linkage group, hxxhk markers taking part in both runs.

1134

1135 For each of the two phasing passes (maternal and paternal), markers are joined together  
1136 into a minimum spanning tree (MST) using the relevant linkages identified in step (1) in  
1137 descending order of LOD, such that the LOD of the weakest edge in the tree is minimised. If  
1138 cxr and rxc linkage was allowed (--ignore\_cxr=0) these linkages are given lower priority than  
1139 any other linkage, because they provide only partial phasing information. A warning is written  
1140 to the logfile (if option --log is being used) if the MST required the using any cxr/rxc linkage.  
1141 Phasing is aborted with a warning if a single tree was not produced after all edges were  
1142 used.

1143

1144 An arbitrary starting point marker is assigned to phase 0 and phasing is propagated  
1145 throughout the tree using a depth-first search, propagating the same phase value across  
1146 linkages where rf is less than 0.5 and switching to the other phase value where rf is greater  
1147 than 0.5. If cxr/rxc edges are encountered the phase information is incomplete and is simply  
1148 guessed as always being cxr (i.e. coupling in the maternal and repulsion in the paternal),  
1149 which will be incorrect on average 50% of the time. If cxr/rxc linkage is required to correctly  
1150 form linkage groups at a stringent LOD threshold but causes incorrect phasing,  
1151 crosslink\_group can be run a second time on each linkage group separately to rephase them  
1152 excluding cxr/rxc linkage with a lower LOD threshold (helper script cl\_phase.sh assists this  
1153 process).

1154

1155 (5) Impute Missing Values

1156 Missing genotype calls are imputed using the k nearest neighbour algorithm, whereby  
1157 missing calls are set to the mean value of the k most similar markers which do not lack the  
1158 call. Set the value of k using option --knn. By default --knn=0 disabling missing data  
1159 imputation. Small odd values, such as 1 or 3 are likely to give better results, but performance  
1160 will obviously depend on marker density, genotyping errors and the proportion of missing  
1161 data. Only markers connected by edges from step (1) are considered as potential  
1162 neighbours. hk genotype calls are treated as missing but are not imputed here. Similarity is  
1163 defined using a lower bound of the rf rather than the maximum likelihood rf to guard against  
1164 overestimation between marker pairs with many missing values. Specifically the rf is  
1165 estimated assuming missing values differ with a probability of 0.5 (mimicking unlinked  
1166 markers). If no neighbours are found or there is a tie the value will be imputed to a random  
1167 value and a warning written to the log file. With polyploid genomes often the loss of whole  
1168 chromosomes can be tolerated, meaning some progeny could be aneuploid. If so all the  
1169 marker calls for one chromosome could be missing, and would be filled with random values,  
1170 which should be obvious from the log file or from examining the linkage group using  
1171 crosslink\_graphical. In such a case the linkage group should have the effected progeny  
1172 removed and then be reprocessed.

1173

## 1174 (6) Approximate Marker Ordering

1175 The markers are approximately ordered using a modified version of the MST method  
1176 employed by MSTMap. This involves building a map-distance minimising spanning-tree,  
1177 extracting a longest path through the tree as the map backbone, then collapsing any  
1178 remaining side branches onto the backbone at their point of attachment. This overall  
1179 procedure is performed separately for maternal and paternal maps, which will then be  
1180 merged in the next step. The MST is built using Kruskal's algorithm using the linkages from  
1181 step (1) sorted into ascending order by inter-marker map distance (option --map\_func

selects either the Haldane or Kosambi function). As hk genotypes of hkxhk markers must be treated as missing at this stage in the mapping process, a simple implementation of the MSTMap's approach would cause the backbone to form along a path where missing hk values are masking the largest possible number of recombination events, and therefore likely distort the true ordering. To counter this Kruskal's algorithm is run with the edge list sorted so that linkage involving hkxhk markers appear below those involving only lmxll and nnxnp markers, as the latter edges cannot mask recombination events to the same extent. Since the linkage groups were formed using both maternal and paternal markers there is no guarantee that the separate maternal and paternal MST will span the full linkage group. If this is the case the approximate ordering is abandoned (crosslink\_map can still be used to produce a final ordering). Once the MST is built a longest path is extracted using two depth-first searches. The first search starts at an arbitrary marker and finds the most distant marker from the starting point. The second search starts at this marker and finds the marker most distant from it in turn. The map backbone path is extracted by backtracking from this second marker to the first, and is traversed to assigned map positions to backbone markers. Finally, a depth-first search from each backbone marker of any side branch markers assigns them the same position as the associated backbone marker.

