QTL Cartographer
Version 1.17

Christopher J. Basten Bruce S. Weir Zhao-Bang Zeng

January 28, 2005

QTL Cartographer

QTL Cartographer

A Reference Manual and Tutorial for QTL Mapping

Christopher J. Basten, Bruce S. Weir and Zhao-Bang Zeng
Program in Statistical Genetics
Bioinformatics Research Center
Department of Statistics
North Carolina State University

QTL Cartographer

Program in Statistical Genetics
Bioinformatics Research Center
Department of Statistics
North Carolina State University
Raleigh, NC 27695-7566.

Printed in the United States of America
Typeset in LaTeX2e on a Macintosh G4.

Preface

Notational Conventions

We will attempt to follow certain conventions in this manual.

1. Computer hostnames, C language subroutines and new terms will be italized. Example: The main server for ***QTL Cartographer*** is *statgen.ncsu.edu.*

2. Directories (and folders), filenames and computer commands was well as program names will be in bold type. Example: **Rmap** produces an output file called **qtlcart.rc** in the folder **Macintosh HD:QTLCartMac:test**.

3. The content of files and snippets of code will be in the verbatim environment, which is a fixed width font.

   ```
   set i=1
   while ( i < 10 )
   Rmap -A -V
   @ i++
   end
   ```

   UNIX commands will also be in this typeface. Also, a percent sign (%) should be understood to mean the UNIX prompt when in the context of UNIX commands.

Operating Systems

The programs described here work under various operating systems. They have been tested under three main classes of sytems: We refer to them as Windows, Macintosh and UNIX.

When we refer to "Windows" binaries, we mean programs that are compiled to run under the Microsoft Windows operating system with 32 bit support. These binaries should run under Windows 95, 98, ME, NT, 2000 and XP. They can be run either by double clicking their icons in a filesystem navigation application, or run in a command window by typing their command names. Binaries for 16-bit Windows systems (Windows 3.1 and 3.11) are available in an older version of ***QTL Cartographer***. The last version for 16-bit Windows was 1.12f.

Macintosh binaries come in two flavors: Those that are compiled for the "Carbon" libraries and those that aren't. If you have Macintosh OS X or higher, you should be able to use the carbonized binaries. Version 1.13g is the last version that had binaries for the older 68k Macintoshes (that is, the pre-PowerPC Macintoshes). Beginning with Macintosh OS X you have the option of compiling the programs for use in the UNIX environment.

UNIX command line programs need to be compiled from the source code. There may be some quirks about your local compiler that will require some editing of the **Makefile**: See the **INSTALL** file in the source code distribution for more information.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Overview

QTL Cartographer is a suite of programs for mapping quantitative trait loci (QTLs) onto a genetic linkage map. The general experimental paradigm begins with a pair of inbred parental lines that differ in the trait of interest and in the set of marker genotypes. The programs use linear regression, interval mapping (Lander and Botstein 1989), composite interval mapping (Zeng 1993; Zeng 1994) and multiple interval mapping (Kao and Zeng 1997; Kao, Zeng, and Teasdale 1999; Zeng, Kao, and Basten 1999) methods to dissect the underlying genetics of the quantitative traits. Mapping is done onto a set of linked genetic markers with known recombination frequencies. Genetic linkage maps and data files can be imported from **Mapmaker/EXP** (Lander et al. 1987). The mapping program uses a dynamic algorithm that allows a host of statistical models to be fitted and compared, including various gene actions (additive and dominance), QTL-environment interactions, and close linkage.

This package consists of several programs written in C to perform various tasks, including simulating, reformatting or analyzing data and visualizing the results of the analyses. Presently, the mapping programs can handle data from backcrosses, intercrosses and recombinant inbreds, as well as a few other experimental designs (see Table 1.1).

All input and output files are plain text and can be viewed or imported into many text editors and graphics packages on various computing platforms. The programs were originally written for the UNIX operating system and have since been ported to the Macintosh and Microsoft Windows operating systems. Present development is on a Macintosh using **Metroworks Codewarrior**, which produces binaries for both the Macintosh and Windows operation systems. The UNIX distribution is of the source code and must be compiled at the user's site. This project is ongoing and suggestions are welcome for further improvements and enhancements. The source code and compiled binaries are freely available and may be obtained by anyone over the internet.

1.1.1 Definition of the Problem

Often traits in plants and animals are influenced by many genes rather than a single locus (Falconer and MacKay 1996, for an excellent general review). These traits are termed quanti-

tative traits and the loci that control these traits quantitative trait loci, abbreviated henceforth as QTLs. An important goal in genetics and breeding is to identify and characterize QTLs, especially those that contribute to variation in quantitative traits both within and between populations or species. The recent advances in molecular biology have allowed the construction of genetic linkage maps based on molecular markers. Such genetic linkage maps can span the genome at regular intervals. The experimenter can then look for correlations between these mapped markers and the trait of interest in controlled breeding experiments to gain insight into the regions of the genome that control the trait.

1.1.2 Experimental Design

The paradigm for the programs in the ***QTL Cartographer*** package is that of highly inbred lines with very little genetic variation within lines but variation between lines. We shall refer to these inbred lines as parental lines and denote them by the symbols P_1 and P_2. As a general rule, the P_1 lines will correspond to the "high" lines with respect to the trait of interest, that is they will have mean values larger than the P_2 or "low" lines. These parental lines can be crossed to produce F_1 lines which are heterozygous for both markers and QTLs. One can then cross the F_1 populations with either parental line to produce backcrosses. The symbols B_1 and B_2 will refer to backcrosses involving the P_1 and P_2 lines, respectively. Alternatively, the F_1 lines can be intercrossed to produce F_2 lines.

In each of these cases, the resultant lines will have variation in both the trait of interest and the underlying quantitative trait loci and marker genotypes. These crosses are illustrated in Figure 1.1. We can then look for correlations between the trait in question and marker genes that have been mapped previously.

We have also included options for more complex experimental designs, including recombinant inbred lines, general F_t lines produced by selfing or random crossing of F_{t-1} lines, *etc.* The programs in the ***QTL Cartographer*** system will need to know the type of experimental design used to create the data. This design is encoded by a string of characters. If the letter i stands for some integer, then the possible crosses will be B_i, SF_i, RF_i, RI_i, $T(XX)SF_i$ and $T(XX)RF_i$. The B stands for a backcross and the integer attached to it will indicate the parental line to which the F_1 line was crossed to (either 1 or 2). If there was repeated backcrossing to one of the parental lines, this can be indicated by attaching two integers to the B: B_{ij} indicates that there were j generations of backcrossing to parental line i. B_{11} is equivalent to B_1. SF_i stands for selfed intercross lines and the integer indicates the generation ($i = 2,\ 3,\ \ldots$). RF_i stands for randomly mated intercross lines. RI means recombinant inbred lines, and the integer can take on one of three values: 0, 1, and 2. A 1 indicates RI lines derived by selfing, a 2 by sib mating and a 0 means doubled haploid lines.

The T indicates that the data are the result of a test cross. For a test cross, genotyping is done on an intercross (SF_i or RF_i) and phenotyping on a cross derived from that intercross. The first part of the string, $T(XX)$ indicates that phenotyping is done on the XX population and the second part (SF_i or RF_i) indicates the genotyped population. XX can be a B_1, B_2, SF_i or D_3 for SF_i lines or B_1 or B_2 for RF_i lines. D_3 stands for Design III experiments (Cockerham and Zeng 1996).

All of the above experimental designs can be simulated, and all but the Design III experi-

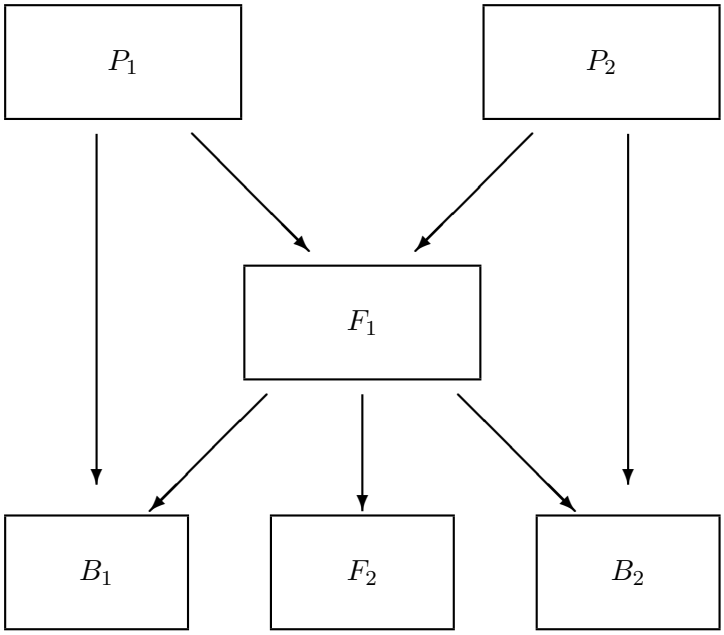

Figure 1.1: Basic Cross

Design	Code	Example
Backcross to P_i	B_i	B1
Backcross j times to P_i	B_{ij}	B13
Selfed generation i intercross	SF_i	SF3
Randomly mated generation i intercross	RF_i	RF2
Doubled Haploid	RI_0	RI0
Recombinant Inbred via selfing	RI_1	RI1
Recombinant Inbred via sib mating	RI_2	RI2
Testcross of SF_i to P_j	$T(B_j)SF_i$	T(B1)SF3
Testcross of SF_i for j generations	$T(SF_{i+j})SF_i$	T(SF4)SF3
Testcross of RF_i to P_j	$T(B_j)RF_i$	T(B1)RF3
Design III	$T(D3)SF_i$	T(D3)SF5

Table 1.1: Summary of Experimental Design Codes

ments can be analyzed. Table 1.1 lists all the experimental designs and their ***QTL Cartographer*** codes. The experimental designs of Table 1.1 can be specified in **Rcross** for simulations or in certain data input files (see Section 6.3.2).

1.1.3 Genetic Linkage Maps

A known genetic linkage map will be required for the analysis. A good genetic linkage map will comprise a set of Mendelian marker loci that are evenly spaced and span the genome. Average intermarker distances of 5 to 10 centimorgans would be optimal. We have provided ways to simulate linkage maps as well as to convert linkage map information into a format suitable for ***QTL Cartographer***. Presently the user has two options for genetic linkage map input. The first is a format designed for the ***QTL Cartographer*** system that allows for free annotation of the data file. An example is given in (6.1.2). A second option allows the user to import the results of a **Mapmaker/EXP** session. This is covered in more detail in (2.1) and (5.8.1).

1.2 Programming Philosophy

These programs were originally developed on a UNIX workstation. Consequently, the programming philosophy is heavily influenced by the UNIX operating system. All the programs have command line options which mimic those of regular UNIX commands. We have added interactive menus so as to make the programs more user friendly on Macintoshes and PCs running Microsoft Windows^{TM}.

There are a number of different programs in the package rather than one program that does everything. In this way, each program does a small job, and the user can combine the programs as a group to do a complete analysis. The user can examine the input and output files for each step and have a better idea of what the programs are doing. All input and output files are plain ASCII text. They can be transferred to any platform and viewed or edited there.

We have also been influenced by the Free Software Foundation in that we charge no fee for this program package. We have attempted to integrate these programs with other free software (most notably **Gnuplot** and **Mapmaker/EXP**).

1.3 Copyright Information and Acknowledgments

1.3.1 QTL Cartographer Copyright Information

1.3.2 Citing QTL Cartographer

In publications, you should cite our original short announcement (Basten, Weir, and Zeng 1994) and this manual.

- C. J. Basten, B. S. Weir and Z.-B. Zeng, 1994. Zmap–a QTL cartographer. In *Proceedings of the 5th World Congress on Genetics Applied to Livestock Production: Computing Strategies and Software,* edited by C. Smith, J. S. Gavora, B. Benkel, J. Chesnais, W. Fairfull, J. P. Gibson, B. W. Kennedy and E. B. Burnside. Volume 22, pages 65-66. Published by the Organizing Committee, 5th World Congress on Genetics Applied to Livestock Production, Guelph, Ontario, Canada.

- Basten, C.J., B.S. Weir and Z.-B. Zeng, 2004. QTL Cartographer, Version 1.17. Department of Statistics, North Carolina State University, Raleigh, NC.

1.3.3 Gnuplot Copyright Information

We suggest that you download and make use of the fine plotting package **Gnuplot** (Williams and Kelley 1993). which we use as the graphics engine to display the results of analyses. **Gnuplot** is freely available for UNIX, Macintosh and MS-Windows machines. It is quite easy to use, produces nice results and all the input files are plain text. We reprint the copyright information for **Gnuplot** verbatim:

```
GNUPLOT copyright information:

  Copyright (C) 1986 - 1993   Thomas Williams, Colin Kelley

Permission to use, copy, and distribute this software and its
documentation for any purpose with or without fee is hereby granted,
provided that the above copyright notice appear in all copies and
that both that copyright notice and this permission notice appear
in supporting documentation.

Permission to modify the software is granted, but not the right to
distribute the modified code.  Modifications are to be distributed
as patches to released version.

This software is provided "as is" without express or implied warranty.

AUTHORS

 Original Software:
   Thomas Williams,  Colin Kelley.

 Gnuplot 2.0 additions:
     Russell Lang, Dave Kotz, John Campbell.

 Gnuplot 3.0 additions:
     Gershon Elber and many others.

For more information on GNUPLOT, see the documentation
that comes with the program.
```

1.3.4 LINPACK Copyright Information

We have translated some of the FORTRAN procedures of LINPACK (Dongarra et al. 1979) into C. We have used all of the basic linear algebra subroutines (BLAS) as well as subroutines to do the QR factorization of the matrix $\mathbf{X}$ of the linear system

$$\vec{y} = \mathbf{X} \cdot \vec{b}$$

These include the subroutines *SQRST, STRSL, SPODI, SQRSL* and *SQRDC.* Not all of the optimizations have been translated. These subroutines are used quite extensively in the analysis modules. The original FORTRAN subroutines are Copyright (C) 1979 by the Society for Industrial and Applied Mathematics.

1.3.5 Numerical Recipes in C Information

We have made extensive use of the ideas from *Numerical Recipes in C* (Press, Flannery, Teukolsky, and Vetterling 1988). The source file "Utilities.c" contains subroutines for allocating memory that are derived from the functions for creating arbitrary offset vectors and matrices in Appendix D. We have also used modified versions of the subroutines listed in Table 1.2. The original subroutines are Copyright (C) 1987, 1988 *Numerical Recipes Software.*

subroutine	section
indexx()	8.3
moment()	13.1
sort()	8.2
gammln()	6.1
gammp()	6.2
gasdev()	7.2
gcf()	6.2
gser()	6.2
poidev()	7.3
betai()	6.3
beta()	6.1
betacf()	6.3

Table 1.2: Subroutines from *Numerical Recipes in C*

1.4 How to Get and Install QTL Cartographer

Point your web browser to *http://statgen.ncsu.edu/* and follow the link from the software submenu to ***QTL Cartographer***. You can then follow the link to the ftp site and click on the files you want to download.

QTL Cartographer is also downloadable via anonymous ftp at *statgen.ncsu.edu*. Use "ftp" as your username and your email address as the password. Here is an example.

```
username: ftp
password: basten@statgen.ncsu.edu
```

Next, change directory into the distribution subdirectory, **/pub/qtlcart**, and view what is available. For example,

```
ftp> cd /pub/qtlcart
ftp> ls
1.10b/
1.12f/
1.13g/
1.14d/
1.15d/
1.16c/
1.17f/
ChangeLog
QTLCartMac.sit.hqx
QTLCartUnix.tar.gz
QTLCartWin.zip
README
WQtlSetup.exe
data/
gnuplot-dist/
manual.pdf
perlscripts.pdf
swang/
```

Download the appropriate version. Presently, the following versions are available.

- **QTLCartWin.zip** is for Microsoft Windows. These are 32-bit applications. You will need an unzip utility to unpack this file. There should be an unzip utility in the Windows system folder.
- **QTLCartMac.sit.hqx** is for Macintoshes.
- **QTLCartUnix.tar.gz** is for UNIX, and can be used with Mac OS X in the UNIX environment and Windows (XP, 2000) in Cygwin.
- the **gnuplot-dist** directory contains the distributions of **Gnuplot** for various platforms.
- The **1.10b, 1.12f,** *etc* directories contain older versions of ***QTL Cartographer***.
- The **WQtlSetup.exe** file is the current graphical user interface version written by Shengchu Wang.
- The **data** subdirectory has the raw data from some publications.

- The **manual.pdf** and **perlscripts.pdf** are the documentation for the command line programs. They are included in the distribution files, so you may not need to download them separately. **ChangeLog** summarizes the changes to the programs and **README** should be read by all.

You can usually download a file by using the **get** command with a filename. On Macintoshes, using the server mode may require you to use the **put** command, as you are putting the files onto your local machine rather than getting them from the remote server. It is best to do the transfer in an empty subdirectory so that you don't inadvertently delete some important files. You will also want to download the **README** file if you don't already have a copy of it. The **README** file in the **/pub/qtlcart** subdirectory will often be more recent than the one in the archive.

The **manual.pdf** file is an Adobe Portable Document File of the manual and the UNIX manpages. It is contained within the distribution for each platform in a sudirectory **doc/pdf** (or the folder **pdf** inside the folder **doc**). You can view or print these files with Adobe's **Acrobat Reader**, which is freely available from Adobe at *http://www.adobe.com*.

The following sections indicate how to install the programs onto various computing platforms.

1.4.1 MS-Windows

Download the file **QTLCartWin.zip** in binary format to your computer's hard drive. Move the program to a directory where you want ***QTL Cartographer*** to reside. Use an unzip utility to unpack the MS-Windows distribution (often, this can be done with a double-click).

The programs will be unpacked in the directory you choose. You will want to do this in a directory created for ***QTL Cartographer***, so let's assume that it is **c:\qtlcart**.

You may also want to download **Gnuplot** for MS-Windows. In binary format, get the self-extracting archive **gnuplot.exe**. Put it in a subdirectory (say **C:\gnuplot**) and while in that subdirectory, run **gnuplot** from the DOS command line.

The programs can be run by double clicking their icons in the **Windows Explorer** application. An alternate method is to open a **Command Window** and type in the program names.

You can view the output files in any text editor, although you should be aware that some editors in MS-Windows cannot load large files. The **Notepad** application provided with the current (21st century) versions of Windows should work fine.

Another option for MS-Windows XP or MS-Windows 2000 is to download and install the freeware package **Cygwin** (*http://www.cygwin.org*) and use the UNIX version. The full install of **Cygwin** includes the **gcc** compiler and **perl**, so you will be able to run the scripts. Explore the **Cygwin** website for more information.

1.4.2 UNIX

Download the file **QTLCartUnix.tar.gz** in binary format from *statgen.ncsu.edu.* It is in the same directory that **README** file came from. On your local machine, create a subdirectory

for the distribution, then move the file **QTLCartUnix.tar.gz** to it. Uncompress and untar the file as follows:

```
% gunzip QTLCartUnix.tar.gz
% tar xf QTLCartUnix.tar
```

Follow these steps to compile and install ***QTL Cartographer***.

1. Move into the **src** directory and edit the file **LocalD.h**. It is annotated and you can follow the directions in the file if compilation doesn't work the first time.

2. You will also need to edit the **Makefile** and choose a compiler. The default is **cc**, which is a link to the **gcc** compiler under Macintosh OS X versino 10.3. If you don't have **cc**, you might try **gcc**. Finally, you will want to set the install directory. By default it is `BINDIR = /usr/local/bin`, but you can change it to whatever you wish. Note that to install the programs in the install subdirectory, you will need write permissions for that subdirectory.

3. Change into the root directory of the distribution and make the programs:

   ```
   % make install
   ```

4. The binaries will be in the *BINDIR* subdirectory defined above. Make sure that this subdirectory is in your path variable, and then **rehash**.

Presently, we use **gcc** version 3.3 under Macintosh OS X version 3.3. If you have troubles compiling, you may need to update your operating system or compiler.

If you would also like to have the **perl** scripts installed, **cd** into the **doc/scripts** subdirectory and follow the instructions in the **INSTALL** file there.

1.4.3 Macintosh

Macintosh OS X is built on a foundation of UNIX: We recommend using the UNIX version under OS X. If you are using OS 9 or earlier, then you will need a Macintosh with a PowerPC or newer central processing unit. Download the file **QTLCartMac.sit.hqx** and use **StuffitExpander** unpack the binaries and supplemental files. Some programs such as **Netscape** or **Fetch** will unbinhex the files for you, although they may require a helper application.

Once the **QTLCartMac.sit.hqx** file has been unpacked, you will have a folder called **QTLCartMac** with separate folders for the binaries, documentation and examples. The subfolder **cbin** has carbonized binaries, while the **bin** folder has binaries that do not require the Carbon libraries. To use the programs, open the appropriate binary folder and double click on the program you want to run. You will first be presented with a console window. All you need to do is click on **OK** to get to the interactive menu for setting options. You can also enter command line options in that box if you like.

1.5 Getting Help

All three distributions come with a **doc** subdirectory containing a **README** file in plain text format and five subdirectories. The subdirectories have documentation in various formats.

html This contains the manual pages in hypertext markup language. You can view these files in any web browser. They are also the files available online at *http://statgen.ncsu.edu/qtlcart*

ppt Contains a PowerPoint slide presentation that Chris Basten gives to introduce QTL Cartographer. You will need **Microsoft Powerpoint** to view it.

pdf Contains the ***QTL Cartographer*** manual in Adobe portable document format. You will need Adobe's **Acrobat Reader** to view and print this file. **Acrobat Reader** is free and can be obtained from *http://www.adobe.com.*

As of version ***QTL Cartographer*** version 1.16, **perlscripts.pdf** will be included in the **pdf** subdirectory. This document has a set of manual pages explaining the **perl** scripts for automation of certain tasks.

txt Contains documentation files in plain text. It includes some sample input files, announcements and a list of program changes.

scripts Contains a number of **C** shell and **perl** scripts to automate repetitive tasks such as bootstrapping and permutation tests. These scripts are mainly for **UNIX** platforms.

You can also get up to date documentation via the World Wide Web by pointing your web browser to

```
http://statgen.ncsu.edu
```

and following the link to ***QTL Cartographer*** from the *Software* menu.

1.5.1 Mailing List

The address for the mailing list server is *MajorDomo@statgen.ncsu.edu.* Please join the mailing list for ***QTL Cartographer***. It will be a forum for problems you may have in using the programs, and we will post announcements of updates and bug fixes. To subscribe, send the following two line message to the server:

```
subscribe qtlcart
end
```

The second line in the message stops MajorDomo from interpreting your .sig. Note that the subject line of your mail message will be ignored. If the subscription was successful, you will receive a confirmation note saying as much. You may also put an email address after the "subscribe qtlcart" (on the same line) to subscribe that address:

```
subscribe qtlcart basten@statgen.ncsu.edu
end
```

A message like the above with "unsubscribe" rather than "subscribe" would unsubscribe the address. The command "help" would cause the server to return a list of commands that can be sent to the MajorDomo server. Remember that all commands should be directed to *MajorDomo@statgen.ncsu.edu,* while messages for people on the list go to *qtlcart@statgen.ncsu.edu.*

1.5.2 Bug Reports

Send any bug reports to *qtlcart-bug@statgen.ncsu.edu.* There is certain information that will greatly aid in diagnosing the problem. The ***QTL Cartographer*** distribution should come with a file called **problems.txt** with the following questions in it:

1. Computing platform
   (a) What machine are you using? Is it a
      i. UNIX based workstation?
      ii. PC running Windows?
      iii. PowerPC based Macintosh?
   (b) What operating system is it running?
   (c) What is the version of the Operating system?
   (d) How much memory and free hard disk space do you have?
2. Programs
   (a) Which program is giving you trouble, and what parameter values were used?
   (b) Are the input files simulated or real?
   (c) Would it be possible to send me the input files, the log file and the resource file (qtlcart.rc)?
   (d) When the program crashed, did it give any diagnostics?
   (e) When did you download the programs?
   (f) What is the version number? (This is valid for programs downloaded after 1 January 1996, and supersedes the previous question.)