## (7) Combined Map Positions

A combined map is produced using the maternal and paternal maps created in the previous step, following the method used by JoinMap®. In order to produce a useful combined map there must be at least two hkxhk markers (preferably widely spaced) otherwise it is not possible to work out the relative orientations of the two maps. A warning is written to the log file if any linkage group has less than 2 hkxhk markers. Otherwise, the covariance between the maternal and paternal positions of all hkxhk markers is found, and the paternal map positions are inverted if the covariance is negative. hkxhk marker combined positions are set

1208 to the average of their maternal and paternal map positions. lmxll and nnxnp marker  
1209 combined positions are interpolated between the nearest two flanking hkxhk markers, or  
1210 extrapolated from the nearest hkxhk if they are not flanked by two. Finally, the markers are  
1211 sorted into the order defined by their combined map position.

1212

1213 This approximate ordering defines the marker order in the output genotype and map files.  
1214 For each linkage group the genotypes are written to a file named as the option --outbase  
1215 followed by the linkage group number with a .loc extension. The corresponding maps are  
1216 output to files names as the --mapbase option with a .map extension. These files are not  
1217 written unless the --outbase and --mapbase options are explicitly set.

1218

## 1219 **crosslink\_map**

1220

```
1221 --inp=STRING input genotype file (required)
1222 --out=STRING output genotype file (optional)
1223 --log=STRING output log file (optional)
1224 --map=STRING output map file (optional)
1225 --seed=UNSIGNED random number generator seed, 0=use system time
1226 (default: 1)
1227 --map_func=UNSIGNED mapping func, 1=Haldane,2=Kosambi (default: 1)
1228 --randomise_order=UNSIGNED start from a random initial marker ordering
1229 (default: 0)
1230 --bitstrings=UNSIGNED use bitstring data representation internally
1231 (default: 1)
1232 --ga_gibbs_cycles=UNSIGNED number of GA-Gibbs cycles (default: 5)
1233 --ga_iters=UNSIGNED number of GA iterations per GA-Gibbs cycle (default:
1234 100000)
```



```

1235 --ga_use_mst=UNSIGNED how many GA-Gibbs cycles to perform initial MST
1236 ordering before the GA (0=none,N=up to and including the Nth cycle)
1237 (default: 999)
1238 --ga_minlod=FLOAT min LOD for MST construction and global order
1239 optimisation scoring (default: 3.0)
1240 --ga_mst_nonhk=UNSIGNED prioritise non-hk linkage when building the MST
1241 (default: 0)
1242 --ga_optimise_meth=UNSIGNED 0=optimse map total recombination events,
1243 1=optimise total map distance, 2=optimise a global measure of map
1244 quality (sets --ga_skip_order1=1 --randomise_order=0) (default: 0)
1245 --ga_prob_hop=FLOAT probability a mutation moves a single marker
1246 (default: 0.333)
1247 --ga_max_hop=FLOAT max distance a single marker can move as proportion
1248 of whole linkage group (default: 0.1)
1249 --ga_prob_move=FLOAT probability a mutation moves a block of multiple
1250 markers (default: 0.333)
1251 --ga_max_mvseg=FLOAT max number of markers in the block as proportion of
1252 whole linkage group (default: 0.1)
1253 --ga_max_mvdist=FLOAT max distance the block of markers can move as
1254 proportion of whole linkage group (default: 0.1)
1255 --ga_prob_inv=FLOAT probability the block of markers also inverts as
1256 well as moves (default: 0.5)
1257 --ga_max_seg=FLOAT for in-place inversion mutations, max number of
1258 markers to be inverted as proportion of whole linkage group (default:
1259 0.1)
1260 --ga_cache=UNSIGNED 1=use cache of rf values in GA (default: 1)
1261 --ga_em_tol=FLOAT for 2 point rf calculations, convergence tolerance for
1262 EM algorithm (default: 1e-5)
1263 --ga_em_maxit=UNSIGNED for 2 point rf calculations, max EM iterations
1264 (default: 100)
1265 --ga_skip_order1=UNSIGNED 1=skip first GA ordering, go straight to Gibbs
1266 using the marker order from the input file (default: 0)
1267 --ga_report=UNSIGNED GA log reporting period, 0=disabled (default: 0)

```

```

1268 --gibbs_samples=UNSIGNED number of Gibbs samples to collect per GA-Gibbs
1269      cycle (default: 300)
1270 --gibbs_burnin=UNSIGNED Gibbs burn in cycles (default: 10)
1271 --gibbs_period=UNSIGNED Gibbs cycles per sample (default: 1)
1272 --gibbs_prob_sequential=FLOAT probability Gibbs cycle uses sequential
1273      mode (default: 0.0)
1274 --gibbs_prob_unidir=FLOAT probability Gibbs cycle uses unidirectional
1275      mode (default: 1.0)
1276 --gibbs_min_prob_1=FLOAT minimum permitted probability of a state
1277      transition at the start of burn in period (default: 0.1)
1278 --gibbs_min_prob_2=FLOAT minimum permitted probability of a state
1279      transition by the end of burn in period (default: 0.0)
1280 --gibbs_twopt_1=FLOAT weighting given to two point rf at start of burn
1281      in period (default: 1.0)
1282 --gibbs_twopt_2=FLOAT weighting given to two point rf by the end of burn
1283      in period (default: 1.0)
1284 --gibbs_report=UNSIGNED Gibbs log reporting period, 0=disabled (default:
1285      0)
1286 --homeo_minlod=FLOAT detect cross homeolog markers, minlod (default:
1287      1.0)
1288 --homeo_maxlod=FLOAT detect cross homeolog markers, maxlod (default:
1289      16.0)
1290 --homeo_mincount=UNSIGNED report as possible cross homeolog if
1291      implicated more than this many time, 0 to disable (default: 0)

```

1292

1293

1294 This program, also implemented in C, performs the second part of the mapping pipeline,  
1295 which produces the final map ordering, imputes the missing information for hk genotypes  
1296 and calculates final map positions. It also has a method to detect cross-linkage-group hxxhk  
1297 markers which have strong linkage to more than one true linkage group. It loads only one  
1298 linkage group per invocation, from the genotype file specified by required option --inp. The  
1299 final hk-imputed genotypes in their final ordering are output to a file specified by option --out,

1300 with final map positions going into the file given by option --map and with optional logging  
1301 information going to a file specified by --log. The --seed , --randomise\_order , --bitstrings  
1302 and --map\_func options have the same function as for crosslink\_group (see above). On  
1303 multicore systems multiple linkage groups can to processed in parallel (helper script  
1304 cl\_order\_hkimpute.sh can be used to facilitate this).

1305

1306 The two main steps are (1) ordering the markers using a genetic algorithm (GA) and (2)  
1307 imputing the missing hk information using a Gibbs sampler. These steps are alternated the  
1308 specified number of times. Option --ga\_gibbs\_cycles controls the number of cycles of  
1309 GA+Gibbs that are run, --ga\_iters controls how many iterations of the GA are run each cycle,  
1310 and --gibbs\_burnin, --gibbs\_samples and --gibbs\_period control the iterations of the Gibbs  
1311 sampler (see below). The --ga\_skip\_order1 option allows the first round of GA ordering to  
1312 be skipped, so that the approximate ordering output from crosslink\_group will be used  
1313 directly by step (2) before any further ordering changes are made. If not skipped, the first  
1314 round of ordering will use two-point rf values (treating hk genotypes as missing). All  
1315 subsequent ordering rounds use multipoint rf values from the imputed hk information. Final  
1316 map positions are calculated in the same way as crosslink\_group following the method of  
1317 JoinMap, by averaging the maternal and paternal positions of the shared hkxhk markers,  
1318 and interpolating/extrapolating the positions of lmxll and nnxnp markers based on the  
1319 nearest hkxhk marker(s). Detection of cross-linkage-group hkxhk markers takes place before  
1320 the final cycle of marker ordering.

1321

## 1322 (1) Improve Marker Ordering With A Genetic Algorithm

1323 The genetic algorithm works by scoring the optimality of the current marker ordering, making  
1324 a random change to the ordering, then reassessing the score. If the score improves or stays

1325 the same the new ordering is accepted, otherwise the change is rejected. The idea behind  
1326 accepting neutral changes is that they may allow the ordering to escape from local optima.

1327

1328 For each of the cycles specified by option `--ga_gibbs_cycles` (minus the first cycle  
1329 if `--ga_skip_order1` is enabled) the GA runs for `--ga_iters` iterations, trying one mutation per  
1330 iteration. There are three available scoring methods, all of which take account of both  
1331 maternal and paternal information simultaneously: `--ga_optimise_meth=0` minimises the  
1332 sum of recombination events in maternal and paternal maps, `--ga_optimise_meth=1`  
1333 minimises the combined map lengths of the maternal and paternal maps, quantised to  
1334 100ths of a centimorgan, and `--ga_optimise_meth=1` maximises a metric which rewards  
1335 small recombination fractions between markers which are also nearby in the current  
1336 ordering. The third method provides more of a "global" picture of the ordering optimality by  
1337 comparing all against all markers (with a LOD threshold) instead of just adjacent ones, but is  
1338 more computationally intensive for this reason. For options 0 and 1, each time a pair of  
1339 markers are compared the calculated recombination count or map distance is cached  
1340 (disabled using `--ga_cache=0`) so that next time those two markers are compared the value  
1341 will not need to be recalculated. For the global scoring method, at the start of each GA round  
1342 a list of rf and LOD values are calculated by comparing all markers against all others, those  
1343 with LOD values lower than option `--ga_minlod` are dropped, the otherwise the rf values are  
1344 converted into map distances. To find the overall score for a map ordering each map  
1345 distance stored in the list is given a score which disproportionately rewards short map  
1346 distances which are also nearby in the ordering. Longer map distances and/or larger  
1347 separations in the ordering reduce the score. The best score is obtained by having as many  
1348 short map distances correspond to nearby ordering positions as possible. This will penalise  
1349 any strongly linked markers which are widely separated by the ordering even where all  
1350 immediately adjacent pairs have low recombination counts. The overall score for the  
1351 ordering is the sum of all the individual scores.