When reporting a problem, try to include the answers to all of the questions above. Some of them may not be relevant for you particular case and can be ignored. Email is generally the best way to report problems as the messages stay on a queue until they are dealt with.

One of the most difficult steps in using the ***QTL Cartographer*** system is to reformat datasets. Question 2(c) above asks whether you would be willing to send us your data in order to diagnose a problem. We would like to emphasize that if you send us your data files, they will be kept in the strictest confidence and deleted upon your request. If you have published data sets and would like to make them available for the scientific community, we would be happy to put them on our ftp server.

1.5.3 Contacts

For any other problems with ***QTL Cartographer***, contact Christopher J. Basten via any of the methods listed in Table 1.3. In general, email is the best method for indicating a problem. Chris may not always get back to you right away, but will try to.

Name	Dr. Christopher J. Basten
Email	basten@statgen.ncsu.edu
Phone	(919)515-1934
Fax	(919)515-7315
Address	Bioinformatics Research Center 1523 Partners II Building North Carolina State University Raleigh, NC 27695-7566 USA
Courier	840 Main Campus Drive Raleigh NC 27606
MajorDomo	MajorDomo@statgen.ncsu.edu
Bug Report	qtlcart-bug@statgen.ncsu.edu

Table 1.3: Contact for Help

1.6 General Usage of the Programs

The programs in the ***QTL Cartographer*** suite all have the same look and feel and are heavily influenced by UNIX programs. They can be used as command line programs, or in an interactive mode where a menu of options is presented. Some command line options that are common to all the programs are discussed in 1.6.1. The new user should become familiar with these options. In addition to the command line interface, all the programs have an interactive menu for setting options. The user need only start up any program in the suite and a list of options will appear. Selecting the number of an option will allow the user to change the value of the option. When all options are set to the user's satisfaction, choosing a zero '0' will cause the program to run. Choosing the penultimate numbered option will allow you to exit the program without changing any files. The last option saves any parameters you have set before exiting.

1.6.1 Options for all programs

Table 1.4 shows the command line parameters that are valid for all the programs in ***QTL Cartographer***.

Working directory

A working subdirectory (folder) to hold all input and output files is a convenient way to organize your work. We suggest using a different subdirectory (folder) for each data set. In the UNIX

world, you can simply change into the working directory and run the programs. In the Macintosh and MS-Windows environs, you need to run the programs from where they reside and specify where the working directory is. Use the **-W** command line option to specify a working directory, or set it in the interactive menu. Be sure to follow the conventions of the particular operating system that you are working on. For UNIX, you might specify it as

```
-W /home/myaccount/qtlcart/workdir
```

While for MS-Windows it might look like

```
-W C:\qtlcart\workdir
```

And on a Macintosh, assuming that your Hard drive is called "MacintoshHD",

```
-W MacintoshHD:qtlcart:workdir
```

The programs will automatically add a file separator to the end of the path if you don't put it in. Thus

```
-W MacintoshHD:qtlcart:workdir:
```

is equivalent to the first incarnation of the Macintosh work directory. The Macintosh file separator ":" is equivalent to the DOS "\" and the UNIX "/".

Option	Default	Explanation
-e	qtlcart.log	Error and Log File
-s	795793333	Random Number Seed
-h	(off)	Show help and exit
-R	qtlcart.rc	Resource File
-W	(none)	Working Directory
-A	(off)	Automatic mode
-X	qtlcart	Filename Stem
-V	(on)	Verbosity

Table 1.4: Command Line Options for all programs

You may also use relative pathnames for the working subdirectory. In the UNIX and Windows environments, a single period (.) means from here and a pair of periods (..) indicates one higher directory level. Thus,

```
-W ..\workdir
```

would indicate go up one level from the binary subdirectory, where you will find a **workdir** subdirectory. In UNIX it might look like

```
-W ../workdir
```

For the Macintosh you use extra colons: If the binaries are in the **bin.ppc** folder inside the **qtlcart** folder, then

```
-W ::workdir:
```

would indicate that there is a folder called **workdir** in the **qtlcart**, whereas

```
-W :workdir:
```

would indicate that the **workdir** folder is inside the **bin.ppc** folder.

Listing options

Using the **-h** option will print out a list of all command line options and their values. The program will then exit without doing anything. I find this most useful when I just want a reminder of what the programs expect. This may not seem as useful now that there is an interactive menu to set options, but if you only want to use the programs in batch mode, it is a quick way to see what the values of all parameters are.

Random Number Seed

Many of the simulation programs make use of a pseudo-random number generator that requires a seed. If none is provided, the number of seconds since some date in the past is used. The **-s** option allows you to specify a seed for the random number generator. You can use this to repeat simulations to see if the same answers are obtained. If you don't use this option, the random number seed is set to the number of seconds since some arbitrary past date (for example, 1 January 1970 for Sun Workstations). The random number seed is printed to the output files of the programs on the first line. This means that if you don't specify a random number seed, each file should have a unique identifier associated with it. This identifier will also be written to the log file.

If you run a shell script loop on a really fast machine, each turn of the loop may take less than one second. This can be important if you are using the default random number seed, in that separate invocations of the program in a loop might get the same random number seed. For example, consider the pseudo code to produce 1,000 simulated data sets:

```
i = 1
while (i < 1000) {
        Rcross -A -V -o qtlcart.cro.$i
        i = i + 1
}
```

It is possible that each loop in the above code could take a fraction a second, and groups of output files would then be identical. You would want to put a **sleep** command in the loop to avoid this problem.

Verbosity

For debugging purposes and simply to inform the user about what is happening, many diagnostic messages will be printed out as the programs run. The user can turn these diagnostic messages

off. When the messages are displayed, we refer to this as the verbosity mode. The verbosity mode can be turned off by using the the **-V** option. This means that the time and summary of options will not be printed on the standard output at runtime. This is a useful flag for batch files. Most of the messages printed to the screen are also printed to the log file.

Automatic Mode

By default, when the user starts up a program, an interactive menu for setting program options is displayed. The opposite of this is the Automatic mode. The **-A** flag turns off the interactive setting of program options. This is another flag useful for batch programming. The automatic mode should only be used by those familiar with the ***QTL Cartographer*** programs.

Debugging Level

This option is off by default. Using **-D** or The **-D0** will set the debugging level to 0, which is the same as being off. Using **-D1** will set the debug level to 1, which means that all memory allocation and deallocation will be recorded to a file **memalloc.txt** in the current working directory. Finally, **-D2** will tell some of the programs to print extra diagnostic messages to the log file.

Resource File

A resource file is an ASCII text file that keeps track of the parameters that the user specifies in using the programs. The same file is read and updated by all the programs in the suite. You can specify a resource file using the **-R** option. It is **qtlcart.rc** by default and should be in the directory that you are currently working in (for UNIX machines) or where the binaries are (for PCs and Macintoshes). If you change any options (either via the command line or the menus), they will be saved to the file specified. If you decide to use a file other than **qtlcart.rc** as the resource file, you will need to specify it for each program you run.

Initially, the user may want to create a resource file with two lines to specify the working directory and a "stem" for filenames. Here is an example of a resource file for the Macintosh version of the programs:

```
-workdir      ::test:        # (The working directory)
-stem         corn           # (Stem for filenames)
```

The working directory must be specified according to the rules of the operating system. This was explained in using the **-W** option in previous section. In the above example, a relative pathname was used. The programs will assume that there is a folder called **test** in the folder one level up from the folder that the applications reside in. The analogous lines for the MS-Windows version would look like:

```
-workdir      ..\test\        # (The working directory)
-stem         corn            # (Stem for filenames)
```

The working directory must exist before you run ***QTL Cartographer***.

Filename stem

The filename stem is an important concept in the usage of ***QTL Cartographer***. Beginning with version 1.12, the ***QTL Cartographer*** programs utilize the filename stem "qtlcart". All files are then named using this stem and filename extensions relevant to the filetype. In the resource file example above, the "-stem" entry specifies "corn" as a stem for filenames. This means that when new files are created, they will have the stem "corn" followed by a logical extension. An example would be **corn.map** for a genetic linkage map. With some practice, you will know the contents of a file by its extension. You can set the filename stem on the command line with the **-X** option.

Log File

It's often useful to keep a log of the work done using the programs. The **-e** option can be used to specify the log or error file. Each time a program in the ***QTL Cartographer*** system runs, a summary of all the parameters and options is written to the log file. The file also keeps track of when the program was run and may contain other diagnostic information. The log file is appended to with each run rather than overwritten.

Remember that the log file is appended to during each invocation of any of the programs. This is something to keep in mind if you do a bootstrap in a batch file. After a thousand replications, the log file will tend to grow large. The batch file examples included with the ***QTL Cartographer*** system (see 2.4.2) take this into account by saving a copy of the log file before running the bootstrap, and deleting the large (and unnecessary) temporary log file at the end.

Interactive Mode

The default behavior for the ***QTL Cartographer*** programs is to present the user with a menu of numbered options. This menu is in a loop, so the user can pick options and change them, one at a time. when satisfied that the proper options have been set, selecting "0" will tell the program to continue. There will always be an option to quit without doing anything. This will be the last numbered option.

When "0" is chosen, the programs will present a summary of the options and continue. At termination, the options will be written to the resource file so that the options and parameter values are remembered.

A General Test

Suppose you have unzipped the Windows version of ***QTL Cartographer*** and it all resides in **c:\QTLCartWin**. Further, suppose you create a subdirectory of **c:\QTLCartWin** called **test**. If you open a command window and **cd** to the **c:\QTLCartWin\bin** directory, you could test the ***QTL Cartographer*** system with the following set of commands.

```
c:\QTLCartWin\bin> Rmap -W ..\test -X test -A -V
c:\QTLCartWin\bin> Rqtl -A -V -q 4
```

```
c:\QTLCartWin\bin> Rcross -A -V
c:\QTLCartWin\bin> Qstats -A -V
c:\QTLCartWin\bin> LRmapqtl -A -V
c:\QTLCartWin\bin> SRmapqtl  -V
c:\QTLCartWin\bin> Zmapqtl -M 6  -V
c:\QTLCartWin\bin> Eqtl -A -V
```

(Note that **c:\QTLCartWin\bin**> is the command line prompt). You would then find a number of files in the **c:\QTLCartWin\test** subdirectory. Look at their contents.

For the Macintosh versions of the programs, you could double click on the icons and put the command line options in the box that pops up. For example, you would double-click the **Rmap** program and put **-W ::test -X test -A -V** in the box for the first command.

1.6.2 Filenaming Conventions

The ***QTL Cartographer*** system reads and creates many files and each has a default name. For example, the default output file for **Rmap** is **qtlcart.map**. We find it convenient to specify a filename stem and allow for the filename extension to indicate which program created it, and what it contains. Suppose we were working on a corn data set. We might use "corn" as the filename stem. Then **Rmap** would write its output to **corn.map** and its error messages to **corn.log**. **Rqtl** would write its output to **corn.qtl**, *etc.* Table 1.5 summarizes the standard file name extensions in the ***QTL Cartographer*** system. Beginning with version 1.12, the default behavior of ***QTL Cartographer*** is to use a filename stem: If none is given, then "qtlcart" will be the stem. Unless specifically written in the **qtlcart.rc** file, the old default names of **Rmap.out**, **Rqtl.out**, *etc.,* will no longer be used. These old default names will be used as filetype identifiers. In the output files, there will be a token "-filetype" followed by a token from the fourth column of Tables 1.5-1.6. Note that **Zmapqtl** creates some interim files, and that **Preplot** will create many other files in addition to the **Gnuplot** control file: See Section 4.2 for details. The "-filetype" specifier will greatly aid programs such as **Rmap** and **Rcross** in translating files. As ***QTL Cartographer*** develops, this feature will be used more extensively. Once the stem is set in the menu, it will be remembered as long as a resource file is present.

In the interactive menu, if you pick an item to change (say a filename), you can wipe it out by inputting a solitary period. This way, if you had specified an input file in an earlier run, you can delete it.

In addition to the files specified in the table, we assume that files with extensions "maps" and "raw" are **Mapmaker/EXP** genetic linkage map and raw data files, respectively. These and other files recognized by ***QTL Cartographer*** are listed in Table 1.6.

Program	Extension	Contents	-filetype
Rmap	.map	genetic linkage map	Rmap.out
Rqtl	.qtl	QTL model	Rqtl.out
Rcross	.cro	data file (markers, traits)	Rcross.out
Qstats	.qst	**Qstats**Analysis	Qstats.out
LRmapqtl	.lr	Single Marker Analysis	LRmapqtl.out
SRmapqtl	.sr	Stepwise Regression Analysis	SRmapqtl.out
Zmapqtl	.z	IM-CIM Results	Zmapqtl.out
JZmapqtl	.z#	Multitrait Results	JZmapqtl.out
Prune	.mpb	pruned genetic linkage map	Rmap.out
Prune	.crb	pruned data file	Rcross.out
Preplot	.plt	**Gnuplot** Control file	Preplot.plt
Eqtl	.eqt	Summary of **Zmapqtl** Results	Eqtl.out
MImapqtl	.mim	**MImapqtl** results	MImapqtl.out

Table 1.5: Standard Filename Extensions and File types for Output Files

Program	Example	Contents	-filetype
Rmap	qtlcartm.inp	genetic linkage map	map.inp
Rmap	qtlcart.maps	**Mapmaker/EXP** output	mapmaker.maps
Rqtl	qtlcartq.inp	genetic model file	qtls.inp
Rcross	qtlcartc.inp	data file (markers, traits)	cross.inp
Rcross	qtlcart.raw	**Mapmaker/EXP** input	mapmaker.raw
Zmapqtl	qtlcart.z3c	Perm. test interim file	ZipermC.out
Zmapqtl	qtlcart.z3e	Perm. test interim file	ZipermE.out
Zmapqtl	qtlcart.z3a	Bootstrap interim file	Ziboot.out
Eqtl	qtlcart.z3b	Bootstrap summary file	Ziboots.out
Zmapqtl	qtlcart.z3i	Jackknife interim file	Zijack.out
MImapqtl	qtlcart.res	Residuals dataset file	Rcross.out
MImapqtl	qtlcart.mqt	Model output file	Rqtl.out
Eqtl	qtlcart.z3j	Jackknife summary file	Zijacks.out
JZmapqtl	qtlcart.zr	Input for **MultiRegress**	JZmapqtl.zr
MultiRegress	qtlcart.mr	Output of **MultiRegress**	qtls.inp

Table 1.6: Miscellaneous Files and File types

Chapter 2

Simulating/Reformatting Data

The first phase in using ***QTL Cartographer*** is to create some data. You have two options for this: You can either simulate a data set or collect one yourself. The end result will be to have two files. One will contain the information on a genetic linkage map (marker order, chromosome assignment and recombination fractions) and the other a data set from a cross, which contains the markers, trait values and other explanatory variables. ***QTL Cartographer*** cannot create a genetic linkage map from a data set: You will have to use another program such as **Mapmaker/EXP** for that task.

Figures 2.1–2.2 present a schematic of the data simulation/reformatting process. There are four main programs involved in this phase: **Rmap**, **Rqtl**, **Rcross** and **Prune**. **Rmap** is a program designed to create random genetic linkage maps, or reformat linkage maps that were prepared by **Mapmaker/EXP**. **Rqtl** is a program that creates a genetic model for simulation. One can specify the positions, effects and the number of loci for each trait, or have the program do it randomly. Finally, **Rcross** uses the genetic linkage map and the model to create a random data set, by simulating a cross. **Rcross** can also reformat **Mapmaker/QTL** raw data files or specially formatted data files. The fourth member of this group is **Prune**. With **Prune**, the user can eliminate individuals, markers or traits from the data set. In addition, **Prune** allows one to bootstrap or permute the data, as well as to simulate missing or dominant markers.

Regardless of whether the data are simulated or real, the important output files from this step are the genetic linkage map and the data set. We will refer to these files as **qtlcart.map** and **qtlcart.cro**, although you can name them anything you like. In fact, we generally decide on a filename stem and use filename extensions to indicate what is in the various files. If we were working on a corn data set, we might have files **corn.map** and **corn.cro** for the genetic linkage map and marker/trait data set, respectively. The naming scheme would be consistent throughout the analysis.

One note on the behavior of **Rmap**, **Rqtl** and **Rcross**: If you choose to translate a data file, then the parameters for simulations are unnecessary and they disappear from the interactive menu. If you specify no input file for any of these programs (by entering a period "." all by itself for the input filename), then the simulation parameters will reappear for the user to change.

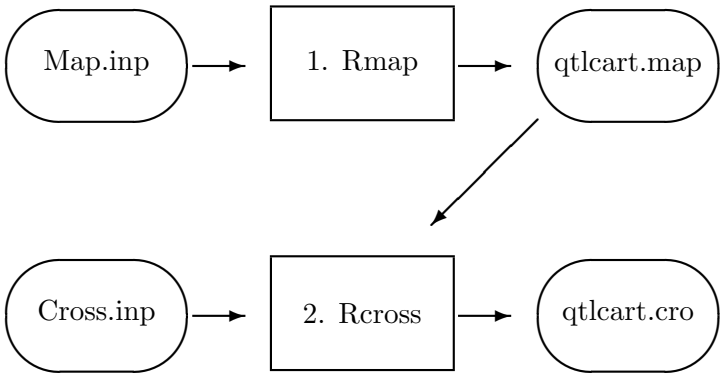

Figure 2.1: Reformatting Data

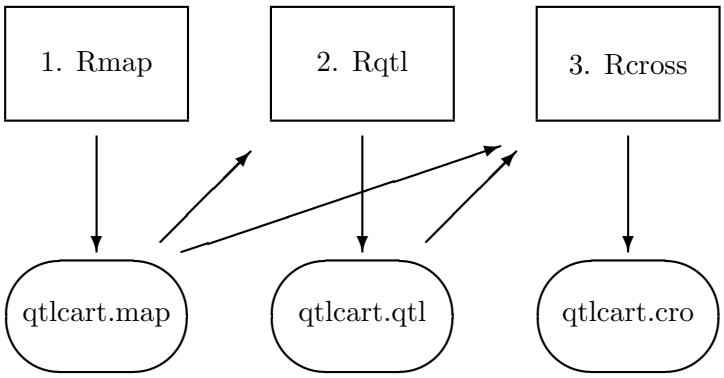

Figure 2.2: Simulating Data

2.1 Rmap

Originally, the program **Rmap** was designed to simulate a genetic linkage map. The "R" in **Rmap** was meant to convey the meaning of "Random Map". Since then we have included the ability to translate genetic linkage map information from various formats into that required by the ***QTL Cartographer*** system. Thus, the "R" can now mean reformat or random.

If you have no data, you can simulate a genetic linkage map. **Rmap** allows the user to specify the number of chromosomes, markers per chromosome and average intermarker distance for the simulation. You can also specify standard deviations for the latter two quantities. This would yield a simulated map that better approximates one that you might actually produce in the lab. Finally, you can also specify whether you want some genetic material outside the most telomeric markers on the chromosomes.

Rmap can also read in files in three formats. The first format is the same as its output format. We will refer to this as "Rmap.out" filetype format. This feature is provided so that you can create as set of output files that **Gnuplot** can read and display a graphic representation of your markers.

The second format is that which is produced by **Mapmaker/EXP** (Lander et al. 1987; Lincoln et al. 1992): We will refer to it as a "mapmaker.maps" filetype format. **Rmap** will read in the **Mapmaker/EXP** output and reformat into the "Rmap.out" format. The third format is defined in Section 6.1.2 and in the file **map.inp** included with the distribution of the programs. Remember: **Rmap** will overwrite output files. If you specify an output file that already exists, **Rmap** will destroy it when creating a new file. For this reason, we recommend that all work is done in a working subdirectory on copies of the original input files.

2.1.1 Simulating a Map

As an exercise in learning to use the programs, you can simulate a genetic linkage map. The main parameters that you will need to specify are the haploid number of chromosomes, average number of markers per chromosome, and average intermarker distance between consecutive markers. You can also simulate linkage maps in which the telomeres don't have marker information.

Figure 2.3 shows how **Rmap** simulates a genetic linkage map where $M_{i,j}$ is marker j on chromosome i. We denote the number of chromosomes by c, the average number of markers per chromosome by m and the average intermarker distance by d in centimorgans. Furthermore, the average amount of "tail" DNA (DNA outside the most telomeric markers) will be specified by t, again in centimorgans. The standard deviations of m and d will by σ_m and σ_d, respectively. All of these variables can be specified by command line options, the resource file or by the interactive menu. The standard deviation of t will be $\sigma_t = (\frac{t}{d})\sigma_d$. For the ith chromosome, **Rmap** decides how many markers are on that chromosome (m_i) by picking a random number from a normal distribution with mean m and standard deviation σ_m. Once this is done, the amount of DNA between consecutive markers ($d_{i,j}$) is simulated as a normal random variable with mean d and standard deviation σ_d. Finally, the amount of telomeric or tail DNA ($d_{i,0}, d_{i,m_i}$) is simulated as a normal random variable with mean t and standard deviation σ_t. Setting a standard deviation equal to zero means that the quantity in question is not a random variable, but set equal to its mean value.

The parameters c, m, d, t, σ_m and σ_d can be set using the command line options of Table 2.1 or in the interactive menu. Note that if an input file is specified, all these parameters are ignored and **Rmap** attempts to translate the input file.

An alternate method of simulating the genetic linkage map can be invoked by changing the simulation mode parameter from 0 to 1 using the *-M* command line option. Figure 2.4 presents the method graphically. The length of chromosome i (d_i) will be normally distributed with mean d and standard deviation σ_d. The number of markers on chromosome i (m_i) will still be normally distributed with mean m and standard deviation σ_m, but will be placed on the chromosome following a uniform distribution on the interval $[0, d_i]$. You should set the values of d and σ_d to appropriate levels, as they are for chromosome length rather than intermarker distance in this mode. For example, if you want roughly the same results from this mode as

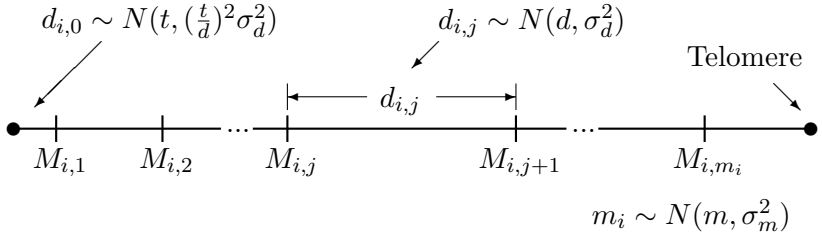

Figure 2.3: Chromosome i simulated under mode 0

that in the original, then set $d = 16 \times 10 = 160$ in this mode.