1352 Option `--ga_use_mst`, when set to greater than zero, causes the markers to be pre-ordered  
1353 using the same MST method as use by `crosslink_group` at the start of GA cycles. This  
1354 process is not supported for the first GA cycle (as it is assumed that the ordering given by  
1355 `crosslink_group` would be available already). It is possible to have all subsequent cycles  
1356 approximately ordered, or only up to a given cycle number. For example, numbering the GA  
1357 cycles starting from 0, setting `--ga_use_mst=3` would cause cycles numbers 1, 2 and 3 to  
1358 use approximate ordering, but subsequent cycles to not use it. The specified number of GA  
1359 iterations are always performed regardless of whether pre-ordering is used.  
1360 Option `--ga_mst_nonhk=1` causes the pre-ordering to prioritise linkages not involving `hkxhk`  
1361 markers, whereas `--ga_mst_nonhk=0` (the default) pre-orders making full use of `hkxhk`  
1362 markers at the same priority as the other marker types.

1363

1364 If the first round of ordering is not skipped then `rf` values are calculated using the same two-  
1365 point methods as `crosslink_group`, with the EM algorithm tolerance and maximum iterations  
1366 defined by parameters `--ga_em_tol` and `--ga_em_maxit` respectively.

1367

1368 Three types of mutation operator are provided: single marker hop, segment move and  
1369 segment inversion, chosen with probabilities controlled by options `--ga_prob_hop`  
1370 and `--ga_prob_move` (with the probability of inversion being the remainder after subtraction  
1371 from 1.0). A single marker hop moves a single marker a random distance up to a proportion  
1372 of the total marker count controlled by option `--ga_max_hop`. A segment move moves a  
1373 randomly chosen contiguous block of markers with a size from 1 up to a proportion of the  
1374 total marker count controlled by option `--ga_max_mvseg`, a random distance up to a  
1375 proportionate maximum controlled by option `--ga_max_mvdist`, and inverts the block with a  
1376 probability `--ga_prob_inv`. A segment inversion inverts a randomly chosen block of markers  
1377 in-place with a size of 2 up to a proportionate maximum controlled by option `--ga_max_seg`.

1378 If logging is enabled by option --log, the option --ga\_report controls how often the current  
1379 ordering score is written to the log file, --ga\_report=0 disables reporting the ordering score,  
1380 whereas values greater than 0 report the score every nth iteration.

1381 At the start of the final cycle of marker ordering if option --homeo\_mincount is greater than  
1382 zero then detection of cross-linkage-group hkxhk markers takes place.

1383

1384 In our allo-polyploid strawberry data we have encountered a few hkxhk markers whose  
1385 maternal information comes from one linkage group while the paternal information comes  
1386 from one of the homeologous linkage groups. These cases are detected by comparing pairs  
1387 of hkxhk markers, where we expect the markers to be linked according to one parent but  
1388 unlinked according to the other. For a pair of hkxhk markers if one parent indicates a LOD  
1389 below --homeo\_minlod and the other a LOD above --homeo\_maxlod then both markers are  
1390 given one point. Markers accumulate points for each qualifying hkxhk marker pair they are  
1391 involved in. Those that receive a total number of points greater than or equal  
1392 to --homeo\_mincount are flagged as being cross linkage group markers, and reported in the  
1393 log file (provided logging is enabled).

1394

## 1395 (2) Impute hk Information By Gibbs Sampler

1396 For each hkxhk marker genotype called as hk it is unknown whether the h allele came from  
1397 the maternal and the k allele from the paternal parent or vice versa. Without this information  
1398 the rf values cannot be known as accurately, as all hk genotypes must be treated as  
1399 missing. Here using the convention of listing the maternal allele first, the two possible  
1400 imputed forms of the genotype call will be written as HK and KH, using uppercase to indicate  
1401 they have been imputed (note lowercase is still used in the actual genotype files even for  
1402 imputed markers). The marker is present in both the maternal and paternal maps, therefore  
1403 both need to be considered simultaneously. The most likely choice between HK or KH

1404 depends on the current marker ordering, whether either imply a recombination before and/or  
1405 after the marker and the current recombination fractions between the previous and next  
1406 markers in the maternal and paternal maps. The problem then is to find the most probable  
1407 set of choices for all the hk genotypes considered jointly. A Gibbs sampler can address this  
1408 problem by initially randomising all hk genotypes, and then resampling each individually in a  
1409 random order many times. The final state of each is then set to the most common during  
1410 resampling, discarding an initial burn-in period. Resampling an hk genotype entails  
1411 assuming its true state is unknown but that all other states are known and correct,  
1412 calculating the probabilities of HK and KH given the neighbouring marker states, and  
1413 assigning the state to either HK or KH with the respective probabilities.

1414 Three implementations are provided all based on the basic Gibbs sampler algorithm: (A) a  
1415 Gibbs sampler which resamples hk genotypes in a completely random order, (B) a Gibbs  
1416 sampler which resamples states sequentially along the current map order one progeny at a  
1417 time, treating the progeny in a random order, and in either forward or reverse map order at  
1418 random (C), a simplified heuristic which resamples sequentially per progeny in the same  
1419 manner as method B but only takes account of the preceding marker states as it progresses.

1420

1421 Each Gibbs cycle begins by randomising all hk genotype states. Then, during a burn-in  
1422 period, the states are resampled a given number of times (controlled by  
1423 option --gibbs\_burnin) using one or more of methods A-C. A counter associated with each hk  
1424 genotype is then increment if the state is currently HK or decremented if it is KH. Then the  
1425 states are resampled again a given number of times (controlled by option --gibbs\_period)  
1426 and the counters updated again. This is repeated until the required number of samples  
1427 (controlled by option --gibbs\_samples) have been obtained. For example if --gibbs\_burnin is  
1428 20, --gibbs\_period is 1 and --gibbs\_samples is 300 then every state will be resampled 20  
1429 times during the burn-in before the first counter update, then exactly once between all

1430 subsequent counter updates until each counter has been updated a total of 300 times.  
1431 Therefore each Gibbs iteration consists of resampling each `hk` state exactly once.

1432

1433 Each iteration the method of resampling is chosen from the three available methods  
1434 (controlled by options `--gibbs_sequential`, the probability of using method B,  
1435 and `--gibbs_unidir`, the probability of method C, the probability of A being the remainder after  
1436 subtraction from 1.0), such that any one method can be used exclusively or a random  
1437 mixture of some or all.

1438 The initial burn-in period is used to allow the states to converge from their initial random  
1439 values towards a more optimal part of the state space. Some additional options are provided  
1440 to fine tune the resampling process during burn-in. Firstly, to prevent the state getting stuck  
1441 prematurely in a suboptimal part of state space, options `--gibbs_min_prob_1`  
1442 and `--gibbs_min_prob_2` control the minimum probability that a resampled `hk` state will  
1443 change to the other value. For the very first iteration of burn-in the minimum probability  
1444 is `--gibbs_min_prob_1`, for the last burn-in iteration it is `--gibbs_min_prob_2`, and for  
1445 intermediate iterations the probability changes linearly from the former to the latter. After  
1446 burn-in, during normal iterations, the minimum probability remains at `--gibbs_min_prob_2`. At  
1447 the start of burn-in, multipoint `rf` values are available immediately, but are initially determined  
1448 from the randomly chosen `hk` states, consequently the two-point `rf` values may be more  
1449 accurate until some point through the burn-in period. For this reason, when resampling `hk`  
1450 states a weighted average of the multipoint and two-point `rf` values is used to calculate the  
1451 state probabilities. Two control parameters are provided, `--gibbs_twopt_1`  
1452 and `--gibbs_twopt_2`, the weighting given to two-point values being changed linearly from the  
1453 first to the second over the course of the burn-in period and remaining at the second value  
1454 during normal iterations.



1455 If logging is enabled by option --log, the option --gibbs\_report controls how often the current  
1456 total number of recombination events is written to the log file, --gibbs\_report=0 disables  
1457 reporting, whereas values greater than 0 report the count every nth iteration.

1458

### 1459 (3) Calculate Final Map Positions

1460 This step follows the same procedure as the final step of the crosslink\_group program.

## 1461 **crosslink\_pos**

```
1462     --inp=STRING name of input genotype file (required)
1463     --out=STRING name of output map file (optional)
1464     --seed=UNSIGNED random number generator seed, 0=use system time (default: 1)
1465     --map_func=UNSIGNED mapping func, 1=Haldane,2=Kosambi (default: 1)
1466     --bitstrings=UNSIGNED use bitstring data representation internally (default: 1)
```

1467

1468 This program loads genotype data from the file specified by option --inp and, treating  
1469 the data as fully hk-imputed genotypes, calculates the final map positions. This is a  
1470 convenience function in that it only implements a subset of the functionality of  
1471 crosslink\_map, and the options it shares have the same meaning.