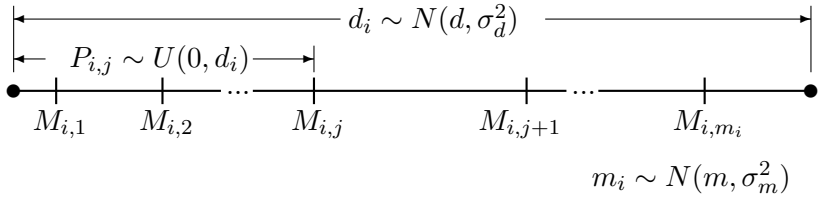

Figure 2.4: Chromosome i simulated under mode 1

2.1.2 Using MAPMAKER/EXP files

QTL Cartographer has the added capability of reading map files generated by **Mapmaker/EXP** (Lander et al. 1987; Lincoln et al. 1992). Genetic marker order and chromosome assignment may be accomplished using **Mapmaker/EXP**. Once map order is established, chromosomes may be saved to external files using the following **Mapmaker/EXP** commands (in **Mapmaker/EXP**):

```
make chromosome c1
seq M1 M2 M5 M4 M3
attach c1
framework c1
```

A chromosome c1 is defined, and the marker order (for example: M1, M2, M5, M4, M3) assigned. The "attach" and "framework" commands tell **Mapmaker/EXP** to save this marker order on chromosome c1. See Section 5.8.1 for a more detailed example of using **Mapmaker/EXP** to create the genetic linkage map.

After all chromosomes are defined and marker order assigned, exit **Mapmaker/EXP**. You will find files in your directory with the extensions, "*.data", "*.maps", "*.traits", "*.xmaps". The "*.raw" file contains the original genotype and phenotype information. The "*.maps" file contains the saved marker order per assigned chromosome, as well as the estimated recombination fractions between each marker in the established order. On MS-DOS machines, the

extension may be "*.map" rather than "*.maps". It would be a good idea to rename this file with a "*.mps" ending, so as not to confuse ***QTL Cartographer*** with its own genetic linkage map file.

The map order, chromosome, and recombination fraction estimate information may be used in ***QTL Cartographer*** by specifying "*.maps" as the input file for **Rmap**. The "*.raw" file is the input for the **Rcross** utility.

2.1.3 QTL Cartographer user input format

The third format is one defined for the ***QTL Cartographer*** system. It is similar to the **Mapmaker/EXP** output format, but has commands embedded in the file to allow the program to read in the data more easily. There is an example and further explanation of this format in Section 6.1.2. It can be annotated quite freely; the example file **map.inp** is self documenting.

2.1.4 Command Line Options

Table 2.1 summarizes the command line options for **Rmap**. Most of these were explained in 2.1.1. The default options in Table 2.1 would produce a genetic linkage map on four chromosomes with 16 markers each. The markers would be equally spaced at 10 centimorgan intervals and would span the genome.

Option	Default	Explanation
-i		Input File
-o	qtlcart.map	Output File
-f	1	Map Function
-p	0.0	Map function parameter
-g	1	Output Flag
-c	4	Chromosomes
-m	16	Markers per Chromosome
-vm	0.0	Standard deviation of Markers per Chromosome
-d	10.0	Intermarker Distance (cM)
-vd	0.0	Standard deviation of Intermarker Distance
-t	0.0	Tails (Flanking DNA, in cM)
-M	0	Simulation Mode (0,1)

Table 2.1: Command Line Options for Rmap

Map Function

A map function is a mathematical relationship between recombination probabilities and map distances measured in centimorgans or Morgans. ***QTL Cartographer*** presently allows for eight map functions specified by an integer. The numbers 1, 2 or 3 correspond to the Haldane, Kosambi and Morgan (formerly Fixed) mapping functions, respectively. The default is the

Haldane mapping function. If r corresponds to the recombination frequency between a pair of markers and d_M is the distance between them in Morgans, then the Haldane mapping function is defined by

$$d_M = -\frac{1}{2}\ln(1-2r) \tag{2.1}$$

$$r = \frac{1}{2}[1-\exp(-2d_M)] \tag{2.2}$$

The Kosambi function is

$$r = \frac{1-\exp(-4d_M)}{2[1+\exp(-4d_M)]} \tag{2.3}$$

$$d_M = \frac{1}{4}\ln[\frac{1+2r}{1-2r}] \tag{2.4}$$

and the Morgan function assumes $d_M = r$, which is complete interference. All eight mapping functions are discussed at length in Ben Lui's book (Liu 1998): We direct the reader there for the details. Table 2.2 lists the mapping functions and their integer codes for QTL Cartographer. Some of these map functions require an extra parameter. This parameter can be set in the **Rmap** menu. See Section 10.3.1 of Liu (1998) for the details.

Code	Reference	Note
1	Haldane (1919)	default
2	Kosambi (1944)	
3	Morgan (1994)	"Fixed"
4	Carter and Falconer (1951)	
5	Rao et al. (1979)	$0 \leq p \leq 1$
6	Sturt (1976)	L
7	Felsenstein (1979)	$-\infty < K < \infty, K \neq 2$
8	Karlin (1984)	binomial, $N > 0$

Table 2.2: Command Line Options for Rmap

Output Flags

The output flag takes on values of 1, 2 or 3. A 1 indicates that **Rmap** should output a file in the "Rmap.out" format. A 2 indicates that a set of files that can be plotted in **Gnuplot** should be created while a 3 indicates that both should be done. The option to display the map in **Gnuplot** allows a general overview of the spacing of markers. If you choose to create the **Gnuplot** files, then **Rmap** will write one file per chromosome summarizing the linkage information. Each file will have two columns: The first indicating the position of the marker from the telomere and the second for the chromosome number. The file for chromosome 1 will be **Chrom.1**, and other files are named accordingly. Finally, a control file, **Chrom.plt**, will have the plotting commands understood by **Gnuplot**. This file should be loaded by **Gnuplot** to view the linkage map. Marker names are not written on the map.

Input Files

Again, note that if an input file is specified, all options from "Chromosomes" down in Table 2.1 will be ignored and **Rmap** will attempt to translate the input file. Remember that **Rmap** overwrites any files with the same name as its output file, so avoid giving your input and output files the same name.

2.2 Rqtl

Given a genetic linkage map, **Rqtl** can place a random set of quantitative trait loci on the map. The program simulates the positions and effects (additive, dominance and epistatic) of the QTL. It can also reformat a given set of QTLs defined in an input file of filetype "qtls.inp" that is explained in Section 6.2.1. The given set of QTLs might be made up by the user, or a set of estimates from a previous analysis of a data set. Table 2.3 presents the command line options for **Rqtl**. The default values from the table tell **Rqtl** to simulate nine QTLs for one trait.

For simulations, the user can specify the average number of QTLs per trait, the number of traits, and parameters for dominance and additive effects. Epistatic effects are simulated with the same parameters used for the dominance effects. We use the convention that Q_1 alleles are from for P_1 lines and Q_2 from P_2 lines.

The output file encodes QTL with their positions and effects. A QTL is defined with a line beginning in "-l" and followed by its number, chromosome, left flanking marker, two recombinatin fractions and its additive and dominance effects. Here is an example of a set of simulated QTL:

```
-k      2   for trait -number  1
#       #  ..Chrom..Markr.  .RecombiL.  .RecombiR.  .Additive.  .Dominance
-l      1       2       9       0.0477      0.0475      0.2326      0.0000
-l      2       3       8       0.0906      0.0001      0.1687      0.0000
```

QTL number 1 is on chromosome 2 following marker number 9. Marker 9 is thus the left flanking marker and has a recombination frequency with the QTL of 0.0477. The right flanking marker would be marker 10 on chromosome 2, and it has a recombination fraction of 0.0475 with the QTL. Figure 2.5 graphically illustrates how QTL positions are encoded. QTL number 1 in the above example has an additive effect of 0.2326 and no dominance.

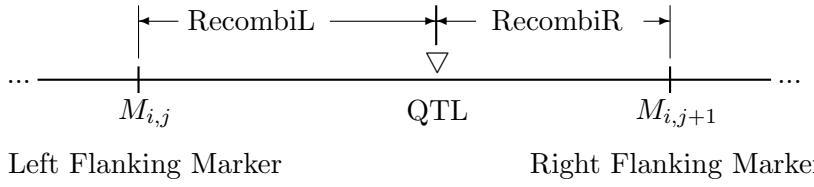

Figure 2.5: Simulated QTL position: Chromosome i near marker $M_{i,j}$

Dominance can take on the values 1, 2, 3 or 4. 1 means no dominance, while 2 means Q_1 is dominant and 3 means Q_2 is dominant. A value of 4 means that dominance for each QTL will be random in magnitude and sign. The degree of dominance will be a Beta random variable d with shape parameters $\beta_1,\ \beta_2$. The density function for d is

$$f(d) = \begin{cases} \frac{d^{\beta_1-1}(1-d)^{\beta_2-1}}{B(\beta_1,\beta_2)} & \beta_1,\ \beta_2 > 0;\ -1 \leq d \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where

$$B(\beta_1, \beta_2) = \frac{\Gamma(\beta_1)\Gamma(\beta_2)}{\Gamma(\beta_1 + \beta_2)} \tag{2.6}$$

and $\Gamma(x)$ is the gamma function

$$\Gamma(x) = \int_0^\infty y^{(x-1)} e^{-y} dy$$

Epistatic effects are generated from the same distribution as the dominance effects. For a k QTL model, there are $2k(k-1)$ potential dominance effects. For each unordered pair of loci, there are Additive by Additive, Additive by Dominance, Dominance by Additive and Dominance by Dominance terms, and thus $4k(k-1)/2$ possible epistatic interactions. Only a proportion of these will be nonzero, with that proportion specified by the **-E** option. The proportion should be in the range $[0.0,\ 1.0]$.

Option	Default	Explanation
-i	(None)	Input File
-o	qtlcart.qtl	Output File
-m	qtlcart.map	Genetic Linkage Map File
-t	1	Number of Traits
-q	9	Number of QTL per Trait
-b	2.0	Additive effect parameter beta
-1	2.0	Dominance effect parameter β_1
-2	2.0	Dominance effect parameter β_2
-d	1	Dominance
-E	0.0	Porportion of Epistatic Effects
-M	0	Simulation Mode

Table 2.3: Command Line Options for Rqtl

The additive effects of the QTLs are independent, identically distributed random variables sampled from the gamma distribution (Zeng 1992, page 993, equation 12) and reprinted here:

$$f(a) = \frac{\beta^\beta a^{\beta-1} e^{-a\beta}}{\Gamma(\beta)}, \qquad 0 < a < \infty, \qquad 0 < \beta < \infty \tag{2.7}$$

The shape parameter β allows a wide variety of different genetic models to be generated. The additive effect of substituting an Q_1 allele for an Q_2 allele is a. When multiple traits are

simulated, the number of QTLs per trait is simulated as a random variable with mean specified by the **-q** option.

Simulated QTL are placed on the genome following a uniform distribution with the restriction that all QTL for a specific trait must reside in different intervals. If the simulation mode is set to zero (using **-M 0**), then **Rqtl** imposes the further restriction that all QTL will have at least one open interval between them. These restrictions mean that the number of markers will limit the number of QTL that can be simulated. QTLs for different traits are simulated independently.

If an input file is specified, then it is translated into a format readable by **Rcross** and the options in Table 2.3 from "Number of Traits" and below are ignored. The input file format "qtls.inp" is defined in Section 6.2.1. This input file format will allow a wide variety of genetic models to be simulated.

2.3 Rcross

Rcross uses the information generated by **Rmap**and **Rqtl**and randomly simulates a data set. Alternatively, it can also reformat **MAPMAKER** raw data files and " cross.inp" filetype formatted files. Table 2.4 presents the options for **Rcross**. The default values would create a simulated sample of 200 individuals backcrossed to P_1 with a heritability of 0.5 for the quantitative trait.

Option	Default	Explanation
-i	(None)	Input File
-o	qtlcart.cro	Output File
-m	qtlcart.map	Genetic Linkage Map File
-q	qtlcart.qtl	QTL Data File
-n	200	Sample Size
-c	1	Type of Cross
-H	0.5	Heritability
-I	0	Interactive flag
-g	0	Output format
-E	-1.000000	Environmental Variance (used if > 0)

Table 2.4: Command Line Options for Rcross

2.3.1 Simulating Data

Rcross will simulate a dataset using the genetic linkage map prepared by **Rmap** and the genetic model prepared by **Rqtl**. The user can specify the sample size, type of cross and heritability or environmental variance. An interactive mode allows the user to generate arbitrary crosses. **Rcross** can automatically generate backcrosses, intercrosses or any of the other experimental designs defined in Section 1.1.2. Below we describe how each individual is created. The process is repeated as many times as are necessary to get the sample size specified.

Generation of Individuals

For generating backcrosses or intercross samples, the parental lines are known. Individuals in the F_1 are all heterozygous, and all pairs of loci are in coupling. Samples derived from F_2 and later crosses need to take into account the different possible parents. This section explains how individuals are simulated in a general way.

We assume that there is one or two parental samples that will be used to create the next generation. Refer to these as lines 1 and 2. We assume monoecious, diploid individuals. To generate a new individual, one parent is selected from line 1 and one from line 2. If line 1 and line 2 are the same sample (for example, crossing two F_2 lines to form an F_3) then selfing is a possibility. Once the parents have been selected, gametes are produced, one from each parent.

The first step in producing gametes is to simulate recombination. We assume that the number of crossovers on each chromosome is distributed as a Poisson random variable with mean equal to the length of the chromosome in Morgans. A separate random integer is generated for each chromosome subject to the Poisson, and this indicates the number of crossovers on that chromosome. These crossovers are placed on the chromosome subject to a uniform distribution.

Once the crossovers are in place, gametes are generated. Starting with the first chromosome, one of the two homologs is chosen at random. This chromosome is followed until a crossover is encountered, at which point the other homolog is used. At the end of the first chromosome, a homolog from the second chromosome is chosen at random and the process continues. At the end, a gamete is created which contains the markers and QTLs. The gametes from each parent are then combined to form a new individual. Genetic values are calculated from the genotypes of the new individual using Cockerham's general genetic model for the partitioning of genetic variance (Cockerham 1954). Phenotypic values can then be generated based on the genetic variance and the heritability.

Phenotypic Values

Phenotypic values are calculated from the genotypic values for each individual for each trait. Each individual's phenotypic value is calculated from its genotypic value with an environmental effect determined by the heritability h^2. The individual's genotypic value is based on the alleles it inherited at the quantitative trait loci. To calculate genetic values, we use Cockerham's general genetic model (Cockerham 1954).

$$\begin{aligned} G_i &= \sum_{r=1}^{m} a_r x_{ir} + \sum_{r=1}^{m} d_r z_{ir} + \sum_{r \neq s} b_{rs}^{AA} x_{ir} x_{is} + \sum_{r \neq s} b_{rs}^{AD} x_{ir} z_{is} \\ &\quad + \sum_{r \neq s} b_{rs}^{DA} z_{ir} x_{is} + \sum_{r \neq s} b_{rs}^{DD} z_{ir} z_{is} \end{aligned} \tag{2.8}$$

The parameters a_r, d_r are the additive and dominance effects of QTL r. The b's are epistatic interactions. The superscripts on the b's are for the type of interaction: We distinguish between additive by additive (AA), additive by dominance (AD), dominance by additive (DA) and dominance by dominance (DD) interactions. The x and z are coded variables denoting

the genotype of the QTL. The x_{ir} take on values $(1,\ 0,\ -1)$ for QTL genotypes $(QQ,\ Qq,\ qq)$, while the z_{ir} are $1/2$ for heterozygotes and $-1/2$ for homozygotes.

This results in a vector of genotypic values, one entry per individual in the simulated data set. The genetic variance is the sample variance of this vector of genotypic values. Call it σ_g^2. The environmental variance, σ_e^2 is defined by

$$\sigma_e^2 = \sigma_g^2(\frac{1}{h^2} - 1) \tag{2.9}$$

where h^2 is the heritability of the trait. The extra environmental effect is taken from a normal distribution with mean 0 and variance σ_e^2. If the environmental variance is specified, the heritability is ignored and the environmental variance is used directly. For each individual in the data set, a random variable with mean zero and variance σ_e^2 is generated and added to the genotypic value. This is the phenotypic value of that individual, and is printed in the output file.

2.3.2 Translating Data

Similar to **Rmap** and **Rqtl**, **Rcross** can translate files in a pair of special formats. The first format is the input format for **Mapmaker/QTL**. These would be the **Mapmaker/QTL** "*.raw" files. Simply invoke **Rcross** and specify that the input file is one of these files. The parameters that are for simulations are then ignored. **REMEMBER:** The first two words of a **Mapmaker/QTL** raw file should be "data type". Older versions of **Rcross** cannot process comments at the beginning of a raw file. In fact, it depends on those first two words to recognize the file as a **Mapmaker/QTL** raw file. Beginning with version 1.12, comments will be allowed in the beginning of a "mapmaker.raw" file if you include the "-filetype mapmaker.raw" indicator within the first 100 lines of your file. It is usually best to put this on the first line. **Rcross** will recognize the command and translate the file appropriately. You might want to get into the habit of putting the "-filetype" token (with an appropriate identifier) in your input files, as it will become more important in future releases of ***QTL Cartographer***. There are two other things to keep in mind when using **Mapmaker/QTL** files. The first is that marker and trait names are truncated to eight characters in the output: Versions of ***QTL Cartographer*** prior to 1.12 will be tripped up by this. Secondly, **Mapmaker/EXP** has been known to translate underscores "_" as minus signs "-" in its output, so you might want to avoid them.

Another format is one designed for the ***QTL Cartographer*** system. It is defined in the file **cross.inp** included with the distribution and outlined in Section 6.3.2. Finally, **Rcross** can read files in it's own output format (filetype "Rcross.out") for translation to "mapmaker.raw" or "cross.inp" filetype formats.

Rcross can also read data files formatted for **PLABQTL** (Utz and Melchinger 1996). You need to specify the filetype on the first line of the **PLABQTL** input file. The two filetypes are *plabqtl0.inp* for matrix input and *plabqtl1.inp* for vector input. In addition, if you have data with raw measurements from different environments then you need to add the phrase *-environments* x at the end of the first line, where x is the number of environments.

2.3.3 Output

The flag **-g** can be used to indicate the output format of **Rcross**. In contrast to the input formats, there are seven options for output. **Rcross** will write output in a format suitable for **Mapmaker/QTL** if the **-g** option is used with the integer 2, while a cross.inp formatted file will be written with the value 1. Using a value of 3 produces a file suitable for input into **Splus** or **R**, while a 4 generates a **SAS** program file. The integers 5 and 6 tell **Rcross** to produce files suitable for import into **PLABQTL** (Utz and Melchinger 1996): Use 5 if you want the matrix format and 6 for the vector format.

The default is what we term the qtlcart.cro format, and is indicated by using zero with the **-g** option. Here is an example of the output of **Rcross**.

```
#     1472574604   -filetype Rcross.out
#
# QTL Cartographer V. 1.12c, March, 1997
#
-n        300   is the sample size
-p         63   is one more than the number of markers
-cross   B1   is the type of cross
-traits     1   is the number of traits
-Names of the traits...
  1 Trait.1
-otraits    0   is the number of other traits
#
-s
  1  1
1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1
2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
   7.035406650635
  2  1
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 1 1 1 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1
   3.555115422473
  3  1
2 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 1 1 1
2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1
   4.165996548162
.
.
```

```
.
-e
```

The section prior to the '-s' token is self-explanatory. The area between the '-s' and the '-e' is the data. It starts with an identification number (1, 2, 3, *etc*), and is followed by a "1". At the moment this number "1" is ignored. With the convention that A_1 alleles originated from the P_1 line and A_2 from the P_2, marker genotypes will be encoded with the following integer values:

- 2 for A_1A_1
- 1 for A_1A_2
- 0 for A_2A_2
- 12 for A_1-, that is individuals with at least one dominant A_1 allele
- 10 for A_2-, that is individuals with at least one dominant A_2 allele
- -1 for unknown genotypes. **Rcross** read something but could not translate it.
- -2 is also for unknown genotypes. In this case, no data had been read in.

The trait values follow the marker genotypes, and finally the "other" (categorical or qualitative) traits follow at the end. The sequence repeats for all individuals. Note that there is a permissible range for trait values. By default, all trait values must be real numbers with absolute value less than one million (10^6). Any trait value that is less than negative one million is treated as a missing phenotype by the programs.

Other Traits

Other traits can be thought of as qualitative or categorical traits. Examples include sex, brood, plot, *etc.* In some cases these factors will have been "regressed out", that is a regression of the quantitative trait of interest on the categorical trait will have been performed and the residuals used as the phenotypes in the analysis. Presently, they can be input via a file of filetype "cross.inp" but not automatically analyzed. One includes these other traits in the regression model by prepending a plus sign (+) to the other trait name. For example,

```
-Names of the other traits...
  1 +Sex
  2 -Line
```

would incorporate a Sex effect in the regression model, while ignoring the Line effect.

Interface with R and Splus

If **-g** is used with the integer 3, then **Rcross** will create a file with all the trait, marker and categorical data that is suitable for loading into **Splus** or **R**. In addition, it will create data frames for the different types of data and set up a set of commands to do one-factor ANOVA of each trait on each marker, and each trait on each categorical trait. From there, the user can do higher level ANOVA analyses in **R** or **Splus**. In the output you will see data frames with names *Markers$markername*, *Traits$traitname*, and *Otraits$otraitname*, where *markername* is the name of a specific marker, *traitname* is the name of a specific trait and *otraitname* is the name of a specific categorical trait. If markers have no names, then the data for marker Y on chromosome X will be in *Markers$cXmY*. A similar convention holds for traits and categorical traits: the trait numbered Z will go into *Traits$tZ* and categorical trait W will go into *OTraits$oW* . The output file will have filetype *RSplus.inp*.

Once the *RSplus.inp* file has been created, it can be imported into **Splus** or **R** with the *source* command. Here is an example of using **R** in a UNIX environment using the **mletest.cro** file has ben converted with **Rcross** and the **-g 3** option:

```
prompt % Rcross -X mletest -i mletest.cro -o mletest.r -g 3 -A -V
prompt % R

R : Copyright 2000, The R Development Core Team
Version 1.0.0 Patched (April 4, 2000)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type    "?license" or "?licence" for distribution details.

R is a collaborative project with many contributors.
Type    "?contributors" for a list.

Type    "demo()" for some demos, "help()" for on-line help, or
        "help.start()" for a HTML browser interface to help.
Type    "q()" to quit R.

> source("mletest.r")

.....lots of output that looks like....

Response: Traits$Trait.1
                    Df  Sum Sq Mean Sq F value Pr(>F)
Markers$Marker4.16   1    0.34    0.34  0.0978 0.7547
Residuals          298 1038.21    3.48
> q()
prompt %
```

The ANOVA of the trait on marker 16 from chromosome 4 has a p-value of 0.7547, indicating little evidence of linkage with a QTL.

Since the output is going into **R**, you need to take care in naming your markers, traits and categorical traits. Special characters like underscores and dollar signs mean something to **R** and should be avoided.

Interface with SAS

Similar to the **R/Splus** output option, you may use **-g 4** to get a conversion of your data to **SAS** format. The file will contain all the data and a set of **PROC ANOVA** statements for each trait on each marker and categorical trait. Note that there is a **RUN** statement at the very end of the output file, so that if you load it into **SAS**, then all the ANOVA's will be done automatically. You will need to take care with marker, trait and categorical trait names. See the **SAS** documentation for limitations.