1472

## 1473 **crosslink\_viewer**

```
1474     --inp=STRING input genotype file(s) (required)
1475     --window_size=UNSIGNED window size (pixels) (default: 1000)
1476     --datatype=STRING state of the genotype data: imputed, phased, unphased (default:
1477         imputed)
1478     --bitstrings=UNSIGNED 1=use bitstring representation of the data internally
1479         (default: 1)
1480     --hardware=UNSIGNED 1=use hardware graphical acceleration when available
1481         (default: 0)
1482     --skip=UNSIGNED how many markers to skip at the start of the genotype file
1483         (default: 0)
```

```
1484      --total=UNSIGNED how many markers to load in total, 0=load all (default: 0)
1485      --marker=STRING named marker to centre initial view on (optional)
1486
```

1487 This program allows visualisation of the pattern of LOD, phasing and map  
1488 distances between markers on one or more linkage groups. Input is read from  
1489 a list of one or more genotype files specified with option --inp. To view more  
1490 than one linkage group at once surround the space-separated file list with  
1491 double or single quotes. The markers are treated as unphased, phased or  
1492 fully hk-imputed according to whether option --datatype is set to unphased,  
1493 phased or imputed respectively. --window\_size specified the width and height  
1494 of the plot in pixels (excluding an window decoration). --bitstrings has the  
1495 same meaning as for crosslink\_group. --hardware=1 activates hardware  
1496 accelerated rendering is available. --skip=N allows the first N markers to be  
1497 dropped and --total=M means to retain only the next M markers in total  
1498 (useful for viewing part of a large dataset). --marker allows the initial view to  
1499 be centred on a particular named marker. See the Quickstart Guide section  
1500 for a explanation of the coloured plot produced by this program.

1501 The control keys to control the plot are:

```
1502 CURSOR KEYS  = pan
1503 +/-          = zoom
1504 ENTER        = recentre
1505 m,p,c        = maternal,paternal,combined mode
1506 q,escape,0-9 = quit with various bash exit codes ($?)
```

1507

1508 **crosslink\_graphical**

1509

```

1510 --inp=STRING input genotype file(s) (required)
1511 --sizex=UNSIGNED window width (pixels) (default: 1000)
1512 --sizey=UNSIGNED window height (pixels) (default: 1000)
1513 --datatype=STRING state of the genotype data: imputed, phased, unphased
1514         (default: imputed)
1515 --bitstrings=UNSIGNED 1=use bitstring representation of the data
1516         internally (default: 1)
1517 --hardware=UNSIGNED 1=use hardware graphical acceleration when available
1518         (default: 0)
1519 --skip=UNSIGNED how many markers to skip at the start of the genotype
1520         file (default: 0)
1521 --total=UNSIGNED how many markers to load in total, 0=load all (default:
1522         0)
1523

```

1524 This program displays the pattern of genotypes along one or more linkage groups. Options  
1525 are the same as for `crosslink_viewer` except: option `--sizex` controls only the width  
1526 and `--sizey` only the height of the plot in pixels. The control keys are:

```

1527 CURSOR KEYS          = pan
1528 SHIFT+CURSOR KEYS    = reshape
1529 +/-                  = zoom
1530 ENTER                = recentre
1531 m,p,c,z              = maternal,paternal,combined,phased/unphased mode
1532 q,escape,0-9         = quit with various bash exit codes ($?)
1533

```

1534 See the Quickstart Guide section for an explanation of the coloured plot produced.

## 1535 **create\_map**

```

1536 --output-file=STRING output map specification file (required)
1537 --random-seed=UNSIGNED random number generator seed (0=use system time)
1538         (default: 0)

```

```

1539 --numb-lgs=UNSIGNED divide map into this number of equally sized linkage
1540 groups (default: 10)
1541 --map-size=FLOAT total map size in centimorgans (default: 100.0)
1542 --marker-density=FLOAT average markers per centimorgan (default: 1.0)
1543 --prob-both=FLOAT probability marker is heterozygous in both parents
1544 (default: 0.3333)
1545 --prob-maternal=FLOAT probability marker is heterozygous only in
1546 maternal parent (default: 0.333)
1547 This program outputs a map specification file to the file specified by option --output-
1548 file. --random-seed has the same meaning as the --seed option of crosslink_group, except
1549 the default value is 0 (meaning to use microsecond resolution system time). --numb-lgs
1550 determines the number of linkage groups, all of equal size. --map-size determines the total
1551 map length in centimorgans. --marker-density is the average number of markers per
1552 centimorgan. --prob-maternal is probability that a marker is a maternal (lmxll) marker, --prob-
1553 both is the probability of a shared (hkxhk) marker, the probability of a paternal (nnxnp)
1554 marker is the remainder after subtraction from 1.0. The map specification file is used by
1555 sample_map to create simulated outcross genotype data.