2.4 Prune

Prune takes a genetic linkage map and a data set as input. The user can either eliminate some of the data (markers or traits), bootstrap, permute or simulate missing data. Table 2.5 summarizes the command line options for Prune.

Originally, **Prune** was strictly a command line program. In adding the interactive menu it became necessary to add a second level of interaction. When **Prune** is invoked in the interactive mode, the user will see a menu in which all the parameters of Table 2.5 can be set. The user will then proceed to another interactive menu in which data manipulation can be performed. The second menu will list actions that can be taken. The user selects an action and provides the proper values at which time the action is taken. This second menu is in a loop. The user can continue to take actions until the option to quit is chosen. At the end, the data set is printed out. A few actions can be done automatically. They are bootstrapping, permuting and simulating missing data. These are provided so that **Prune** can be run in a batch file for permutation tests or bootstrap experiments.

The output files of **Prune** may include a genetic linkage map and a data file. If markers had been eliminated, then the linkage map is regenerated to take this into account. The new output files will have the extensions .mpb and .crb, and filename stems specified by the **-o** option.

2.4.1 Pruning Datasets Interactively

The pruning of datasets occurs in an interactive menu. After setting parameters in the first menu, continue on to the second interactive menu where actions can be taken. The second interactive menu looks like this:

```
You can loop through items 1-6, but 7-11 terminate.


 No.                Action
```

Option	Default	Explanation
-o	qtlcart	Output Filename Stem
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-i	qtlcart.cro	Data File
-s	860436420	Random Number Seed
-I	1	Interactive mode (0,1)⇒(no,yes)
-b	0	B (1), P (2), M (3) or D (4)
-M	0.100000	Percent missing data to simulate

Table 2.5: Command Line Options for Prune

```
 1. Eliminate A- marker systems (P1 dominant)
 2. Eliminate a- marker systems (P2 dominant)
 3. Eliminate marker m on chromosome c
 4. Eliminate trait t
 5. Eliminate individuals with missing phenotypes for trait t
 6. Eliminate individuals with more than m% missing markers
 7. Bootstrap the data
 8. Permute the traits in the data
 9. Simulate m% missing markers
10. Simulate m% dominant markers
11. Simulate m% selective genotyping
12. Write modified dataset and exit
13. Exit without writing anything

   Pick a number to do an action...
```

Dominant Markers

There are actions to eliminate dominant markers from the data set. These options were included at a time when **Zmapqtl** and **LRmapqtl** couldn't analyze dominant markers. With the addition of subroutines to analyze dominant markers, the need for these options has lessened. Selecting option 1 or 2 in the second interactive menu eliminates one dominant markers or one type or the other.

Eliminating markers and traits

Option 3 of the interactive menu has an action to eliminate a specific marker. You should be aware that the order of elimination is important. If all the markers to be eliminated are on separate chromosomes, the order is unimportant. If two markers from the same chromosome are to be eliminated, higher numbered marker should be eliminated first. The same concept holds for traits with option 4: Eliminate them in the order of highest to lowest. You will need

to know the marker number and chromosome number rather than the marker name to use this option.

Culling sparse data

Some markers or traits may have been typed for a small proportion of individuals in the dataset. Such markers or traits can be eliminated from the data set. Option 5 will allow you to specify a trait number, and then eliminate individuals with missing data for that trait. Choosing option 6 will require a tolerance level for the percentage of missing marker data. The **-M** option specifies this number, which must be in the range $(0.0, 100.0)$. If option 6 is selected, **Prune** will eliminate individuals with this percentage of missing marker data.

Resampling data

Options 7, 8 and 9 allow the user to bootstrap, permute or simulate missing data for the dataset. If a bootstrap is chosen, then a new dataset of the same size will be resampled (with replacement) from the original data. A permutation simply permutes the trait values. The simulation of missing data requires a percentage level to simulate: That percentage of markers will then be set to unknown. These options are examined in more detail in Section 2.4.2.

Selecting 7, 8 or 9 will do the requested action, write the output and exit. The other options require you to specify when to write and exit. You also have the option of exiting without writing anything.

2.4.2 Recreating Datasets

Command Line Actions

When you use the **-b** option with an integer value from one to six, **Prune** will take one action, print out a new dataset and exit. There are six possible actions, and some require that you use the **-M** option with a percentage as well. The six possible actions are listed below: use the number preceeding the action with the **-b** option to do that action.

1. Bootstrap the data. This option samples individuals with replacement to produce a new data set of the same size as the original.

2. Permute the data. This option will randomly shuffle the traits against the genotypic arrays. If there are multiple traits, then there is a shuffling for each trait. If you want to keep trait arrays together, see item six below.

3. Simulate missing markers. This option requires you to specify a percent $(0 - 100\%)$ of missing marker information. **Prune** will then randomly reset that percentage of marker data to unknown.

4. Simulate dominant markers. Again, you need to specify the percentage of dominant marker systems. A random set of markers are chosen and converted to dominant markers. The direction is random (50/50 each way) for each marker chosen.

5. Simulate selective genotyping. The **-M** option specifies what proportion of individuals that are typed. Do this with single trait data sets. The individuals are ordered according to their trait values, and one-half the specified proportion in the tails are retained in the output. The individuals in the middle of the distribution are deleted from the data set.

6. Permute the data. This is the same as option 2 above except that with multi-trait data sets, there is a shuffling of the trait arrays against the genotype arrays. If you think the traits are correlated, you can use this to maintain that correlation. A value of 12 is the same sas this option.

Bootstrapping

The **-b** option with a value of 1 tells **Prune** to create a single bootstrapped data set. This option should be used alone. It will sample the data set with replacement, creating a new data set of the same sample size and writing it to the file **qtlcart.crb**. Of course, you can change the output file name by changing the output filename stem with the **-o** option. Using **Prune**, one can perform a bootstrap experiment on the data set. This is much easier to do on a UNIX workstation than a Macintosh or MS-Windows machine because it can be automated in a batch file.

Suppose qtlcart is the filename stem, and use m as the integer code for the model, h for the hypothesis test, and r for the number of repetitions, then this snippet of psuedo-code will do a bootstrap analysis

```
Zmapqtl -A -V -X qtlcart -M m
SSupdate.pl -I h < qtlcart.z > qtlcart.z.boot
move qtlcart.z qtlcart.z.save
i = 1
while (i < r) {
        Prune -A -V -i qtlcart.cro -b 1
        Zmapqtl -A -V -M m -i qtlcart.crb
        SSupdate.pl -I h -f qtlcart.z.boot < qtlcart.z > qtlcart.z.new
        move qtlcart.z.new qtlcart.z.boot
        delete qtlcart.z
        i = i + 1
}
SSupdate.pl -I h -c -f qtlcart.z.boot > qtlcart.z.booted
move qtlcart.z.save qtlcart.z
```

Except for the *while* loop line, he symbols $<, >, >>$ above are for input and output redirection. **SSupdate.pl** is a **Perl** script available with the distribution of the programs (look in **doc/scripts** subdirectory). Note that the work is done in the *while* loop. For each repetition, a bootstrapped data set is created with **Prune**. This data will be placed in the file ending with *.crb*. **Zmapqtl** then analyzes this bootstrapped data. The script **SSupdate.pl** reads the results of the **Zmapqtl** run and updates a file with the sum and sum of squares of the test statistic and estimates of effects. Also, **SSupdate.pl** is run to initialize the boot file, so you will need to run **Zmapqtl** on the original data before doing the bootstrap. When this is finished, the script runs **SSupdate.pl** with the **-c** option to get the mean and variance of the

likelihood ratio, additive effect and dominance effect at each test site. There is a **C** shell script called **Bootstrap** in the **scripts** subdirectory of the distribution that implements the above idea.

Permutation Tests

Zmapqtl can perform permutation tests using interval mapping, but if you want to do a proper permutation test using composite interval mapping and reselecting your background markers during each permutation, you will need to do it in a batch file similar to the one for bootstrapping. **Prune** can create a single permuted dataset by using the **-b** option with a value of 2 or 6. If you use 2, then each trait is permuted against the genotypes. Alternatively, using a 6 means that the traits as a block are permuted against the genotypes to retain any correlation the traits may have.

Suppose qtlcart is the filename stem, and use m as the integer code for the model, c for the **Zmapqtl** output file column to test, and r for the number of repetitions, then this snippet of psuedo-code will do a permutation test

```
Zmapqtl -A -V -X qtlcart -M m
GetMaxLR.pl -i < qtlcart.z > qtlcart.z.ewt
CWTupdate.pl -C c < qtlcart.z > qtlcart.z.cwt
move qtlcart.z qtlcart.z.save
i = 1
while (i < r) {
        Prune -A -V -i qtlcart.cro -b 2
        Zmapqtl -A -V -M m -i qtlcart.crb
        GetMaxLR.pl -r i -C c < qtlcart.z >> qtlcart.z.ewt
        CWTupdate.pl -f qtlcart.z.cwt -C c < qtlcart.z >> qtlcart.z.new
        move qtlcart.z.new qtlcart.z.cwt
        delete qtlcart.z
        i = i + 1
}
EWThreshold.pl < qtlcart.z.ewt
move qtlcart.z.save qtlcart.z
```

The **Perl** scripts **GetMaxLR.pl**, **EWThreshold.pl** and **CWTupdate.pl** are in the **doc/scripts** subdirectory. A **C** shell example script, **Permute**, also resides there.

You could also have **SRmapqtl** redo the stepwise regression in the above script so that the background markers in composite interval mapping reflect the permuted data set rather than the original. There is a small quirk in this type of simulation if you are using **SRmapqtl** with stepwise forward-backward regression and **Zmapqtl** with model 6. Sometimes a permuted data set will result in no markers being sufficiently correlated with the trait of interest to be added in the forward phase of the stepwise regression. Thus, **Zmapqtl** will think there are no markers to be used as covariates, and default to interval mapping. Thus, you may not get the exact number of permutations specified to the above script.

Simulating Missing Data

You can also use **Prune** to simulate missing data. You set the amount of missing marker data you would like to simulate with the **-M** option. This will be a percent, and should be specified before you invoke the bootstrap option, which actually does the simulation. Use a value of 3 to tell **Prune** to randomly set some of the markers to missing. Over the entire data set, approximately the percentage of markers that had been set with the **-M** option will be set to -10. The results will be in a file with the filename extension ".crb". Similar to simulating missing data, some of the markers can be made dominant by using a value of 4 with the bootstrap option. The percentage of markers transformed is set with the *-M*. The direction of dominance is random: Half of those changed will convert the P_1 allele to dominant, while the other half will convert the P_2 allele.

If you use a value of 5 with **-b** and specify a percentage for **-M**, then you can investigate how selective genotyping compares to having typed all the individuals. As an example, suppose you have a data set typed for 500 individuals and use **-M 20** and **-b 5.** The individuals are ordered with respect to the trait of interest and those whose trait values are in the lowest 10% are retained along with those in the highest 10%. Those in the 10 to 90 percent range are deleted. A new data set with the ".crb" filename extension will contain the results.

2.5 Emap

Emap is a module to infer a genetic linkage map from a data set. Initially, it checks each marker for segregation distortion and allows the user to discard distorted markers by specifying the size of the statistical test. **Emap** then sorts and order the remaining markers into linkage groups using rapid chain delineation.

Table 2.6 presents the options available to **Emap**. You are required to supply a data input file, either with the **-i** command line option or from the menu. This file can be in any format that **Rcross** can translate. You do not need to specify a map input file: This will be inferred. You can specify a map input file if you simply want to refine the order of markers on linkage groups.

The map and the data will be printed in the standard ***QTL Cartographer***formats (Rmap.out and Rcross.out) to the files specified with the **-l** and **-o** options, respectively.

The **-S** and **-L** options allow the user to set the size of the statistical tests for segregation (**-S**) and linkage (**-L**). The default values are 0.01 and 0.25, respectively. You can disable the test for segregation distortion by setting the size of that test to 0.0.

Section 2.1 explains the map function and map function parameters: These are set with the **-f** and **-p** options.

The linkage map method is set with the **-M** option and can take on values 10, 11, 12, or 13. The value 10 generates an initial genetic linkage map. Methods 11, 12 and 13 use that initial map, but do various permutations of the markers in linkage groups to refine the order of markers.

2.5.1 Objective functions

Emap can use one of two functions to compare different map orders. One is to minimize the sum of adjacent recombination estimates (SAR), and the other is to maximize the sum of the likelihoods (SAL) for these estimates. Use the **-O** option with a value of 0 for SAL and 1 for SAR.

2.5.2 Rapid Chain Delineation

Doerge and Weir have developed the rapid chain delineation method to infer a genetic linkage map. The method requires pairwise recombination fractions between all markers. The first linkage group is initiated with the pair of markers having the smallest recombination fraction. The remaining markets are placed in a "pool" awaiting placement on the map. The linkage group is extended by adding markers from the pool of unlinked markers. Each terminal marker of the linkage group is a candidate for extension of the chain: The unlinked marker that has the smallest recombination fraction with either is added to the chain subject to the provision that the recombination fraction is statistically significant at a prespecified level (see option **-L** above). This process is repeated as long as markers can be added to the chain. When no markers can be added, then a new linkage group is created from the pool of unlinked markers. Some markers may not show any linkage to any other markers: They are discarded.

Option	Default	Explanation
-i		Data Input File
-o	qtlcart.cro	Data Output File
-e	qtlcart.log	Error File
-m		Map Input File
-l	qtlcart.map	Map Output File
-s	1039791519	Random Number Seed
-M	10	Linkage Map method
-S	0.01	Segregation test size
-L	0.25	Linkage test size
-r	0	Permutations: not active
-f	1	Map Function [1,8], 1 => Haldane
-p	0.0	Map Function Parameter
-O	0	Objective Functions (0=>SAL, 1=>SAR)

Table 2.6: Command Line Options for Emap

2.5.3 Estimating Recombination

We use maximum likelihood estimators for recombination fractions between pairs of loci. For loci A and B with alleles A, a and B, b, respectively, we observe nine genotypes:

$$AABB,\ AABb,\ AAbb,\ AaBB,\ AaBb,\ Aabb,\ aaBB,\ aaBb,\ aabb$$

The double heterozygote ($AaBb$) will be in coupling in the F_1 generation, but can not be assumed so in other generations. For ease of presentation, we number the above genotypes from one to nine and define the observed number of that genotype by n_i and the expected proportion p_i as shown in Table 2.7. For example, the observed number of $AABB$ individuals is n_1 and the expected proportion is p_1. The p_i are conditional probabilities that depend on the recombination fraction and the cross that produced the data.

Class	Genotype	Obs. number	Exp. Proportion
1	$AABB$	n_1	p_1
2	$AABb$	n_2	p_2
3	$AAbb$	n_3	p_3
4	$AaBB$	n_4	p_4
5	$AaBb$	n_5	p_5
6	$Aabb$	n_6	p_6
7	$aaBB$	n_7	p_7
8	$aaBb$	n_8	p_8
9	$aabb$	n_9	p_9

Table 2.7: Observed and expected proportions of genotypes

Backcrosses and Recombinant Inbreds

We will not observe all of these classes in backcrosses and recombinant inbred lines. Those that we do observe will have the proportions given in Table 2.8.

Class	B_1	B_2	RI_0	RI_1	RI_2
1	$\frac{1}{2}(1-r)$	0	$\frac{1}{2}(1-r)$	$\frac{1}{2+4r}$	$\frac{1+2r}{2+12r}$
2	$\frac{1}{2}r$	0	0	0	0
3	0	0	$\frac{1}{2}r$	$\frac{r}{2+4r}$	$\frac{2r}{2+12r}$
4	$\frac{1}{2}r$	0	0	0	0
5	$\frac{1}{2}(1-r)$	$\frac{1}{2}(1-r)$	0	0	0
6	0	$\frac{1}{2}r$	0	0	0
7	0	0	$\frac{1}{2}r$	$\frac{r}{2+4r}$	$\frac{2r}{2+12r}$
8	0	$\frac{1}{2}r$	0	0	0
9	0	$\frac{1}{2}(1-r)$	$\frac{1}{2}(1-r)$	$\frac{1}{2+4r}$	$\frac{1+2r}{2+12r}$

Table 2.8: Backcross and Recombinant inbred proportions

For a backcross to parental line 1, there are two non-recombinant classes (1, 5) with proportions $\frac{1-r}{2}$ and two recombinant classes (2, 4) with proportions $\frac{r}{2}$. The likelihood equation for r is

$$L(r) = (\frac{1-r}{2})^{(n_1+n_5)}(\frac{r}{2})^{(n_2+n_4)}$$

and the maximum likelihood estimator for r is

$$\tilde{r} = \frac{n_2 + n_4}{n_1 + n_2 + n_4 + n_5}$$

For backcrosses to parental line 2, non-recombinant classes are 5 and 9, while the recombinant classes are 6 and 8. The corresponding likelihood equation is

$$L(r) = (\frac{1-r}{2})^{(n_5+n_9)}(\frac{r}{2})^{(n_6+n_8)}$$

and the maximum likelihood estimator is

$$\tilde{r} = \frac{n_6 + n_8}{n_5 + n_6 + n_8 + n_9}$$

What about advanced backcrosses?

Ri_0 lines are also known as doubled haploids. The likelihood equation for r is

$$L(r) = (\frac{1-r}{2})^{(n_1+n_9)}(\frac{r}{2})^{(n_3+n_7)}$$

and the maximum likelihood estimator is

$$\tilde{r} = \frac{n_3 + n_7}{n_1 + n_3 + n_7 + n_9}$$

Recombinant inbred lines generated via selfing are coded as Ri_1: The likelihood equation is

$$L(r) = (\frac{1}{2+4r})^{(n_1+n_9)}(\frac{r}{1+2r})^{(n_3+n_7)}$$

and we estimate r by

$$\tilde{r} = \frac{n_3 + n_7}{2(n_1 + n_9)}$$

For sib-mated recombinant inbred lines (Ri_2) we have

$$L(r) = (\frac{1+2r}{2+12r})^{(n_1+n_9)}(\frac{2r}{1+6r})^{(n_3+n_7)}$$

and estimate r with

$$\tilde{r} = \frac{n_3 + n_7}{4(n_1 + n_9) - 2(n_3 + n_4)}$$

Intercrosses

For codominant markers in the F_2 generation, Table 2.9 presents the expected proportions of the observable genotypes

When markers are dominant, certain genotypic classes are indistinguishable from others. For example, if B is dominant to b, then six classes result as shown in Table 2.10.

Finally, for a pair of dominant markers we can distinguish four genotypic classes. Suppose A, B are the dominant alleles, then Table 2.11 results

In all cases, the likelihood is

$$L(r) = \prod_i (p_i)^{n_i}$$

Since $0 \leq r \leq 0.5$, we can numerically solve for r via iteration.

Class	Genotype	Obs. number	Exp. Proportion, p_i
1	$AABB$	n_1	$\frac{1}{4}(1-r)^2$
2	$AABb$	n_2	$\frac{1}{2}(r-r^2)$
3	$AAbb$	n_3	$\frac{1}{4}r^2$
4	$AaBB$	n_4	$\frac{1}{2}(r-r^2)$
5	$AaBb$	n_5	$\frac{1}{2}(1-2r+2r^2)$
6	$Aabb$	n_6	$\frac{1}{2}(r-r^2)$
7	$aaBB$	n_7	$\frac{1}{4}r^2$
8	$aaBb$	n_8	$\frac{1}{2}(r-r^2)$
9	$aabb$	n_9	$\frac{1}{4}(1-r)^2$

Table 2.9: F_2 expected proportions

Class	Genotype	Obs. number	Exp. Proportion, p_i
1	$AAB-$	n_1	$\frac{1}{4}(1-r^2)$
2	$AAbb$	n_2	$\frac{1}{4}r^2$
3	$AaB-$	n_3	$\frac{1}{2}(1-r+r^2)$
4	$Aabb$	n_4	$\frac{1}{2}(r-r^2)$
5	$aaB-$	n_5	$\frac{1}{4}(2r-r^2)$
6	$aabb$	n_6	$\frac{1}{4}(1-r)^2$

Table 2.10: F_2 proportions with one dominant locus

Class	Genotype	Obs. number	Exp. Proportion, p_i
1	$A-B-$	n_1	$\frac{1}{4}(3-2r+r^2)$
2	$A-Abb$	n_2	$\frac{1}{4}(2r-r^2)$
3	$aaB-$	n_3	$\frac{1}{4}(2r-r^2)$
4	$aabb$	n_4	$\frac{1}{4}(1-r)^2$

Table 2.11: F_2 proportions with two dominant loci

2.5.4 Map Refinement

Emap uses rapid chain delineation to create an initial genetic linkage map. The user can also specify an initial map and **Emap** can refine the order of the markers within linkage groups. **Emap** can read any file that **Rcross** can translate.

The genetic map can be refined in three stages.

In stage 1, all adjacent pairs of markers are swapped. For each swapping, the objective function is recalculated and the best order is retained. If a new order results, this step is repeated.

Stage 2 is similar except that all adjacent triplets of markers are permuted.

In stage 3, each marker is placed in every other intervals on the same chromosome.

Genome reordering

If you would like to specify certain markers as anchors for chromosomes, centromeres or telomeres, then you can encode those specifications in the marker names. For a marker named "MARKER", you would rename it "Anchor|p|c|MARKER", where the p and c are for position and chromosome. The position (p) can take on three values: l for left telomere, c for centromere and r for right telomere. In general, $p = c$ has no effect. If $p = l$, then **Emap** reorders the markers so that more markers are to the right of the anchor than to the left (to make this anchor toward the left side of the chromosome). A value of $p = r$ does the opposite, attempting to put the anchor toward the right side of the chromosome. **Emap** will not necessary place the marker on the telomere, so you can choose a marker that you think is near the telomere rather than the telomeric marker itself.

The variable c sets the chromosome number for the anchor and all markers in the same linkage group. After markers are reordered and reassigned to chromosomes, the "Anchor|p|c|" prefix is deleted from the marker name for the final output of the map.

Chapter 3

Analysis

A recent review (Doerge, Zeng, and Weir 1997) summarizes the statistical issues for mapping QTLs. It is the best place to start for a general overview of the analytical methods used in ***QTL Cartographer***.

Figure 3.1 shows a schematic of the analysis procedure. There are five programs in this step. **Qstats** does some basic quantitative genetic statistics and summarizes missing data. It is a useful program to run at the beginning of your analysis. **LRmapqtl** does single maker analysis using linear regression. It also runs very fast and will give some idea of where QTLs are. **SRmapqtl** does stepwise regression, either forward, backward or forward with backward elimination. The program **Zmapqtl** implements interval mapping (Lander and Botstein 1989) and composite interval mapping (Zeng 1993; Zeng 1994) for a single trait at a time. **JZmapqtl** is an extension to multitrait mapping (Jiang and Zeng 1995). Multiple interval mapping (Kao and Zeng 1997; Kao, Zeng, and Teasdale 1999; Zeng, Kao, and Basten 1999) is available in **MImapqtl**.

The basic requirements for using these three programs is a genetic linkage map and a data file. The linkage map should be of filetype format "Rmap.out" and the data file of "Rcross.out". Whether the files are simulated, real or bootstrapped data is irrelevant: The analysis is the same regardless of the origin of the data.

3.1 Qstats

Qstats is a good place to start in analyzing your data. It computes some basic statistics on the quantitative traits and summarizes missing data. Let $\{y_1, y_2, \cdots, y_n\}$ be a vector of quantitative trait values. For each trait in turn, it calculates the sample size (n), mean ($\bar{y} = \frac{1}{n}\sum_1^n y_i$), variance ($s^2 = \frac{1}{n-1}\sum_1^n (y_i - \bar{y})^2$), standard deviation ($s = \sqrt{s^2}$), skewness, kurtosis and average deviation, $\frac{1}{n}\sum_{i=1}^n |y_i - \bar{y}|$. The coefficient of variation is the sample standard deviation divided by the sample mean.

Lynch and Walsh (1998) provide a lucid explanation of some of the statistics calculated by **Qstats**. Let the kth sample moment be $M(k) = \frac{1}{n}\sum_{i=1}^n y_i^k$. Clearly, $M(1) = \bar{y}$. Using the

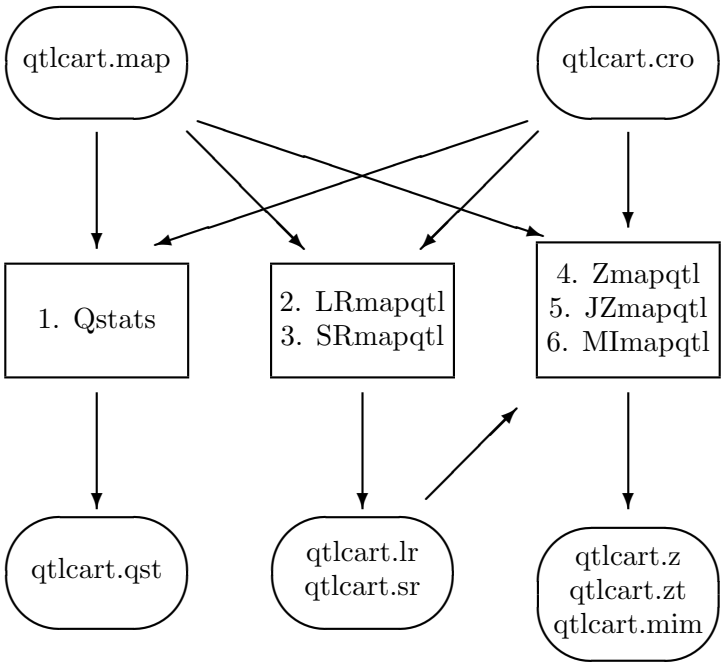

Figure 3.1: Analysis Schematic

notation $\bar{y^k} = M(k)$, we can estimate the sample variance with

$$s^2 = \frac{n}{n-1}(\bar{y^2} - \bar{y}^2) \tag{3.1}$$

An estimate of the skewness is

$$Skw(y) = \frac{n^2}{(n-1)(n-2)}(\bar{y^3} - 3\bar{y^2}\bar{y} + 2\bar{y}^3)$$

The standard error of skewness depends on the underlying distribution but can be approximated by $\sqrt{6/n}$. The coefficient of skewness, k_3 is

$$k_3 = \frac{Skw(y)}{s^3}$$

where the sample standard deviation, $s = \sqrt{s^2}$, is estimated from (3.1). Kurtosis is estimated by

$$Kur(y) = \frac{n^2(n+1)}{(n-1)(n-2)(n-3)}(\bar{y^4} - 4\bar{y^3}\bar{y} + 6\bar{y^2}\bar{y}^2 - 3\bar{y}^4)$$

and the coefficient of kurtosis is

$$k_4 = \frac{Kur(y) - 3s^4}{s^4}$$

Like skew, the standard error of kurtosis is dependent upon the population distribution. We give the estimate $\sqrt{24/n}$. A test of normality for the vector y then involves the test statistic

$$S = \frac{nk_3^2}{6} + \frac{nk_4^2}{24}$$

which is distributed as a χ^2 with two degrees of freedom. The critical values for the rejection of normality are 5.99 and 9.21 for tests at the 5% and 9% levels, respectively.

An example of the output follows:

```
-------------------------------------------------------
-------------------------------------------------------
    This is for -trait 1 called szfreq
-------------------------------------------------------
        Sample Size.............          119
        M(1)....................       0.4349
        M(2)....................       0.2184
        M(3)....................       0.1195
        M(4)....................       0.0694
        Mean Trait Value........       0.4349
        Variance................       0.0295
        Standard Deviation......       0.1718
        Coefficient of Variation...    0.3951
        Average Deviation.......       0.1398
        Skw..LW(24).............      -0.0010
        .....Sqrt(6/n)..........       0.2245
        Kur..LW(29).............       0.0022
        .....Sqrt(24/n).........       0.4491
        k3...LW(24).............      -0.1922
        k4...LW(28).............      -0.5250
        S (5%: 5.99, 1%: 9.21).....    2.0992
-------------------------------------------------------
-------------------------------------------------------
```

In the above example, LW(i) refers to a page number in Lynch and Walsh (1998) where one can find an explanation of the quantity. The value of the test statistic S is 2.0992, thus one would fail to reject the hypothesis that this trait is normally distributed.

After the basic statistics, **Qstats** draws a histogram of the quantitative trait. It is a simple histogram in that the range of the data are divided into 50 equally sized bins, and the number of data points falling into each bin are counted and plotted. A small table following the histogram gives the sample size, minimum, first quartile, median, second quartile and maximum.

3.1.1 Command Line Options

Table 3.1 summarizes the command line options for **Qstats**. There are very few of them. You can specify the data set, genetic linkage map file and output file. In addition, all the global options of Table 1.4 are valid.

Option	Default	Explanation
-i	qtlcart.cro	Data Input File
-o	qtlcart.qst	Output File
-m	qtlcart.map	Genetic Linkage Map File
-p	no	Calc. Marker Probabilities

Table 3.1: Command Line Options for Qstats

Another function of **Qstats** is to summarize the missing data for markers, traits and individuals. Following the histogram, there will be a table. For each trait, it will present a summary of missing data for each marker in turn. The table will consist of seven columns. The first three columns indicate the chromosome, marker number and name of the marker (if there is a marker name). The fourth column specifies what type of marker **Qstats** thinks it is. There are three types that are recognized. The first is codominant and is indicated by a "co" token. The other two are dominant markers and **Qstats** distinguishes between marker systems in which A_1 is dominant to A_2 (indicated by the token "A-") and those in which A_2 is dominant to A_1 ("a-"). Column 5 has the counts of individuals with data for the marker, while column 6 has the counts of individuals with both marker and trait data. Column seven is just the ratio of columns 5 and 6.

At the end of the **Qstats** output file, there will be a summary of missing data for each individual in the data set. **Qstats** will indicate the number of marker systems, quantitative traits and categorical traits. It will then have a table with seven columns. Column 1 is for the individual. Column 2 indicates the number of markers for which the individual is typed, and Column 3 indicates a percent. Columns 4 and 5 do the same for traits while columns 6 and 7 summarize the data for categorical traits.

Something to keep in mind is that some of the analyses require large sample sizes. For example, if the sample sizes are too small, the ECM algorthm may fail in **Zmapqtl**. When difficulties in analysis are encountered, check the missing data summaries in the **Qstats** output: Such problems often correspond to areas with a lot of missing data.

3.1.2 Segregation

Qstats also tests for adherence to Mendelian segregation at all marker loci. For a given locus, suppose there are r genotypic classes. Let p_i be the expected frequency, and n_i the observed count for the ith class. For a sample of size n, the expected counts will be np_i and the observed frequencies will be n_i/n. We can construct a test statistics based on a contigency table

$$T_1 = \sum_{i=1}^{r} \frac{(n_i - np_i)^2}{np_i} = n \sum_{i=1}^{r} \frac{(n_i/n - p_i)^2}{p_i}$$

or a comparison of likelihoods

$$T_2 = -2 \sum_{i=1}^{r} n_i (\ln n_i - \ln np_i)$$

Both T_1 and T_2 should have a χ^2 distribution with one degree of freedom for backcrosses and recombinant inbred lines and two degrees of freedom for intercrosses. Both statistics are calculated and presented in a table in the **Qstats** output.

3.1.3 Marker Probabilities

Marker genotypes can be ambiguous for missing data or dominant markers. **Qstats** can calculate a distribution for the marker genotype (Fisch, Ragot, and Gay 1996; Jiang and Zeng 1997) and output these probabilities for each marker and each individual in the data set. Using the *-p* with a value of *yes* will have **Qstats** produce a table of these values at the end of the output file. The table will also contain expected additive and dominance values that could be used in a regression analysis. An example of the output is

```
-begin markerprobs
-markername BD267                  -chromosome     1 -marker     1
-h   ind.    p(QQ)    p(Qq)     p(qq)    E(a)      E(d)
        1   0.3866   0.6134   0.0000    0.3866    0.1134
        2   0.4150   0.5850   0.0000    0.4150    0.0850
        3   0.4304   0.5696   0.0000    0.4304    0.0696
        4   0.1519   0.8481   0.0000    0.1519    0.3481
        5   0.3131   0.6869   0.0000    0.3131    0.1869
        6   0.3455   0.6545   0.0000    0.3455    0.1545
        7   0.3455   0.6545   0.0000    0.3455    0.1545
        8   0        0        1        -1.0      -0.5
.
.
.
-end markerprobs
```

Here, $p(QQ)$, $p(Qq)$, $p(qq)$ are the probabilities of A_1A_1, A_1A_2 and A_2A_2 genotypes, respectively, for that individual at that marker. These probabilities are non-trivial for missing data or dominant markers. In the above example, individual 8 has the unambiguous genotype A_2A_2 for the marker system that has A_1 dominant to A_2. The first seven individuals have genotype A_1A_1 or A_1A_2. The expected values are

$$E[a] = p(QQ) - p(qq)$$

and

$$E[d] = [p(Qq) - p(QQ) - p(qq)]/2.$$

$E[a]$ is calculated assuming marker values of 1, 0 and -1 for A_1A_1, A_1A_2 and A_2A_2 genotypes, respectively. For $E[d]$, heterozygotes have value 0.5 while homozygotes are set to -0.5.

3.2 LRmapqtl

LRmapqtl fits the data to a simple linear regression model. For each marker in turn, it fits a simple linear model to the trait data. It is a quick way to get an idea of where the QTLs may reside.

3.2.1 Simple Linear Regression

For each marker in turn, **LRmapqtl** fits the phenotypic data to the linear model

$$y_i = b_0 + b_1 x_i + e \tag{3.2}$$

where y_i is the phenotype of the ith individual and x_i is an indicator variable for the marker genotype. Generally,

$$x_i = \begin{cases} 2 & \text{if} \quad A_1A_1 \\ 1 & \text{if} \quad A_1A_2 \\ 0 & \text{if} \quad A_2A_2 \end{cases}$$

but for B_1 crosses

$$x_i = \begin{cases} 1 & \text{if} \quad A_1A_1 \\ 0 & \text{if} \quad A_1A_2 \end{cases}$$

If the marker is missing or dominant, then an expected value for the marker is calculated from the flanking markers (Fisch, Ragot, and Gay 1996; Jiang and Zeng 1997). The regression parameters b_0 and b_1 can be estimated, and e is assumed to have a normal distribution.

LRmapqtl can also take into account categorical traits, that is other variables such as sex or brood, in its analysis. If your data set contains such information, then there should be a list of the names of these other variables near the beginning of the "Rcross.out" formatted file. These names might look as follows:

```
-Names of the other traits...
  1 Sex
  2 Line
```

If you would like to include "Sex" and "Sex by Marker interaction" terms in your analysis, then you need to indicate as much to **LRmapqtl**. If you prefix the name of one of these variables with a plus sign (+), then it will be incorporated into the linear model.

```
-Names of the other traits...
  1 +Sex
  2 Line
```

In **LRmapqtl**, this would consider both Sex and Sex by Marker interaction terms. In **Zmapqtl** and **SRmapqtl**, the Sex by Marker term wouldn't be incorporated, but the Sex factor would. All other variables that have no + sign at the beginning of their names will be ignored in the analysis. For the above example, a pair of models will be considered:

$$y_i = b_0 + b_1 x_i + b_2 Sex + b_3 Sex \times x_i + e \tag{3.3}$$

$$y_i = b_0 + b_2 Sex + e \tag{3.4}$$

Option	Default	Explanation
-i	qtlcart.cro	Data Input File
-o	qtlcart.lr	Output File
-m	qtlcart.map	Genetic Linkage Map File
-r	0	Number of permutations
-t	1	Trait to analyze

Table 3.2: Command Line Options for LRmapqtl

The output will give probabilities that the marker is significant.

Table 3.2 shows the command line options specific to **LRmapqtl**. As with **Qstats**, there are few parameters to change. The *-t* option allows you to specify a trait to analyze. It is trait 1 by default. If you only have one trait, you can ignore this option. If your data set has more than one trait, you can analyze a specific trait by using *-t* with an integer from 1 to the number of traits. If you want **LRmapqtl** to analyze all traits, use a value greater than the number of traits.

3.2.2 Output

LRmapqtl prints out a histogram of the trait (identical to the one from **Qstats**), and the results of simple linear regression. The results are displayed in a table with seven columns. The first column indicates the chromosome, while the second gives the number of the marker on the chromosome. The name of the marker can be found in the genetic linkage map file. The next two columns correspond to the parameters in the linear model (Equation 3.2). Column three is the intercept and column four the slope of the least squares regression line fit to the data. Column five is a likelihood ratio test statistic for the model, and column six is the F statistic. Column seven is the tail probability of the F statistic assuming one and $n-1$ degrees of freedom in the numerator and denominator, respectively. Asterisks attached to these probabilities indicate significance of the F statistics: Significance at the 5%, 1%, 0.1% and 0.01% levels are indicated by one, two, three and four asterisks, respectively.

The results of running **LRmapqtl** are used in **Zmapqtl** for analysis models four and five (see Section 3.4.2).

3.2.3 Permutation Tests

The *-r* option tells **LRmapqtl** to perform a permutation test (Churchill and Doerge 1994). The argument to *-r* indicates how many permutations should be performed. In each permutation, the phenotypes are shuffled relative to the genotypes over individuals and the analysis is redone. The results are summarized at the end of the **LRmapqtl** output file.

3.3 SRmapqtl

SRmapqtl uses the technique of stepwise regression to search for QTLs. For forward and backward regression, it simply ranks the markers for their effect on the quantitative trait. In forward stepwise regression (FS), each marker in turn is tested for its effect on the quantitative trait using linear regression. That marker with the largest partial F-statistic is assigned rank 1 and included in all subsequent analyses. Step two tests all the remaining markers, and assigns rank 2 to the marker with the largest partial F-statistic. This is repeated until all the markers have been ranked.

Option	Default	Explanation
-i	qtlcart.cro	Input File
-o	qtlcart.sr	Output File
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-s	860437285	Random Number Seed
-M	0	FS, BE or FB (0,1,2)?
-t	1	Trait to analyze
-F	0.1	Size: p(Fin) =
-B	0.1	Size: p(Fout) =
-u	100	Maximum number of steps

Table 3.3: Command Line Options for SRmapqtl

Backward elimination regression (BE) starts with all markers in the model. In the first step, each marker in turn is removed and a partial F-statistic is calculated. That marker with the smallest partial F statistic is given the lowest rank and removed from subsequent analyses. This is repeated until all the markers have been ranked.

The above methods seek only to rank the markers: They make no effort to determine whether adding or deleting a marker makes a significant difference for the fit of the model to the data. A third method (FB) is to start with forward stepwise regression, but only keep adding markers while the p-value of the partial F statistic of the marker to be added is below a defined threshold, $p(F_{in})$. When a step is reached in which no more markers can be added, all of the markers are retested to see if they are still significant. Each marker in turn is deleted from the model, a p-value is calculated for the partial F-statistic, and if the p-value is greater than a specified level $p(F_{out})$, it is deleted.

You can put a hard limit on the number of steps in the regression analysis with the -*u* option. The program has internal limits based on the sample size (it won't allow more parameters than sample points), but you can lower this further. It defaults to 100, which is generally more markers than you will need to rank. If **SRmapqtl** seems to run forever, you might run it with this parameter set to a reasonable number, say 20.

As with **LRmapqtl**, any otraits that begin with a plus sign are also used in the regression model. Unlike **LRmapqtl**, no interaction terms are used. The command line parameters for

SRmapqtl are listed in Table 3.3. One added feature is that if you use the *-t* option with an integer value one greater than the number of traits, then all traits will be analyzed in turn.

3.3.1 Output

For the specified trait, **SRmapqtl** will output a small table:

```
-------------------------------------------------------
-------------------------------------------------------
Chromosome    Marker      Rank         F-Stat      DOF
-------------------------------------------------------  -start
         1         1         2       13.38778       114
         2         3         4       10.12742       110
         3         1         5        3.55528       108
         3         2         3       11.15490       112
         4         3         1       28.85236       116  -end
-------------------------------------------------------
-------------------------------------------------------
```

The first two columns indicate the chromosome and marker. The third column gives the rank of that marker as determined by the stepwise regression mode of choice. Then there will be an F-statistic indicating the difference between having that variable in the model or not. Finally, the DOF (degrees of freedom) for the numerator of that F statistic is given. For forward stepwise or backward elimination, **SRmapqtl** will try to rank all of the markers no matter how small the F statistic is. For the forward regression with backward elimination, the program proceeds to add variables until the F statistic p-value is less than that specified by the *-F* option (0.1 by default). Then **SRmapqtl** rechecks all the variables added and will eliminate any with an F statistic p-value less than the value given with the *-B* option.

In general, the FB method is probably the best method for picking background markers to be used with model 6 in **Zmapqtl** and **JZmapqtl**. To this end, **SRmapqtl** should be run prior to using either module. **Zmapqtl** and **JZmapqtl** will read the results of **SRmapqtl** and use the markers that are ranked. You can specify an upper bound to the number of background parameters to be used in **Zmapqtl**. **JZmapqtl** will use all the markers that are listed for all traits in its analysis: The FB method thus selects only a subset of significant markers.

Be aware that **SRmapqtl** tries to determine how many markers can be analyzed at once. The number of parameters has to be smaller than the sample size. If you try to use backward regression, and there are more markers than individuals, then **SRmapqtl** will default to forward stepwise regression and rank as many markers as possible. You should be aware that when dominance can be estimated, each marker will count two towards the total number of parameters and you will need a sample size of at least twice the number of markers to do backward elimination.

3.4 Zmapqtl

Zmapqtl implements interval and composite interval mapping. There are also options to perform a permutation test (Churchill and Doerge 1994; Doerge and Churchill 1996).

3.4.1 Computational Methodology

Composite interval mapping (Zeng 1993; Zeng 1994) combines interval mapping with multiple regression. The statistical model is defined as

$$\mathbf{Y} = \mathbf{x}^* b^* + \mathbf{z}^* d^* + \mathbf{XB} + \mathbf{E} \tag{3.5}$$

where

- $\mathbf{Y}$ is a vector of trait values
- b^* and d^* are the additive and dominance effects of the putative QTL being tested
- $\mathbf{x}^*$ and $\mathbf{z}^*$ are indicator variable vectors specifying the probabilities of an individual being in different genotypes for the putative QTL constructed by flanking makers
- $\mathbf{B}$ is the vector of effects of other selected markers fitted in the model
- $\mathbf{X}$ is the marker information matrix for those selected markers
- $\mathbf{E}$ is the error vector.

Estimates of the parameters are obtained by maximum likelihood through an ECM (for Expectation/Conditional Maximization) algorithm (Meng and Rubin 1993). In each E-step, the probability of an individual being in different genotypes of the putative QTL is updated. In the CM-step, the estimation of parameters b^* and d^* is separated from that of $\mathbf{B}$, and each group is estimated conditional on the others. This procedure is implemented for numerical consideration. As $\mathbf{x}^*$ and $\mathbf{z}^*$ are separated from $\mathbf{X}$, $\mathbf{X}$ is unchanged in each iteration, and its costly recalculation is avoided.

For an F_2 population, the hypotheses for testing are $H_0 : b^* = 0$ and $d^* = 0$ and $H_3 : b^* \neq 0$ or $d^* \neq 0$. This is performed through a likelihood ratio test procedure. In addition, it is possible to test hypotheses on b^* and d^* individually. For a backcross data set, dominance cannot be estimated and d^* is dropped from Equation 3.5.

The trait will have a variance s^2. Under the null hypothesis

$$H_0 : \mathbf{Y} = \mathbf{XB} + \mathbf{E}$$

the sample variance of the residuals will be s_0^2. For a given alternative model, say

$$H_1 : \mathbf{Y} = \mathbf{x}^* b^* + \mathbf{z}^* d^* + \mathbf{XB} + \mathbf{E}$$

the variance of the residuals would be s_1^2. With this in mind we can calculate the proportion of variance explained by a QTL at the test site. The quantity is usually called r^2 and estimated by

$$r^2 = \frac{s_0^2 - s_1^2}{s^2}$$

An alternative estimate would use the total variance. Denote it by

$$r_t^2 = \frac{s^2 - s_1^2}{s^2}$$

r^2 is the proportion of the variance explained by the QTL conditioned on the background markers and any explanatory variables. r_t^2 is the proportion of the total variance explained by the QTL and the the background markers and any explanatory variables. Generally, $r_t^2 \geq r^2$

3.4.2 Models

When we speak of models for analysis, we mean to specify the markers used as cofactors in composite interval mapping. There are presently six models for analysis.

1. Use all the markers to control for the genetic background. This is model 1 from Zeng (1994).

2. Use all unlinked markers to control for the genetic background. This is model 2 from Zeng (1994).

3. Don't use any markers to control for the genetic background. This is also known as interval mapping and is the same as Lander and Botstein's method (Lander and Botstein 1989).

4. This is an ad-hoc model. One marker from each chromosome (except for the chromosome on which we are testing) is used to control for the genetic background. The results of **LRmapqtl** are scanned, and the marker that showed the highest test statistic from each chromosome is used.

5. This is another ad-hoc model. Two markers from each chromosome are used to control for the genetic background. They are the top two markers as determined by **LRmapqtl**. In addition, all the other markers on the chromosome of the test position that are more than 10 cM away from the flanking markers are also thrown in. It may be ad-hoc, but tends to work best at this time. The value of 10 centimorgans can be changed with the *-w* option.

6. Model six will be explained in the next subsection.

7. Model seven requires the results of a prior run of **Zmapqtl** and **Eqtl**. Initially, the user may want to run **Zmapqtl** with interval mapping, summarize the positions and effects of that analysis using **Eqtl**, and then use those estimates as the covariates in the regression model. Virtual markers are created at the best estimates for the positions of the QTLs.

8. Model eight is similar to model seven except that instead of using virtual markers, the nearest flanking markers to the putative QTL are used as cofactors. As in model seven, **Eqtl** is run after **Zmapqtl**, and a table of markers with arbitrary ranks are written to the **SRmapqtl.out** file. You should set the number of background parameters to more than the number of QTL identified by **Eqtl**.

Zmapqtl Model Six

Model 6 requires two additional parameters. One is the number of markers to control for the genetic background (n_p), and the other is a window size (w_s). When invoked, the program will read in the results of a prior run of **SRmapqtl** to pick the most important markers to control for the genetic background. Then, when testing at any point on the genome, it will use up to n_p of these markers. If **SRmapqtl** didn't rank as many markers as specified with n_p, then n_p is reset to the number of markers ranked. The window size will block out a region of the genome on either side of the markers flanking the test site. Since these flanking regions are tightly linked to the testing site, if we were to use them as background markers we would then be eliminating the signal from the test site itself.