```

## 1556 **sample\_map**

```

1557 --input-file=STRING input map specification file (required)
1558 --output-file=STRING output genotype file including any errors
1559 (required)
1560 --orig-dir=STRING output grouped genotype files without any errors
1561 (optional)
1562 --random-seed=INTEGER random number generator seed, 0=use system time
1563 (default: 0)
1564 --samples=INTEGER number of offspring to simulate (default: 200)
1565 --prob-missing=FLOAT probability a genotype call is missing (default:
1566 0.0)
1567 --prob-error=FLOAT probability a genotype call is incorrect (default:
1568 0.0)
1569 --map-function=INTEGER 1=Haldane, 2=Kosambi (default: 1)

```

1570 This program takes a map specification file (--input-file) and produces an output file  
1571 (--output-file) containing simulated outcross genotype data containing optional errors, plus  
1572 an option set of files in a subdirectory (--orig-dir) containing the same markers correctly  
1573 grouped and ordered without an errors. --random-seed has the same meaning as for  
1574 create\_map. --samples determines the number of progeny to simulated from the  
1575 cross. --map-function determines which mapping function to assume (Haldane or Kosambi)  
1576 when calculating the probability of recombination from map distances. --prob-missing  
1577 determines the probability that a genotype call will be set to missing and --prob-error that a  
1578 call will be set to an incorrect value. This program does not simulate any allo-polyploidy  
1579 related errors (see create\_type\_errors.py in the helper script section below for this  
1580 functionality).

## 1581 **Helper Scripts**

1582 It was originally intended that the two programs crosslink\_group and crosslink\_map would  
1583 each have to be run only once on the data to produce a final map. Later it became clear that  
1584 more flexibility was advantageous. Ideally all the functionality would be broken out into  
1585 separate functions accessible via a Python or R interface and available to run in any order.  
1586 The helper scripts described in this section provide a similar functionality, but work by simply  
1587 calling the two main programs internally and ignoring any of the output they don't need.  
1588 Other scripts also automate common tasks such as renaming linkage groups to match an  
1589 existing reference map. See the example pipeline dealing with real data in the QuickStart  
1590 section for examples of how to use some of the helper scripts and for a description of the  
1591 configuration files used by some of them. Example configuration files are also provided in  
1592 the directory \${CROSSLINK\_PATH}/sample\_data/rgxha\_conf. In general if a directory name  
1593 is required as an input then all files with a .loc extension in the directory will be treated as  
1594 linkage groups. If a directory name is required for output then generally .loc genotype files or  
1595 .map files will be output to that directory. The last option for many of the scripts is

1596 configuration file containing the parameters to be passed to crosslink\_group and/or  
1597 crosslink\_map.

1598

1599 **cl\_adjustlgs.sh <input-dir> <vsref-file> <output-dir>**

1600 Renames and reorients linkage groups to match a reference map. All .loc files in the input  
1601 directory will be processed. The vsref file is a file created by cl\_match2ref.sh and specifies  
1602 the new name to be given to each linkage groups and whether it should be reoriented. The  
1603 new .loc files will be written to the output directory, which is created if required.

1604 **cl\_break.sh <input-file> <marker-name> <output-dir> <conf-file>**

1605 Break the linkage group in the input genotype file into two parts after the named marker, and  
1606 output the new linkage groups into the output directory.

1607 **cl\_detect\_crosslg.sh <input-dir> <output-file> <conf-file>**

1608 Detect cross-linkage-group hxxhk markers in all .loc files in the input directory, output a list of  
1609 their names to the output file. Reads configuration options from conf-file.

1610 **cl\_extract.sh <input-file> <redun-file> <output-file>**

1611 Extracts from the input genotype file all markers not listed as redundant in the redun-file  
1612 (created by crosslink\_group or cl\_findredun.sh) and output them to a new output genotype  
1613 file.

1614 **cl\_findredun.sh <input-file> <output-file> <conf-file>**

1615 Run crosslink\_group on the input genotype file with the parameters defined in the conf-file,  
1616 output a list of redundant markers to the output file.

1617 **cl\_fixtypes.sh <input-file> <output-file> <conf-file>**

1618 Fix marker typing errors in input genotype file using parameters from conf-file, output  
1619 corrected genotypes to output file. Can safely overwrite the original if output filename set  
1620 same as input.

1621 **cl\_group.sh <input-file> <output-dir> <min-lod>**

1622 Form markers from input genotype file into linkage groups based on the specified LOD  
1623 threshold, output to new linkage group files in the output directory, which is created if  
1624 required. Any existing .loc or .map files in the output directory are **deleted** first to avoid  
1625 retaining any stale files from a previous grouping attempt.

1626 **cl\_hkimpute.sh <input-file> <output-file> <conf-file>**

1627 Impute hk genotypes in the input genotype file without changing marker order and output to  
1628 a new genotype file.

1629 **cl\_knnimpute.sh <input-file> <output-file> <conf-file>**

1630 Impute missing genotype calls using k nearest neighbour method for markers in input  
1631 genotype file, output to a new genotype file. Note: marker order may not be preserved.

1632 **cl\_loc2joinmap.sh <input-dir> <output-file>**

1633 Convert all the .loc genotype files from the input directory into a JoinMap compatible .loc file.

1634 **cl\_map2joinmap.sh <input-dir> <output-file>**

1635 Convert all the .map files from the input directory into a JoinMap compatible .map file.

1636 **cl\_mappos.sh <input-dir> <output-dir>**

1637 For each .loc file in the input directory create a .map file containing the map positions in the  
1638 output directory (which can be the same as the input directory). Does not alter marker  
1639 ordering.