Note that if $w_s = 0.0$ and n_p equals the total number of markers, then Model 6 reduces to Model 1. If w_s is large (say the size of the largest chromosome) and n_p equals the number of markers, then Model 2 is the result. If n_p is zero, then Model 3 is the result. In the future, we will recommend that people use model 3 or model 6 for analysis. The default values of 5 for n_p and 10 for w_s should be good starting points for Model 6. Increasing n_p will allow better resolution for mapping linked QTLs.

3.4.3 Zmapqtl Options

Table 3.4 shows the command line options specific to **Zmapqtl**. One can select a trait to analyze, a model for analysis and a walking speed along the genome (that is, the interval between successive analysis points). The user can analyze just one chromosome or the entire genome. Finally, permutation tests or bootstraps can be performed by setting the number of permutations or bootstraps to a number greater than 0. Explanatory variables such as Sex or Line are automatically included in the analysis if their names are preceded by a plus sign in the data file. This is similar to **LRmapqtl**, except that interaction terms are not yet used.

Option	Default	Explanation
-i	qtlcart.cro	Input File
-o	qtlcart.z	Output File
-m	qtlcart.map	Genetic Linkage Map File
-l	qtlcart.lr	LRmapqtl Results file
-S	qtlcart.sr	SRmapqtl Results file (Model 6)
-M	3	Model
-t	1	Trait to analyze
-c	0	Chromosome to analyze
-d	2.0	Walking speed in cM
-n	5	Number of Background Parameters (Model 6)
-w	10.0	Window Size in cM (Models 5 and 6)
-r	0	Number of Permutations
-b	0	Number of Bootstraps

Table 3.4: Command Line Options for Zmapqtl

Traits and Chromosomes

The **-t** option allows the user to specify which trait in a data set with multiple traits is to be analyzed. For multiple trait analysis, use **JZmapqtl**. If you set the trait number to one more than the total number of traits, then all traits (except for those whose names begin with a minus sign) will be analyzed in succession: This only works with models 1, 2, 3 and 6.

One can also limit the analysis to a single chromosome with the **-c** option.

Background Parameters and Window Sizes

For models 5 and 6, one can specify the size of the window (w_s) on either side of the test interval that is blocked from having markers in the background. This option is ignored for all models except 5 and 6. The number of background parameters (n_p) is only used with model 6 and is explained above.

Permutations, Bootstraps and Jackknives

Zmapqtl allows for permutation tests and bootstrap or jackknife resamplings. The former is a way to determine experimentwise significance levels and comparisonwise probabilities (Churchill and Doerge 1994; Doerge and Churchill 1996). Phenotypes are shuffled against genotypes and the analyses are redone. For each test position, the comparisonwise probability or p-value is the proportion of permuted datasets that have test statistics less than the observed data set test statistic. It should correspond to the probability of the observed test statistic assuming a χ^2 distribution with one degree of freedom. For the experimentwise significance level, the highest test statistic in each permutation is recorded, and these are ordered at the end of the permutations. The 90, 95, 97.5 and 99th percentile values are then the experimentwise significance levels at $\alpha = 0.1,\ 0.05,\ 0.025$ and 0.01, respectively. Permutation tests are done for interval mapping within **Zmapqtl**, and interim results are stored in the files **qtlcart.z3c** and **qtlcart.z3e**. There are two distinct ways to perform the permutation test in ***QTL Cartographer***. The first is simply to have **Zmapqtl** do the permuting and analysis: You would then use **-r** with the number of permutations to perform. If you choose to do the permutation test entirely within **Zmapqtl**, you must set the number permutations to a value larger than number of permutations already completed. In this way, if you started a permutation test and your machine crashed before the test was complete, you can restart **Zmapqtl** and finish it from where it left off.

An alternative way to do the permutation test is in a batch file. For composite interval mapping, one might want to reselect the background markers with **SRmapqtl** in each permutation. To this end, one would need to permute the traits, reselect the background markers and then run the composite interval mapping. The pseudo code example in Section 2.4.2 shows how to do this without the **SRmapqtl** step.

In the bootstrap, new datasets are created from the original by sampling with replacement. New datasets are the same size as the original. The statistics are redone and printed out. See the section **Prune** as to how to do bootstrapping.

Jackknife resampling is performed by calculating n (the sample size) new estimates of the parameters: The ith estimate is calculated by deleting individual i from the dataset. The

Interim file	Created during	Contains
qtlcart.z6e	permutation test	Experimentwise state
qtlcart.z6c	permutation test	Comparisonwise state
qtlcart.z6a	bootstrap resampling	Iteration i bootstrap
qtlcart.z6b	bootstrap resampling	Iteration $i+1$ bootstrap
qtlcart.z6i	jackknife resampling	Iteration i jackknife
qtlcart.z6j	jackknife resampling	Iteration $i+1$ jackknife

Table 3.5: Examples of Interim Files for Model 6

standard deviation over these n new estimates provides an estimate of the standard deviation for the test statistic and additive and dominance effects. You invoke the Jackknife by setting the number of bootstraps to 2. **Zmapqtl** uses two interim files to perform the jackknife. If you are using Model 6 in **Zmapqtl** and your filename stem is qtlcart, then these files will be called **qtlcart.z6i** and **qtlcart.z6j**. These files contain the sum and sum of squares up to the previous and current iteration, as **Zmapqtl** runs. Initially, the qtlcart.z6i file contains columns of zeros: This is the sum before any iterations are performed. Subsequently, **qtlcart.z6j** will contain the interim state after each odd-numbered iteration, while **qtlcart.z6i** will contain the state after each even-numbered iteration. If individual i has no trait data, then the ith iteration will be skipped. For this reason, one cannot be sure that the file ending in "j" is the last iteration for odd sample sizes. It is best to look at both files at the conclusion of a jackknife experiment, and rename the interim file with the greater number of iterations to qtlcart.z6i. It this is done, then **Eqtl** will recognize it and calculate the means and sample standard deviations of the test statistic and effects.

To clarify the interim file names, we consider an example using Model 6 in **Zmapqtl** and the default filename stem "qtlcart". Table 3.5 lists the interim file names. **Eqtl** automatically looks for files named **qtlcart.z6e**, **qtlcart.z6a** and **qtlcart.z6i**. These files will be processed and the appropriate calculations done. **Eqtl** will overwrite the **qtlcart.z6b** and **qtlcart.z6j** files after completing its calculations, so if you want to save them, do so before running **Eqtl**. If you chose to use another model (say model 3), then the "6" in the filenames of Table 3.5 would be a "3".

3.4.4 Output

Here is a truncated example of the output of **Zmapqtl** for a backcross.

```
#      890840384   -filetype Zmapqtl.out
#
#       QTL Cartographer V. 1.13b, March 1998
#       This output file (qtlcart.z) was created by Zmapqtl...
#
#       It is 10:39:44 on Wednesday, 25 March 1998
#
#
```

```
#The position is from the left telomere on the chromosome
-window      10.00      Window size for models 5 and 6
-background      5      Background parameters in model 6
-Model           6      Model number
-trait           1      Analyzed trait [Trait_1]
-cross          B2      Cross
#  Test Site   * Like. Ratio Test Statistics   *     Additive
 c  m position     H0:H1      R2(0:1)    TR2(0:1)    H1:a        S1
-s
 1  1  0.0001      0.411      0.002      0.473      0.027      1.531
 1  2  0.0133      0.016      0.000      0.472      0.005      1.542
 1  2  0.0333      0.023      0.000      0.472      0.006      1.547
 1  2  0.0533      0.031      0.000      0.472      0.008      1.554
 1  2  0.0733      0.041      0.000      0.472      0.009      1.563
 1  2  0.0933      0.052      0.000      0.472      0.010      1.572
 1  2  0.1133      0.063      0.000      0.472      0.011      1.582
 1  2  0.1333      0.073      0.000      0.472      0.012      1.593
.
.
.
-e
```

For a backcross, let a be the additive effect. We have two hypotheses:

- H_0: no QTL effect at the test position, *i.e.* $a = 0$
- H_1: There is a QTL effect at the test position, *i.e.* $a \neq 0$

The first eight columns correspond to

1. Chromosome of test position
2. Left flanking marker of test position
3. Absolute position from left telomere, in Morgans.
4. Likelihood ratio test statistic for $\frac{H_1}{H_0}$. It is a χ^2 random variable with one degree of freedom for any position, meaning that a value of 3.84 or higher is evidence for a QTL. The significance level over more positions will be higher due to multiple testing.
5. r^2
6. r_t^2
7. Estimate of a (the additive effect) under H_1
8. Test statistic S for the normality of the residuals under H_1

The last 13 columns are not shown because they are only valid for F_2 design experiments. They would all be zeros if shown.

The output for an F_2 design (or any design in which dominance effects can be estimated) is similar, but has more information. For an F_2, you can estimate additive (a) and dominance (d) parameters at each position. Thus, there are four hypotheses.

- H_0: $a = 0$, $d = 0$
- H_1: $a \neq 0$, $d = 0$
- H_2: $a = 0$, $d \neq 0$
- H_3: $a \neq 0$, $d \neq 0$

and twelve full columns of output, corresponding to all possible hypothesis tests and parameter estimates. The 21 columns correspond to

1. Chromosome of test position.
2. Left flanking marker of test position.
3. Absolute position from left telomere, in Morgans.
4. Likelihood ratio test statistic for $\frac{H_3}{H_0}$.
5. Likelihood ratio test statistic for $\frac{H_3}{H_1}$.
6. Likelihood ratio test statistic for $\frac{H_3}{H_2}$.
7. Estimate of a (the additive effect) under H_1.
8. Estimate of a (the additive effect) under H_3.
9. Estimate of d (the dominance effect) under H_2.
10. Estimate of d (the dominance effect) under H_3.
11. Likelihood ratio test statistic for $\frac{H_1}{H_0}$.
12. Likelihood ratio test statistic for $\frac{H_2}{H_0}$.
13. r^2 for $\frac{H_1}{H_0}$.
14. r^2 for $\frac{H_2}{H_0}$.
15. r^2 for $\frac{H_3}{H_0}$.
16. r_t^2 for $\frac{H_1}{H_0}$.
17. r_t^2 for $\frac{H_2}{H_0}$.
18. r_t^2 for $\frac{H_3}{H_0}$.
19. S for H_1.
20. S for H_2.
21. S for H_3.

Permutation Test output

If you chose to do a permutation test (Churchill and Doerge 1994) for the purpose of estimating experiment specific threshold values, **Zmapqtl** will create two auxiliary files to store interim comparisonwise and experimentwise test statistics. If the filename stem is "qtlcart" and the model for analysis is "6", then these files will be **qtlcart.z6c** and **qtlcart.z6e**. The former file should look something like this:

```
#Row Chrom Mark Position  Original    P-Val     Count    -perm 899
   -start
    1   1   1  0.00010    0.00000   0.982202        883
    2   1   1  0.02010    0.00000   0.976641        878
.
.
.
```

whose columns are

1. Integer indicating the row.
2. Chromosome of test position.
3. Left flanking marker of test position.
4. Absolute position of test from left telomere, in Morgans.
5. Likelihood ratio test statistic of actual data. For backcrosses, this is $\frac{H_1}{H_0}$, while for F_2's, it is $\frac{H_3}{H_0}$.
6. Proportion of permuted data sets with an LR greater than or equal to the observed LR.
7. Actual count of the number of permuted data sets with an LR greater than or equal to the observed LR.

In each step of the permutation test, this file is rewritten and the number following the "-perm" token incremented. This way, if the computer crashes during a run, **Zmapqtl** can be restarted from where it left off. If you were running **Zmapqtl** with 1,000 permutations, and the process stopped at 899 as above, then restarting **Zmapqtl** with 1,000 permutations will begin with permutation 900 and continue to 1,000.

The second file, **qtlcart.z6e**, will contain two columns of numbers: the permutation and the maximal likelihood ratio over the genome in that permutation. Each permutation will add a line to the output. When enough permutations have been done, **Eqtl** can be run to summarize the experimentwise levels. A small table will be written to the log file that looks like:

```
-start
 Performed 899 permutations of the phenotypes and genotypes
Here are the Experimentwise significance levels for different alpha
Permutation significance level for alpha = 0.1   : 11.6858
Permutation significance level for alpha = 0.05  : 13.3108
Permutation significance level for alpha = 0.025 : 14.6669
Permutation significance level for alpha = 0.01  : 16.8008
-end of shuffling results
```

For each shuffle, the largest likelihood ratio test statistic over all test positions is saved in the file. At the end of the shuffling, these maximum values are sorted, and the $(1-\alpha)\times 899$ th largest is the experimentwise significance level for a test of size α. The number of permutations can be changed from 899 to any integer from 0 to 10,000. This upper bound could be made higher by changing the appropriate definition in the Main.h source file and recompiling. In general, we find that 1000 permutations is a sufficient number. In a test, values of 1000 and 17,000 were used with little difference in the ultimate comparisonwise and experimentwise values.

3.5 JZmapqtl

JZmapqtl implements interval and composite interval mapping for multiple traits (Jiang and Zeng 1995). It is very similar to **Zmapqtl** except that it can jointly analyze more than one trait. It is best used after **Zmapqtl** when one suspects that two traits are correlated.

3.5.1 JZmapqtl Options

Table 3.6 shows the command line options specific to **JZmapqtl**. Most are the same as those for **Zmapqtl**. One thing to note is that there is no facility for permuation tests or bootstraps at this time.

Option	Default	Explanation
-i	qtlcart.cro	Input File
-o	qtlcart.z	Output File
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-S	qtlcart.sr	SRmapqtl results (Model 6)
-E	qtlcart.eqt	Eqtl results (Model 7)
-s	893339277	Random Number Seed
-M	3	Model [3,6,7], 3=>IM
-t	1	Trait to analyze
-c	0	Chromosome to analyze (0=>all)
-d	2.000000	Walking speed in cM
-n	5	Highest rank for Background Parameters (Model 6)
-w	10.000000	Window Size in cM (Model 6)
-I	1	Hypothesis test

Table 3.6: Command Line Options for JZmapqtl

3.5.2 Output

JZmapqtl will create a number of different output files depending on the number of traits in the joint analysis. There will be one file per trait that has estimates for the parameters for that trait. These files will end in .z#, where # is a number indicating the trait. There will be one other file, ending in .z0 that contains the results of the joint liklihood ratio.

The joint results file ending in .z0 will have four columns corresponding to the chromosome, marker, markername and test position. Then there will be column giving the joint liklihoods for the test position for all possible hypothesis tests (see next section).

The single trait files, ending in .z#, will have the results for the numbered trait. In addition to the chromosome, marker, markername and test position, the likelihood ratio and parameter estimates will be given. All columns are labelled, and the parameters are the same as explained in the **Zmapqtl** section.

3.5.3 Usage Hints

Trait Selection

You can select traits to include in the analysis in three ways. Suppose that you have t traits in your data file.

1. Set the trait to analyze at 0, so that no traits except those beginning with a + (plus sign) are analyzed. You would need to edit the .cro file first to prepend a + to all traits you want in the analysis.
2. Set the trait to a value in the range $1 - t$, inclusive. You will then get single trait results for the selected trait.
3. Set the trait to a value greater than t. All traits will be put in the analysis, unless they begin with a − (minus sign).

Hypothesis tests

You need to set the hypothesis test for SF_x and RF_x crosses. The default of 1 is fine for crosses in which there are only two marker genotypic classes (backcrosses and recombinant inbreds). For SF_x and RF_x, values of 30, 31 or 32 are valid. Recall that we have the following hypotheses:

1. $H_0 : a = d = 0$
2. $H_1 : a \neq 0, d = 0$
3. $H_2 : a = 0, d \neq 0$
4. $H_3 : a \neq 0, d \neq 0$

For 30, we test $H_3 : H_0$. For 31, we test $H_3 : H_0$, $H_3 : H_1$ and $H_1 : H_0$. For 32, we test $H_3 : H_0$, $H_3 : H_2$ and $H_2 : H_0$. 30 is probably fine for initial scans.

Also, if you do only have two genotypic classes, then 10 is the same as 1 for the hypothesis test.

Model 6

For Model 6, be sure to run **SRmapqtl** first. Once done, **JZmapqtl** will use all markers for specified traits that have rank as good or better than specified with the **-n** option.

You might also want to edit the **SRmapqtl.out** file prior to running **JZmapqtl** with model 6. You could pick out the top two or three ranked markers for each trait to be absolutely certain of which markers are being used as cofactors.

Model 9

This is a new option that doesn't do any analysis: It merely prepares an input file for the **MultiRegress** module. **JZmapqtl** will walk down the genome, and for each site will calculate the expected value of a QTL genotype. The output file will contain these expected genotypes and the trait values. **MultiRegress** can then be used to estimate QTL positions and additive and dominance effects. See the manual page for **MultiRegress** for more information.

G x E Analysis

One special case of G x E analysis has been incorporated into **JZmapqtl**, namely the situation where a set of genotypes is raised in more than one environment. The value of the trait in each environment is treated as a separate trait for the common genotype. For this type of data, use hypothesis 14 or 34 to invoke the G x E analysis. Hypothesis 14 is for data with two marker genotypes, while 34 is for three marker genotypes. There will be an extra column in the output that give a likelihood ratio for a G x E effect versus no effect. When running **Eqtl** subsequent to doing a G x E analysis, be sure to specify the same hypothesis test.

3.6 MImapqtl

MImapqtl implements QTL mapping analysis for multiple QTL in multiple intervals for a single trait in a single environment. It can begin analysis from an initial model specifying the positions of QTL, or *de novo*, that is with no initial model. If given an initial model, the program will estimate the parameters, refine the estimates of QTL positions within intervals, test the significance of all parameters, search for more QTL, search for epistatic interactions and finally calculate the r^2 and breeding values for the model. If analysis is initiated *de novo*, then there will be a search for QTL, a search for interactions and calculation of r^2 and breeding values. **MImapqtl** can also do a single pass search for a new QTL and create a likelihood ratio profile for a new putative QTL given any initial model.

For m putative QTL, the model is

$$\begin{aligned} y_i &= \mu + \sum_{r=1}^{m} \alpha_r x_{ir}^* + \sum_{r=1}^{m} \delta_r z_{ir}^* + \sum_{r \neq s} \beta_{rs}^{AA} x_{ir}^* x_{is}^* + \sum_{r \neq s} \beta_{rs}^{AD} x_{ir}^* z_{is}^* \\ &\quad + \sum_{r \neq s} \beta_{rs}^{DA} z_{ir}^* x_{is}^* + \sum_{r \neq s} \beta_{rs}^{DD} z_{ir}^* z_{is}^* + e_i \end{aligned} \tag{3.6}$$

where y_i is the trait and e_i the residual for individual i. The parameters α_r, δ_r are the additive and dominance effects of QTL r. The β's are epistatic interactions. The superscripts on the β's are for the type of interaction: We distinguish between additive by additive (AA), additive by dominance (AD), dominance by additive (DA) and dominance by dominance (DD) interactions. There is also a mean denoted by μ and a variance for the residuals (σ^2) which are assumed to have a normal distribution and mean zero. The x^* and z^* are coded variables denoting the genotype of the putative QTL. If there are only two marker genotypic classes, the z^*'s are all set to zero. Otherwise, $z_{ir}^* = 0.5$ for a heterozygote and -0.5 for a homozygote. The x_{ir}^* are

defined differently for different crosses and are shown in Table 3.7. The sum over $r \neq s$ means over all unordered QTL pairs and we use the convention $r > s$.

		x^*				
	z^*	B_1	B_2	SF_i	RF_i	RI_i
QQ	-1/2	1/2	0	1	1	1
Qq	1/2	-1/2	1/2	0	0	0
qq	-1/2	0	-1/2	-1	-1	-1

Table 3.7: Coded variables

The likelihood function of the data given the model is a mixture of normal distributions

$$L(\mathbf{E}, \mu, \sigma^2 | \mathbf{Y}, \mathbf{X}) = \prod_{i=1}^{n} [\sum_{j=1}^{g^m} p_{ij} \phi(y_i | \mu + \mathbf{D}_j \mathbf{E}, \sigma^2)] \tag{3.7}$$

In (3.7), the p_{ij} are the probabilities of the multilocus genotypes conditioned on marker data. The variable g is the number of genotypic classes for the experimental design: For backcrosses and recombinant inbred lines, $g = 2$, while for intercrosses it is $g = 3$. In practice, it is often infeasable to do the sum over all g^m multilocus genotypes: We use a subset of the most frequent genotypes. The parameters are in $\mathbf{E}$ while the coded indicator variables are in $\mathbf{D}$. $\phi(y_i|\mu, \sigma^2)$ is the normal density function with mean μ and variance σ^2. We use an EM (expectation maximization) algorithm to obtain maximum likelihood parameter estimates (Kao and Zeng 1997; Zeng, Kao, and Basten 1999).

3.6.1 MImapqtl Options

Table 3.8 shows the command line options specific to **MImapqtl**.

If you choose to begin analysis from some initial model, it should be formatted as an **Rqtl** output file and specified with the **-E** option. For data sets with multiple traits, you can specify the trait to analyze with the **-t** option. If the specified trait is 0, then all traits whose names begin with a plus sign (+) will be analyzed (one at a time). If the specified trait is larger than the number of traits, then all traits will be analyzed one at a time, unless their names begin with a minus sign (−).

You may specify a maximal number of QTL and epistatic interactions to fit in the model. The limitations of C on a 32 bit computer mean that you can only fit up to 19 QTL in a cross with three marker genotypes (SFx and RFx lines). Thus, 19 will be a hard limit on the number of loci at this time. In practice, it is not wise to try to fit more than $2\sqrt{n}$ parameters, where n is the sample size. Thus, **MImapqtl** will automatically adjust this number so that it is no greater than $2\sqrt{n}$ for backcrosses and recombinant inbred lines, $\sqrt{n}$ for lines with three marker genotypes (since each QTL has an additive and a dominance effect). Once main effects are fitted for the QTL, **MImapqtl** searches for epistatic effects. Again, it will only try to fit up to $2\sqrt{n}$ parameters so only $2\sqrt{n}$ minus the number of main effects will be fitted. For example, if you have a sample size of 400, then $2\sqrt{n} = 40$. For an SF2 line, if you find 10 QTL, then 20

Option	Default	Explanation
-i	qtlcart.cro	Data File
-o	qtlcart.mim	Output File
-e	qtlcart.log	Error File
-m	qtlcart.map	Genetic Linkage Map File
-E	qtlcart.eqt	Initial model file
-O	qtlcart.mqt	Output model file
-s	1590337669	Random Number Seed
-t	1	Trait to analyze
-q	19	Maximum number of QTL to fit
-k	19	Maximum number of epistatic interactions
-d	2.000000	Walking speed in cM
-S	1	Information criterion [1-6]
-L	0.000000	Threshold for adding/dropping parameters
-I	smprtSeC	Work code
-p	0	Phase of analysis

Table 3.8: Command Line Options for MImapqtl

parameters have been fitted (10 additive and 10 dominance effects). Thus, there will be up to $40 - 20 = 20$ possible epistatic effects that can be fitted.

The walking speed is identical to that in **Zmapqtl** and **JZmapqtl** and is used during the refinement of QTL positions and the search for new QTL. For the refinement of QTL position, for each QTL, the position is moved within the QTL interval from one end to the other and an information criterion is calculated for each position. The minium over the interval is the best position for the QTL.

The information criterion is is a function of the likelihood ratio and the number of parameters that gives an indication of how good the model fits the data. This is a function in form

$$I(L_k, k, n) = -2(log(L_k) - kc(n)/2)$$

where L_k is the likelihood for a k parameter model and $c(n)$ is a penalty function and log is the natural log function. The penalty function $c(n)$ takes one of six forms:

1. $c(n) = log(n)$ (Schwarz 1978)
2. $c(n) = 2$ (Akaike 1969)
3. $c(n) = 2log(log(n))$ (Hannan and Quinn 1979)
4. $c(n) = 2log(n)$ (Broman 1997)
5. $c(n) = 3log(n)$ (Broman 1997)
6. $c(n) = 0$

Use the numbers above with the **-S** option to indicate which information criterion you want to use. When comparing a model with k parameters to one with $k-1$ parameters, if $I(L_{k-1}, k-1, n) - I(L_k, k, n)$ is greater than the threshold value specified with the *-L* option, then the parameter is considered significant. During a scan over the entire genome, if the maximum of $I(L_{k-1}, k-1, n) - I(L_k, k, n)$ is greater than the threshold, then that position parameter is retained. A non-zero penalty function supersedes the requirement of a true threshold. Therefore, if using one of the first five penalty functions, you can use a threshold of 0.0. For function six (that is, no penalty), it is best to do a permuation test to determine the initial threshold.

In general, it is probably best to start with information criterion 1 with a threshold of 0.0. We have done some initial simulations to support the utility of this approach. If no QTL are identified using information criterion 1, then one could use no penalty function and do a permutation test for the threshold.

Phase of Analysis

It is often a good idea to rerun an analysis for various reasons. Think of each run as a "phase" or step. For example, the first phase might be to identify QTL main effects. The second phase would then test the significance of each QTL just identified. The third phase might be to refine the positions of the remaining QTL and the fourth phase to estimate interactions. It is convenient to use the phase variable for this type of analysis. The phase variable, if set to an integer value greater than zero, will modify the names of the output file and the model input and output files. For a filename stem *qtlcart* and phase *N*, the output file will be **qtlcartPhaseN.mim**. The output model will be placed in **qtlcartPhaseN.mqt,** and the input model will be assumed to be **qtlcartPhaseN-1.mqt**. The phase variable is set with the **-p** option, and if greater than zero, will be incremented at the end of each successful run of **MImapqtl**.

Here is a sequence using the example dataset **mletest.cro** along with its map file **mletest.map,** both of which come with the programs. Assume that these two files have been placed in an empty subdirectory which is now the current working directory.

```
% MImapqtl -A -V -I smprtSeC  -L 0.0 -S 1 -p 1 -X mletest &
% MImapqtl -A -V -I sMPrTseC    &
% MImapqtl -A -V -I sMPRtseC    &
% MImapqtl -A -V -I sMPrtSeC    &
% MImapqtl -A -V -I sMPrtsBC    &
```

The first invocation sets the filename stem, the information criterion and threshold for adding parameters and indicates that it is phase 1. The **-I** option tells **MImapqtl** to search for additive QTL. The second invocation tests each QTL found in the first phase. The third step refines the positions of all remaining QTL. The fourth step searches for more QTL (and probably won't find any). The fifth step searches for interactions between the identified putative QTL. The phase variable is updated after each step, so **MImapqtl** knows where to find the results from the previous step. The work code specified by the **-I** option is explained in the next section.

Work Code

The *Work Code* must be specified with an eight (8) letter string. Each letter in the string is a flag to tell the program whether to do a certain step. Some of the flags have options to modify the behavior of that step. The eight letter string starts from position 0 and continues on to position 7. In general, a lower case letter indicates that the function should be skipped, while an upper case letter tells **MImapqtl** to do the step. The labels on the items below indicate the positions.

0. Scan flag

   This can take on values **S** or **s**. If **S**, then **MImapqtl** will go into scan mode, that is it will scan the genome for a new QTL beginning with any model. At the end of this scan, it will print out a likelihood profile for the existence of a new QTL. The user can then plot the values and decide where to place a new QTL.

1. Model flag

   This flag tells **MImapqtl** whether to use the initial model specified with the **-E** option. If **M**, then **MImapqtl** will begin its analysis with the initial model. If **m**, then it will start the analysis from a state with no QTL. If you use **m**, then you should also specify **prt** in positions 2, 3 and 4. For example, **smprtSEC** would make sense: It would search for QTL *de novo*.

2. Parameter flag

   Use a **P** here if you want **MImapqtl** to re-estimate the paramters in the initial model. Use a **p** if you want to skip this step. The case of this position should almost always match that of position 1: It makes little sense to estimate parameters in a model without any parameters.

3. Refine postion flag

   Use an **R** here if you want **MImapqtl** to refine the position estimates in the initial model. Use an **r** if you want to skip this step. If you don't have an initial model, then this should be **r**. This is most useful if your initial model was generated from a run of **Zmapqtl**or **JZmapqtl**. The **R** value causes the position to be optimized within the current interval. If you want to extend the refinement to adjacent intervals, use **A**.

4. Test flag

   For initial models, you may want to test the significance of all parameters before searching for new QTL. Use a **T** here if you want **MImapqtl** to test the significance of the parameters in the initial model. Use a **t** if you want to skip this step. If you don't have an initial model, then this should be **t**. This position also allows you to test the dominance effects alone: You can use **D** in place of **T**. Finally, you can test the current set of epistatic interactions by using a **E** in this position.

5. Search flag

Use an **S** here if you want **MImapqtl** to search for more QTL. Use an **s** if you want to skip this step. You can also specify an **A** if you only want to search for the additive effects of putative QTL (that is, don't search for dominance effects in crosses with three marker genotypes). Finally, if you use a **D** here, **MImapqtl** will only search for dominance effects at putative QTL locations that don't already have them.

6. Epistasis flag

   Use an **E** here if you want **MImapqtl** to search for epistatic effects. Use an **e** if you want to skip this step. If you want to use a backward elimination approach, then use a **B** in this position. The backward elimination approach will only be used if the number of possible interactions plus the number of main effects is less than $2\sqrt{n}$. If you want to relax this restriction, then use a **U** in place of the **B**.

7. Covariance flag

   Use a **C** here if you want **MImapqtl** to calculate the variance-covariance matrix, r^2 values and breeding values for the final model. Use a **c** if you want to skip this step. You may also use an **R** in this position. In that case, **MImapqtl** will calculate the residuals for the final model, replace the trait values with the residuals and print out a new data set with the same markers to a file (**stem.res** by default, where **stem** is the filename stem). If the phase variable is set to a number N greater than zero, then the residual data set will be written to **stemPhaseN.res**. The **C** and **R** options are mutally exclusive.

The default work code is **smprtSeC**, which searches for QTL starting with an initial model containing no QTL. If you want to do an analysis withan initial model, then **sMPRTSeC** would be appropriate. For a scan to find a new QTL, you might use **SMPrtSec** recursively, adding a QTL in each step.

If you have a model and simply want to create a new data set with the residuals replacing the traits, then use **sMPrtseR** as the work code.

3.6.2 QTL Search

For a model with k QTL and information criterion $I(L_k, k, n)$, we search for $k+1$st QTL over all intervals that do not presently have a QTL in them. For each of these intervals, we walk along the interval and calculate the information criterion $I(L_{k+1}, k+1, n)$ for the presence of a QTL. We keep track of the minimum information criterion (equivalent to the maximum likelihood) within each interval. When all intervals have been tested, the minimum over intervals is determined and compared to the information criterion of the k QTL model. If $I(L_k, k, n) - I(L_{k+1}, k+1, n)$ is greater than the threshold, the QTL at that site is retained in the model. The process repeats until no new QTL are retained.

If **MImapqtl** is run in scan mode (**S** in position 0 of the work code), then the quantity $I(L_k, k, n) - I(L_{k+1}, k+1, n)$ is written to the output file for each tested position. This yields a likelihood ratio profile giving the strength of evidence for a new QTL at the tested sites.

For crosses with three marker genotypic classes, both an additive and a dominant effect are added when testing for a new QTL. One can scan for additive effects only with the **A** option

used in position five of the work code. If a **D** is used in this position, the search for dominance effects is restricted to previously identified QTL.

3.6.3 Epistatic Interactions

Once QTL have been identified, the program can search for interactions. For each pair of QTL, the information criterion is calculated with an interaction effect. The minimum over all pairs is compared to the information criterion of a model without the epistatic effect. If the difference is greater than the threshold, the effect is retained and the process is repeated. In each step, the new effect is tested within the backgroud of all the QTL main effects as well as previously identified interactions.

For crosses with two marker genotypic classes, only additive by additive effects are tested. For those with three classes, additive by dominance, dominance by additive and dominance by dominance effects are also tested.

By default, **MImapqtl** does a forward stepwise search for epistatic terms. If you want to try a backward elimination approach, use a **B** instead of an **E** in the appropriate position. Also, be aware that if there are too many epistatic terms, the request for a backward elimination approach will be ignored in favor of a forward search. **MImapqtl** only allows for $2\sqrt{n}$ parameters. If you want to relax this constraint, then use a **U** instead of the **B**. **MImapqtl** will then allow for up to $n - 1$ parameters.

3.6.4 Threshold

If no penalty function is to be used, then it is important to specify the proper threshold for adding and dropping parameters. If the threshold is too low, then no QTL will be found. If too high, then **MImapqtl** will continue to find QTL until it reaches its upper limit. One can specify a low threshold and limit the number of QTL to find. Alternatively, one can perform a permutation test to get an inital threshold. We do this using a shell script to cycle through the **Prune** and **MImapqtl** commands for a predetermined number of repetitions.

Suppose qtlcart is the filename stem, and use m as the work code, s for the type I error rate, c for the output column to process, and r for the number of repetitions, then this snippet of psuedo-code will do a permutation analysis for **MImapqtl**.

```
echo "\n-start" > qtlcart.mim.ewt
i = 1
while (i < r) {
        Prune -A -V -i qtlcart.cro -b 2
        MImapqtl -A -V -I m -S ic -i qtlcart.crb
        GetMaxLR.pl -r i -C c < qtlcart.mim >> qtlcart.mim.ewt
        delete qtlcart.mim
        i = i + 1
}
echo "\n-end" >> qtlcart.mim.ewt
EWThreshold.pl < qtlcart.mim.ewt
```

The programs **GetMaxLR.pl** and **EWThreshold.pl** are **Perl** scripts that come with the

UNIX distribution of the programs. There is also a **D** shell script called **PermuteMI** that can be modified for your particular data set. Use the script on the original data to obtain an initial threshold. This threshold can be used with **MImapqtl** and the *SmprtSeR* work code to find a QTL and compute residuals. From there, a new permutation test can be run to get a new threshold to search for more QTL.

Chapter 4

Visualization of Results

The final step in analyzing your data will be to summarize your results, either graphically or as a compact set of estimates for QTL positions and effects. We have provided some utilities that read the output of the analysis programs and reformat it for use in graphics packages. The freeware program **Gnuplot** is recommended as a graphics engine, but the results could be plotted in any plotting package on any machine. All of the results from the analysis programs are simple text files, and all the reformatted files are also simple text.

Figure 4.1 is a schematic of the programs and files that are involved in this step. **Eqtl** is a utility that quickly picks out the possible QTLs from the results of **Zmapqtl**. **Preplot** can read the output of **Rqtl**, **LRmapqtl**, **Zmapqtl**, **MImapqtl** and **Eqtl** and produce simple files containing two columns of text corresponding to the values for the abscissa and ordinate of a plot. These files in turn can be plotted by **Gnuplot**, or imported into various plotting packages on various platforms.

4.1 Eqtl

Zmapqtl outputs a great deal of information: Often the experimenter will want a quick summary of the positions and effects of the QTLs. The program **Eqtl** scans the output of **Zmapqtl** and reformats it. Part of the output of **Eqtl** is identical to the output format of **Rqtl**. This is convenient if the experimenter would like to do simulation studies with a set of estimated QTLs. The output of **Eqtl** can be used as the input to **Rcross** (with the appropriate genetic linkage map), and new data sets can be simulated to examine the power of the different methods to detect the QTLs. Finally, the output of **Eqtl** can be read by **Zmapqtl** and used to create virtual markers to be used as covariates in composite interval mapping (see model seven of Section 3.4.2).

The remaining output is more readable and is appropriate if the experimenter is not interested in doing further simulations. The positions of the QTL are given in Morgans from the telomere rather than recombination frequencies from the flanking markers.

Eqtl also appends some output to the **SRmapqtl.out** file. Once putative QTL are identified, **Eqtl** determines which markers are closest to these QTL and writes them to the *SRmapqtl.out* file. These markers will be in a table similar to the output of **SRmapqtl** itself.

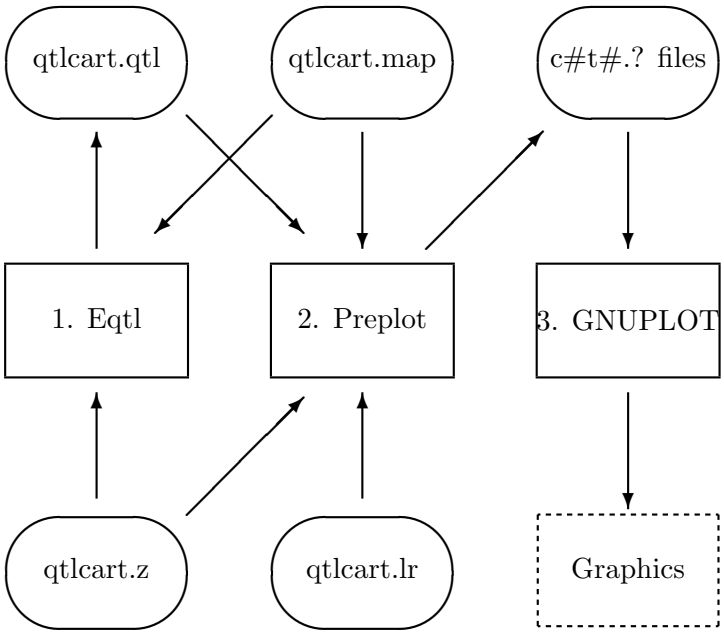

Figure 4.1: Visualization Schematic

They can be read by **Zmapqtl** when running model 8.

In addition to reformatting the output of **Zmapqtl**, **Eqtl** will automatically detect whether a permutation test, jackknife or bootstrap experiment had been done. If such results exist, **Eqtl** will open and summarize them. For example, if you do a permutation test with **Zmapqtl** using interval mapping, an interim file **qtlcart.z3e** is created and appended to for each permutation. **Eqtl** will read this file and calculate experimentwise threshold values from it. Standard significance thresholds will be written to the log file. The user can specify a type I error rate (size) and **Eqtl** will calculate a threshold value relevant to it. Once done, the threshold value will be remembered and used by subsequent runs of **Eqtl** or **Preplot**.

For bootstrap results from **Zmapqtl** using interval mapping, **Eqtl** looks for a file **qtlcart.z3a**. If found, **Eqtl** will read in the sums and sums of squares of the likelihood ratio, additive effect and dominance effect at each position and print the mean and sample standard deviations into a summary file (**qtlcart.z3d**). **Eqtl** does similar calculations for the jackknife results that would be in **qtlcart.z3i**.

Table 4.1 shows the command line options specific to Eqtl.

Option	Default	Explanation
-z	qtlcart.z	(Composite) Interval Mapping Results
-o	qtlcart.eqt	Output File
-m	qtlcart.map	Genetic Linkage Map File
-M	3	Model from Zmapqtl
-H	10	Hypothesis Test (10,14,30,31,32,34)
-S	10.0	Significance threshold
-a	0.05	Size (α)
-L	0	Output LOD scores? (0=no,1=yes)
-I	ZM	Work Code

Table 4.1: Command Line Options for Eqtl

4.1.1 Options

Files

Similar to other programs in the ***QTL Cartographer*** system, the input and output files can be specified. A genetic linkage map and a file containing the results of **Zmapqtl** must exist and be properly specified to **Eqtl**.

Which Results?

The output file from **Zmapqtl** may contain the results of analyzing different traits using different models. Furthermore, in F_2 and other populations in which dominance can be estimated, it is possible to test different sets of hypotheses. The user can specify which results from the **Zmapqtl** output file to process. The **-M** option tells **Eqtl** to examine the results from using the specified analysis model. An integer value should be given after the **-M** option. By default, **Eqtl** looks for the results from Model 3, or interval mapping. If you have done composite interval mapping, with say model 6, then you should specify **-M 6** on the command line (or in the interactive menu). If model 6 was the last model run in **Zmapqtl**, then **Eqtl** should already be aware of that fact.

For F_2 design experiments (and others where there are three genotypic classes), various hypothesis tests can be performed. Recall that there are four hypotheses H_0, H_1, H_2 and H_3 that are explained in Section 3.4.4. Using **-H** with 10, 20, 30, 31 or 32 allows you to specify which likelihood ratio should be the basis for declaring QTL. Table 4.2 summarizes the output for various values used with the **-H** option.

Eqtl can process the results of running permutations, bootstraps and jackknives as well as of **JZmapqtl**. By default, **Eqtl** will only try to process the output of **Zmapqtl** and **JZmapqtl**. To change what **Eqtl** tries to process, use the *-I* option with a work code. This option takes a string as an argument, and each letter in the string tells **Eqtl** to do something. The default string is *ZM*, which indicates to process the single trait (**Zmapqtl**) and multi-trait (**JZmapqtl**) output. The letters *PBJ* tell **Eqtl** to process the permutation test, bootstrap and jacknife analysis files as well. Any combination of *PBJZM* may be used as the work code, but

-H	Likelihood Ratio	a	d	S	$r^2,\ r_t^2$
10, 14	$H_1 : H_0$	a_1	-	S_1	$H_1 : H_0$
20	$H_2 : H_0$	-	d_2	S_2	$H_2 : H_0$
30, 34	$H_3 : H_0$	a_3	d_3	S_3	$H_3 : H_0$
31	$H_3 : H_1$	a_3	d_3	S_3	$H_3 : H_0$
32	$H_3 : H_2$	a_3	d_3	S_3	$H_3 : H_0$

Table 4.2: Output for different **-H** values with Eqtl

if P is to be used, then put it first because order of processing is the same as the order of the letters.

Other Options

Eqtl essentially finds the peaks in the likelihood ratio graph of the results from **Zmapqtl**. It goes along the chromosome, and determines whether the likelihood ratio test statistic is increasing or decreasing. Upon a change, it picks out the position and estimates of other parameters. The user can specify that the peaks of interest need be higher than some "Significance threshold" to be considered QTLs. The default is 3.84, that is, any peak that is less than 3.84 is ignored. This can be changed with the **-S** option. If you have run **Zmapqtl** and done a permutation test, **Eqtl** automatically reads the output and sets the significance threshold subject to the value of the size, set with the **-a** option. For a size of α, the the $100(1-\alpha)$-percentile is calculated from the experimentwise test values.

The final option is a flag to output LOD scores rather than likelihood ratios. The default behavior of the ***QTL Cartographer*** system is to use a likelihood ratio test statistic (LR) rather than a LOD score. For a hypotheses H_i, let L_i be the likelihood of the data given the hypothesis. For a pair of hypotheses H_0 and H_1, this would yield L_0 and L_1. The LOD score is defined as

$$LOD = -\log \frac{L_0}{L_1}$$

The likelihood ratio test statistic (LR) is

$$LR = -2\ln \frac{L_0}{L_1} = -2\ln 10^{-LOD} = 2(\ln 10)LOD = 4.605LOD$$

and thus

$$LOD = -\log \exp(-\frac{LR}{2}) = \frac{1}{2}(\log e)LR = 0.217LR$$

4.2 Preplot

Preplot reformats the output of the analysis programs so that they may be plotted by **Gnuplot**. The output files could be imported into any programs. The default behavior of **Preplot** is what we term the "automagic" mode. **Preplot** reads the **Zmapqtl** output file, determines

Option	Default	Explanation
-o	qtlcart	Gnuplot Control File Name
-m	qtlcart.map	Genetic Linkage Map File
-q	qtlcart.qtl	QTL or Estimated QTL file
-l	qtlcart.lr	LRmapqtl Output File
-z	qtlcart.z	Zmapqtl Output File
-S	10.0	Significance Threshold
-T	x11	Terminal
-L	0	Output LOD scores? (0=no,1=yes)
-i	1	Hypothesis (for F2 design)

Table 4.3: Command Line Options for Preplot

what analyses have been done, and then reformats all of these analyses in a logical way. There will be a separate graph for each trait and each chromosome. **Preplot** will attempt to put the results from different models in **Zmapqtl** and from **LRmapqtl** on the graphs, along with any information from the **Rqtl**, **Eqtl** and **MImapqtl** output file (if they exist), and a significance threshold (which can be set in the interactive menu or on the command line).

Table 4.3 shows the command line options specific to **Preplot**. In general, it will not be necessary to change any options to **Preplot**. Most of the proper values should have been set by other programs in the ***QTL Cartographer*** suite. You might want to use the **-L** command to tell **Preplot** to convert LR values into LOD scores. In any case, the output of **Preplot** is ready for import into **Gnuplot**. There will be a number of output files. One is a plot control file, that has commands that **Gnuplot** understands. The other files simply contain two columns of numbers for the x and y coordinates to plot. The names of the files indicate what the numbers are for. They all start with a lower case c , which indicates chromosome. Following the c is an integer indicating which chromosome, then there is a t followed by an integer indicating the trait. Then there is a period and a file extension that indicates the results contained in the file. For the results of composite interval mapping, the ".z" filename extension will be followed by an integer from 1 to 7 indicating the model used for the analysis. For example, the file **c2t3.z6** would have the results of composite interval mapping for trait 3 on chromosome 2 in it.

4.2.1 Printing Results

One option that is useful to change is the "Terminal" setting. This will be set correctly if all you want to do is view the graphs on your screen with **Gnuplot**. If you want to get a hardcopy printout, you have two alternate choices for the "Terminal" option. If you have a postscript printer, then use "postscript" as the terminal. Run **Preplot**, and then run **Gnuplot** as explained in Section 4.3. You will not see any output, but a file **qtlcart.ps** will be created (or **stem.ps**, where "stem" is your filename stem). This file can be sent to any postscript printer. The other alternative is "hpljii", which does something similar for HP-LaserJet II's (the output file will be **stem.hp**). You could use the "hpljii" option for "Terminal", and then edit the **stem.plt** file to change the type of printer to anything that **Gnuplot**supports: See

the **Gnuplot**manual (Williams and Kelley 1993) for more details.

Extension	Meaning
s	Significance Threshold
lr	Linear Regression results
z#	Composite interval mapping results
q	Quantitative trait locus data (from **Rqtl**)
e	Estimated QTL positions (from **Eqtl**)
m	Estimated QTL positions (from **MImapqtl**)

Table 4.4: Filename extensions for Preplot output

4.3 GNUPLOT

Gnuplot is free plotting software available for UNIX, Macintosh and Windows machines. It is an interactive package. The basic idea behind the program is to read in simple files of numbers and plot them. The files of numbers contain two columns, one for the abscissa and one for the ordinate. **Preplot** takes care of reformatting the output of the analysis so that **Gnuplot** can read the results and plot them. We have placed copies of **Gnuplot** for the three platforms on our ftp server.

4.3.1 Basic GNUPLOT

In many ways, **Gnuplot** is similar to **Mapmaker/EXP**in that it is an interactive, command driven program. Once **Gnuplot** has been started, the user can type "help" to get information on how to use the program. There are commands to change the terminal type, load files and specify the output device. Thus, one can view or print the images created by **Gnuplot**.