1640 **cl\_match2ref.sh <input-dir> <refmap-file> <ps2snp-file>**

1641 This script is specific to the type of markers used in the Redgauntlet x Hapil dataset. Match  
1642 all the per-linkage group .map files in the input directory to linkage groups in the reference  
1643 map described in the refmap file. ps2snp-file is a file matching probesetids to snpids for the  
1644 microarray used to genotype the samples.

1645 **cl\_merge.sh <input-dir> <merge-file> <output-dir> <conf-file>**

1646 Each line of the merge file contains a space separated list of linkage group names (i.e.  
1647 filenames from the input directory with the .loc extension removed) that will be merged and  
1648 placed in the output directory. The extension of the original .loc files are changed to .old and  
1649 the name of the new file is formed by joining the names of the original files with underscore  
1650 characters (\_). The new linkage group is then rephased, ordered and hk-imputed.

1651 **cl\_modifymarkers.sh <input-file> <output-file> <mat2pat-file> <pat2mat-file>**

1652 For markers named in file mat2pat-file manually change the marker type from lmxll to nnxnp  
1653 and for those named in file pat2mat-file change type from nnxnp to lmxll. Markers are read  
1654 from the input genotype file and written to the output file. Markers names must appear one  
1655 per line in the mat2pat and pat2mat files.

1656 **cl\_order\_hkimpote.sh <input-dir> <output-dir> <conf-file>**

1657 Produce final marker ordering and impute missing hk information for each .loc genotype file  
1658 in the input directory, writing output to the output directory, which can be the same as the  
1659 input directory. By setting parameter CL\_PARALLEL\_JOBS in the conf-file to greater than 1,  
1660 multiple linkage groups can be processed in parallel on multicore systems.

1661 **cl\_phase.sh <input-dir> <output-dir>**

1662 Force phasing to complete for each .loc genotype file in the input directory, write output to  
1663 output directory which can be the same as the input directory.

1664 **cl\_refine\_order.sh <input-dir> <output-dir> <trials> <parallel-jobs> <conf-file> [<conf-file>...]**

1665 For each .loc genotype file in the input directory, run map ordering and hk-imputation 'trials'  
1666 times for each of the list of one or more configuration files and retain only the best map  
1667 obtained, writing it to the output directory, which can be the same as the input directory.  
1668 Process 'parallel-jobs' linkage groups in parallel. If the configuration option  
1669 CL\_IGNORE\_PREVIOUS is set to 1 in the first configuration file then the best *new* ordering  
1670 is retained regardless of whether it is worse than the original, otherwise if



1671 CL\_IGNORE\_PREVIOUS=0, the original ordering is retained unless a better one is found.  
1672 Orderings are scored based on the combined lengths of the maternal and paternal maps.

1673 **cl\_reinsert\_loc.sh <input-dir> <all-markers> <redun-file> <output-dir> <conf-file>**

1674 For each .loc genotype file in the input directory, reinsert redundant markers from all-  
1675 markers file using redundancy information from redun-file, and write output to output  
1676 directory, which can be the same as the input directory. Reinserted markers are rephased  
1677 and hk-imputed but the ordering of the original markers is not changed.

1678 **cl\_reinsert\_map.sh <input-dir> <redun-file> <output-dir>**

1679 For each .map file in the input directory, reinsert redundant markers into the map at the  
1680 same position as their representative non-redundant marker, without reordering the map,  
1681 write output to output directory, which can be the same as the input directory.

1682 **cl\_removemarkers.sh <input-file> <output-file> <bad-markers-file>**

1683 Filter out markers names one-per-line in the bad markers file from the input genotype file  
1684 and write to the output file which can be the same as the input file.

1685 **cl\_subgroup.sh <input-file> <min-lod> <output-dir> <conf-file>**

1686 Regroup markers in the input genotype file at the specified LOD threshold, write new linkage  
1687 group(s) to the output directory. Each new linkage group is rephased, ordered and hk-  
1688 imputed.

## 1689 **Glossary**

1690 Imxll - a marker which is heterozygous in the maternal parent but homozygous in the  
1691 paternal

1692 nnnnp - a marker which is heterozygous in the paternal parent but homozygous in the  
1693 maternal

1694 hxxhk - a marker heterozygous in both parents

1695 rf - recombination fraction, the proportion of progeny showing recombination between two  
1696 markers  
1697 LOD - the (linkage) logarithm of odds, a measure of the statistical significance of the linkage  
1698 between two markers, specifically the base-10 logarithm of the ratio of probabilities that the  
1699 markers are truly linked versus unlinked  
1700 GA - genetic algorithm  
1701 MST - minimum spanning tree  
1702