If you have run **Gnuplot**, you should have a plot control file with a ".plt" extension. Suppose that this file was **stem.plt**. You can start up **Gnuplot** and issue the command

```
gnuplot> load "stem.plt"
```

to see the plot specified by **stem.plt**.

See the **Gnuplot** manual for more information on this program (Williams and Kelley 1993). Of special interest may be the different types of printers supported by **Gnuplot**. If you choose "postscript" as your terminal type in **Preplot**, then you will find a pair of lines on the **stem.plt** file that look like this:

```
set term postscript
set output "stem.ps"
```

You can change the token "postscript" token in that file to any printer that **Gnuplot** supports and sent the **stem.ps** file to that printer.

Chapter 5

Tutorial Examples

5.1 General tactics and notes

These ***QTL Cartographer*** exercises have been used at North Carolina State University in class (Statistics 591o) and in the Summer Institute in Statistical Genetics. We have also presented these labs in South Africa and New Zealand. Initially we used the NCSU Statistics Instructional Computing Laboratory (SICL) equipped with Sun workstations running Solaris, but have moved to Windows NT workstations of late. The exercises can be done on any platform that ***QTL Cartographer*** runs on.

As a general rule, we suggest creating a separate subdirectory (folder) for each data set and copying the original input files into that subdirectory. This will help to organize your work and preserve your original files. If any exercise goes awry you can delete the subdirectory and start over.

Beginning with the first program you run, a resource file called **qtlcart.rc** is created and updated for each subsequent program. This file keeps track of all the parameters and file names that you use. In addition, a log file will record which specific parameters were used with which specific programs, and when the programs were run. Thus, the **qtlcart.rc** file keeps track of the current settings, and the **qtlcart.log** file records the history of parameter settings. You can look at any of these files or any other files that ***QTL Cartographer*** creates by opening them in any text editor.

QTL Cartographer on Macintoshes and Windows-based PCs behaves slightly differently than UNIX host. They maintain one copy of the **qtlcart.rc** file in the subdirectory (folder) where the applications are located. You can specify a working subdirectory (folder) in any of the ***QTL Cartographer*** programs, and this will be recorded in the **qtlcart.rc** file (see 1.6.1 for details). For the purposes of this tutorial, create a directory called **qwork**: If you are on a Macintosh or a PC, create it in the directory (folder) one level higher than where the binaries are. If you are on a UNIX machine, create the **qwork** subdirectory in your root directory.

There is a web page for ***QTL Cartographer***

`http://statgen.ncsu.edu/qtlcart/cartographer.html`

which is the good place to keep abreast of new information. The **README** file from the ftp server is linked to the web page. The programs are also linked to the web page, so you can

download them using some web browsers. The entire manual as well as the man pages have been been translated into html.

5.1.1 Conventions

The exact way to run the programs is slightly different under UNIX, Macintosh and Windows. When specific commands are presented, they will be shown as typed on a UNIX workstation. For example, if the current working directory has the **mletest.map** and **mletest.cro** files in it, then you could run **Qstats** in a terminal shell as follows:

```
% Qstats -A -V -X mletest
```

There are a few things to note:

1. The percent sign (%) is assumed to be the UNIX prompt. This prompt may differ on different UNIX systems. For example, the Statistical Genetics Group has a Solaris workstation called brooks and the prompt looks like `brooks: /qwork %` rather than a single percent sign.

2. The name of **Qstats** is `Qstats`, and in the UNIX environment, it is case sensitive. The command `QSTATS` is different and may not be recognized. Throughout this document, the proper UNIX names have been used for the various programs in the ***QTL Cartographer*** suite.

3. Command line options begin with a minus sign and end with a single letter (with a couple of exceptions...see **Rmap**). They should be surrounded by a space, and if they take an argument, that argument should be surrounded by spaces. Command line options cannot be combined. For example, you cannot combine **-A -V** as **-AV**. A command line option that requires an argument must have one provided: **-X** requires a filename stem.

Now we can contrast this with Windows and Macintosh. Macintosh programs are started by double clicking on their icons. The Windows programs can be run in a command window in a fashion similar to a UNIX shell, or by double clicking a program icon similar to the Macintosh system. Consider first the command line invocation. If you open a command window, you can type in ***QTL Cartographer*** commands just as you would in UNIX. The command window is not case sensitive, so `qstats` will work as well as `Qstats`. The command line options are case sensitive. Also, the actual programs have `.exe` extensions: These do not have to be typed. If the `PATH` variable has the **QTLCartWin\bin** directory in it, then you can use the programs in any working directory and the resource file will reside there rather than where the binaries are. Suppose **mletest.map** and **mletest.cro** are in a directory **c:\QTLCartWin\test**. Then you might run **Qstats** as follows

```
c:> cd qtlcart\test
c:\QTLCartWin\test > qstats -A -V -X mletest
```

The command line options are still case sensitive.

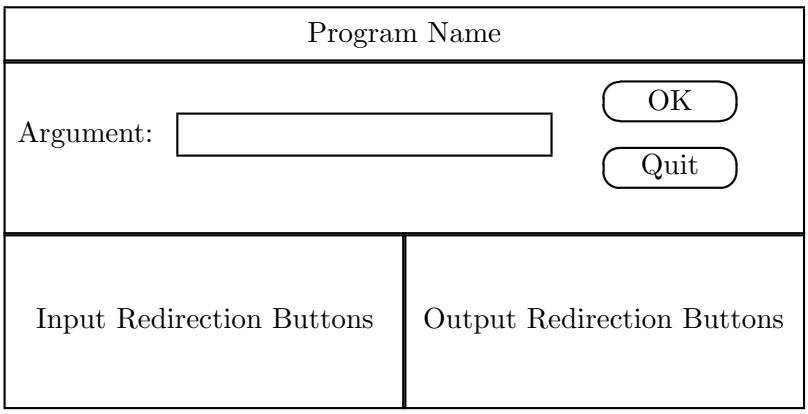

Figure 5.1: Macintosh Console Schematic

You can also double click on program icons in the file manager (Windows) or **Finder** (Macintosh). If in Windows, you will not be able to enter any command line options. The Macintosh versions will present a dialog box similar to Figure 5.1. You can enter command line parameters into the `Argument:` box and then click on the `OK` button to run the program. Do not click on the input/output redirection buttons. When double clicking on icons in either Windows or Macintosh, you will need to specify a working directory because the **qtlcart.rc** file will reside in the same place as the binaries.

5.1.2 Text Files

All input and output files in the ***QTL Cartographer*** system are plain text. They can be opened and viewed in any text editor or word processor. We suggest that you use a fixed-width font for viewing files. In addition, you should turn off automatic line-wrapping. Some of the output files produce tables of numbers: The fixed width font allows the numbers to line up in columns. Some of the tables are quite wide, and the line-wrapping destroys the columns. A good fixed width font available on most systems is **Courier**.

5.1.3 Gnuplot

Gnuplot is a freeware program to plot data. You will use **Preplot** to reformat your analytical results so that **Gnuplot** can create graphical results. Throughout the examples, we will refer to the program as **Gnuplot**, although it may have different names on different systems. For example, it is called **gnuplot 3.7.1c** on my Macintosh, while it comes as **WGNUPLOT.EXE** for Windows machines. Under UNIX, it will probably be called `gnuplot` and reside in **/usr/local/bin**.

Gnuplot will generally be run after **Preplot** and will need to load the plot control file created by **Preplot**. Let's suppose that the plot control file is **stem.plt**. If you are on a Macintosh, you will double click the **Gnuplot** binary and then use the "File" menu and "Open" to navigate and load **stem.plt**. This is similar to Windows except that you start up **WGNUPLOT.EXE**.

On a UNIX machine, you can load the plot control file from the command line:

```
% gnuplot stem.plt
```

or start up **Gnuplot**and load the file with load "sim.plt".

For all the following examples, we will simply say view the results using **Gnuplot**. If the plot control file has more than one graph, you will need to press return to see the next one in the sequence. Also, the only way to go back is to finish viewing all the graphs and reload the plot control file.

5.2 Basic Macintosh

The Macintosh operating system is so easy to use that little instruction is necessary. You might want a copy of **BBEdit Lite** for viewing and editing text files, although any good word processor using fixed-width fonts would suffice. **BBEdit Lite** is freeware, can open large files (as long as you have the memory) and allows you to view and convert text files with DOS, UNIX or Macintosh line endings. Other free programs such as **Fetch** to download files, **NiftyTelnet** to access UNIX servers and **Acrobat Reader** to view and print documents are also useful.

The Macintosh version of the distribution will be unpacked into a folder called **QTLCartMac** containing subfolders **bin, cbin, doc** and **example**. The **bin** folder contains programs compiled for PowerPC Macintoshes that don't have the ability to run Carbon applications. The **cbin** folder contains programs that require the Carbon libraries. If you are running MacOS 10 or higher, then you should be able to use the programs in **cbin**.

As mentioned above, each program will update the **qtlcart.rc** file. You can rename this file for later use if you want to work with different data sets. We suggest that you create a working folder to hold all your data, and that specific data files have their own subfolders of this working folder. For this tutorial, create a folder **qwork** within the **QTLCartMac** folder. Create subfolders in the **qwork** folder called **sim, mletest, realdat** and **sample**. When doing the example using the `mletest` data, set the working directory to `::qwork:mletest` using the menu the first program that you run and make sure that you have made copies of the `mletest` data files in the **mletest** subfolder.

5.3 Basic Windows

This is a quick summary of some basic commands and techniques for working in the Windows environment. There may be some differences between the various versions of Windows. We are assuming Windows NT version 4.0 for the following.

Logging in Using "control+alt+delete" will bring up the login screen. Click in the login box and type your login name. Press tab to get to the password box and type your password and a return. If you then see a timer (which looks like a little clock), you'll know you have succeeded: Just wait while the windowing system starts up.

Logging out You may want to empty the trash before you log out. Right-click on the recycle bin and select "empty" to do so. When you want to log out, simply click the left mouse button on the **Start** icon and select shutdown. This will bring up a menu. Select the "Close all programs and logon as a different user" option and then click "OK".

5.3.1 Navigating disks

We generally use **Windows Explorer** to navigate the disks. You can click on files, copy them and paste them in different directories to make copies of files. If you are not familiar with **Windows Explorer**, take a few minutes to play with it. You can double click on the "My Computer" icon and icons therein to explore your hard drive.

Viewing files

There are a lot of options for viewing files. Generally I recommend using **Notepad**. It is a simple text editor with a fixed width font. You can find it under "Start →Programs→ Accessories → Notepad". When you try to open a file, be sure to tell *Notepad* to look for files of type "All Files". If you don't, then *Notepad* will only show files with a ".txt" extension.

Windows NT does not like files to be accessed by two programs at once. Be sure to clear out **Notepad** by creating a new file before running any programs that might read or write to a file that you are viewing.

As with the Macintosh, you can use any word processor to view your files. It is always a good idea to use a fixed-width font and to turn off wrapping of lines.

Command Prompt

Clicking on "Start → Programs → Command Prompt" brings up a command line window for DOS commands. You can **ftp** or **telnet** from this window if you wish to transfer files or logon to an account elsewhere. There is a text editor that can be started with the command **edit** that will allow you to view files. Again, take care not to open files that are being accessed by other programs.

Transferring Files

You can start a *Command Prompt* and from there ftp files to your home account. You will need the IP number or hostname and domain name to do this. Simply start up the *Command Prompt*, type in the drive from which you want to transfer files, and **cd** to the directory where the files are. Then, **ftp** to your home machine and put the files there. Use `quit` to kill ftp and `exit` to get back to Windows. Here is an example.

```
c:\>  k:
k:\>  cd module5
k:\module5> ftp mymachine.somedomain.net
ftp> prompt
ftp> mput *
```

```
ftp> quit
k:\module5> exit
```

In the above example, the **k:** drive is the users home directory at North Carolina State University. If you are not at NCSU, then you may or may not have the **k:** drive.

The Windows version of the distribution will be unpacked into a subdirectory called **QTLCartWin** containing subdirectories **bin, doc** and **example**. The **bin** folder contains the programs and the example files are in the **example** subdirectory.

As mentioned above, each program will update the **qtlcart.rc** file. You can rename this file for later use if you want to work with different data sets. We suggest that you create a working directory to hold all your data, and that specific data files have their own subdirectories of this working directory. For this tutorial, create a directory **qwork** within the **QTLCartWin** directory. Create subsubdirectories in the **qwork** directory called **sim, mletest, realdat** and **sample**. When doing the example using the `mletest` data, set the working directory to **..\qwork\mletest** using the menu the first program that you run and make sure that you have made copies of the `mletest` data files in the **mletest** subdirectory.

5.4 Basic Unix

Keep in mind that Unix is case sensitive. Feel free to practice any of the following commands (but be careful with **rm** and **mv**).

5.4.1 Help!

The **man** command is one of the most important for the novice and experienced user. If you would like to know what it does, type **man man** at the prompt in a command window. You can use it to get information on most of the commands below.

5.4.2 Basic filesystem commands

Here is a list of basic commands for viewing, copying and moving the files in your directory, creating new subdirectories and navigating. Go ahead and experiment with these commands.

- **ls** is a command to list the files in the present working directory. You can give it options, for example `ls -l` will give listings with more information about the files than `ls`.
- **pwd** tells you where you are. This can be useful if you have created many subdirectories.
- **cd** allows you to change the current working directory. You can give it an absolute or a relative argument. `cd ..` would move you to the next highest subdirectory. `cd /ncsu/pams046/bin` would move you to the the /ncsu/pams046/bin subdirectory, *etc.*
- **mkdir** allows you to create a subdirectory. `mkdir test` would create the subdirectory *test.* `rmdir test` would remove it. You can only remove empty subdirectories.

- **rm** allows you to remove a file. It is aliased as **rm -i**, which means that it will ask if you really want to remove the file. `rm test.log` would remove the file *test.log.*
- **mv** moves a file. `mv file.orig file.new` would move the file **file.orig** to **file.new.** You can think of it as renaming.
- *cp* copies one or more files. `cp file1 file2` copies the file **file1** to the file **file2**. `cp file1 file2 direct` would copy the files **file1 file2** to the directory **direct.**
- **chmod** is a rather complex command to change the permissions on files. You can write batch files, and use **chmod** to allow execution of them.
- **more** will display the contents of a file. Use it as `more filename`. While in **more**, typing a `q` will get you out. On some systems, the command **less** is a more feature-rich replacement for **more**.

5.4.3 Other commands

- **ssh, rlogin, telnet** and **ftp** allow you to initiate sessions on other machines. You need to supply the IP address or nickname of the machine with these commands..
- **exit** closes a terminal window and **clear** clears it.
- **history** shows the last 40 commands issued. They will be numbered, and you can rerun them with an exclamation point and the number of the command, *e.g.,* `!23` would run the command numbered 23 in the *history* list.
- **lpr** sends a file to the printer.
- **alias** allows you to assign Unix commands to more familar words. For example, `alias dir ls` would allow you to type `dir` to list the files in a directory. `alias` with no arguments would list the current aliased commands.

5.5 Simulating and Analyzing data

Assuming that you have a **qwork** subdirectory (folder), create a new subdirectory (folder) within it. You can call it anything you like, but for the purposes of illustration it will be referred to as **sim**. Thus, your working directory for this example will be **::qwork:sim** if on a Macintosh or **..\qwork\sim** if on a PC. If you are on a UNIX machine, cd into the **qwork/sim** subdirectory and don't worry about setting a working subdirectory. Also note that if you are using a PC, the program names will all have an ".exe" ending, while for the Macintosh, you will double click the program icon.

In this exercise, you will simulate a genetic linkage map, then a model and finally a data set. This data will then be analyzed.

1. Start up **Rmap**. Select the option to change the filename stem. Change the filename stem to "sim". Now select the option to change the working directory and set it as suggested above. After this, you can change any parameters that you like. We suggest changing the variances of markers per chromosome and intermarker distance to values other than 0.0. In each case, a value of 2 or 3 would work well for the purposes of this exercise. Don't change the output format. When satisfied with the parameter values, select "0" to run the program. Look at the output (**sim.map**).

2. Start up **Rqtl**. You probably don't need to change any parameters. You can run this program with the "0" option. Look at the output (**sim.qtl**).

3. Start up **Rcross**. Again, you do not need to change any parameters, but you could try a different experimental design. Select the number associated with the experimental design. Change its value from "B1" to "SF3" (or whatever you like from Table 1.1). Run this program with the "0" option. Look at the output (**sim.cro**). From this point on, the analyses will utilize this file and the **sim.map** file.

4. Start up **Qstats**. and run it without changing any parameters. Look at the output (**sim.qst**).

5. Start up **LRmapqtl** and run it without changing any parameters. Look at the output (**sim.lr**).

6. Start up **SRmapqtl**. Run it and look at the output (**sim.sr**).

7. Start up **Zmapqtl**. You won't need to change any parameters. Tell it to go ahead with the analysis and look at the output (**sim.z**).

8. Start up **Zmapqtl**again. This time, choose "Model for Analysis" and change it to "6". Tell it to go ahead with the analysis, and look at the output (which will be appended to what you did in the first run). Model 6 is composite interval mapping, and **Zmapqtl**will use the ranked markers from the **SRmapqtl**run to determine which cofactors to use.

9. Start up **Preplot**. Don't change any parameters: Go ahead with the program.

10. View the results using **Gnuplot**.

11. Start up **Eqtl**. Go ahead with the analysis. Look at the output (**sim.eqt**).

On a UNIX machine, the above can be accomplished with the following series of commands:

```
% mkdir qwork
% cd qwork
% mkdir sim
% cd sim
% Rmap -A -X sim -vm 2 -vd 3
% Rqtl -A
% Rcross -A  -c SF2
```

```
% Qstats -A
% LRmapqtl -A
% SRmapqtl -A
% Zmapqtl -A
% Zmapqtl -A -M 6
% Preplot -A
% gnuplot sim.plt
% Eqtl -A
```

5.6 Analyzing simulated data

Create a working subdirectory (call it **mletest**) and copy the simulated data sets into it. The simulated datasets called **mletest.map** and **mletest.cro** come from Zeng (1994). They are in the **example** subdirectory of the distributions. These are properly formatted, and can be analyzed with **Qstats**, **LRmapqtl**, *etc.* Do the following:

1. Proceed with the analysis programs as in the previous example. Be sure to set the proper filename stem (mletest) and working subdirectory. Run **Qstats**, **LRmapqtl**, **SRmapqtl** and **Zmapqtl**. Look at the output after each run.
2. Start up **Preplot**. Don't change any parameters: Go ahead with the program.
3. View the results using **Gnuplot**.
4. Start up **Eqtl**. Go ahead with the analysis. Look at the output (**mletest.eqt**).

Assuming that the **QTLCartUnix.tar.gz** file was uncompressed and untarred in your home directory, the UNIX equivalents of the above are

```
% cd qwork
% mkdir mletest
% cd mletest
% cp ~/QTLCartUnix/example/mletest.* .
% Qstats -A -X mletest
% LRmapqtl -A
% SRmapqtl -A
% Zmapqtl -A
% Zmapqtl -A -M 6
% Preplot -A
% gnuplot mletest.plt
% Eqtl -A
```

5.7 Analyzing real data

Create a new working subdirectory called **realdat** in you **qwork** subdirectory. Copy the **realdat*.inp** files into it. There should be two files: **realdatm.inp** and **realdatc.inp**. The

former is a genetic linkage map in the standard input format (*map.inp*). The latter is a file with marker and trait data in the standard input format (*cross.inp*). This is a real data set kindly provided by Juan Medrano (Horvat and Medrano 1995). It has also been used as an example in a review on the statistical issues in QTL mapping (Doerge, Zeng, and Weir 1997). You will now translate the data files into the ***QTL Cartographer*** format, and then analyze the data.

1. Start up **Rmap**. Change the working subdirectory, and then the filename stem. You can use "realdat" for the stem. Now, select item 1 from the menu and enter `realdatm.inp`. Now run the program. **Rmap** should read in the prepared genetic linkage map file and reformat it properly.
2. Start up **Rcross**. Select item 1 from the menu and enter `realdatc.inp`. Now run the program. **Rcross** should read in the prepared data file, match marker names from this data file to those in the map file, and reformat the data properly. Look at the output.
3. Proceed with the analysis programs as in the previous examples. Run **Qstats**, **LRmapqtl**, **SRmapqtl** and **Zmapqtl**. Look at the output after each run.
4. Start up **Preplot**. Don't change any parameters: Go ahead with the program.
5. View the results using **Gnuplot**.
6. Start up **Eqtl**. Go ahead with the analysis. Look at the output (**realdat.eqt**).

Again we assume that **QTLCartUnix/example** is based in your home directory. The UNIX equivalents of the above are

```
% cd qwork
% mkdir realdat
% cd realdat
% cp ~/QTLCartUnix/example/realdat* .
% Rmap -A -i realdatm.inp -X realdat
% Rcross -A -i realdatc.inp
% Qstats -A
% LRmapqtl -A
% SRmapqtl -A
% Zmapqtl -A
% Zmapqtl -A -M 6
% Preplot -A
% gnuplot mletest.plt
% Eqtl -A
```

5.8 Analyzing a MAPMAKER data set

5.8.1 Using MAPMAKER/EXP

You will need **Mapmaker/EXP** for this part. If you don't want to use **Mapmaker/EXP**, then you can use the already prepared files that come with the distribution. Otherwise, **ftp**

to *genome.wi.mit.edu* and **cd** to **distribution/mapmaker** to get the programs. A file **sample.raw** comes with **Mapmaker/EXP**.

Each number is a command in a sequence to be done in **Mapmaker/EXP**. Anything inside of square braces are comments and should not be typed into **Mapmaker/EXP**. Start up **Mapmaker/EXP** in an appropriate subdirectory and proceed with these commands:

1. prepare data sample.raw [Input the data from the raw file.]
2. photo sample.tutorial [Save what you do in a log file.]
3. sequence 1 2 3 4 5 6 7 8 9 10 11 12 [Start with all markers.]
4. group [Group them into linkage groups]
5. sequence { 1 2 3 5 7 } [Use randomly ordered group 1 makers.]
6. compare [Compare all orders. For each in turn, calculate the Likelihood.]
7. sequence 1 3 2 5 7 [Decide that this is the best order and specify it.]
8. map [Print the map to the screen. This attaches distances as well.]
9. sequence 4 6 8 9 10 11 12 [Now use the rest of the markers.]
10. list loci [Summarize the number of informative progeny.]
11. lod table [Show pairwise distances and linkage LOD scores.]
12. sequence {8 9 10 11 12} [Use a randomly ordered subset of markers from group 2.]
13. compare [Compare all orders. For each in turn, calculate the Likelihood.]
14. sequence order1 [Use the best order from the compare command.]
15. try 4 6 [Try all possible positions of markers 4 and 6. Also, try unlinked idea.]
16. sequence 4 11 8 12 9 6 10 [This is the best sequence.]
17. make chromosome c1 [Create chromosome 1.]
18. sequence 1 3 2 5 7 [Specify the sequence of markers on chromosome 1.]
19. attach c1 [Attach the sequence to chromosome 1.]
20. framework c1 [Create the framework (puts in distances) for chromosome 2.]
21. make chromosome c2 [Create chromosome 2.]
22. sequence 4 11 8 12 9 6 10 [Specify the sequence of markers on chromosome 2.]
23. attach c2 [Attach the sequence to chromosome 2.]

24. framework c2 [Create the framework (puts in distances) for chromosome 2.]
25. quit [Exit the program. The map will be in sample.maps.]

On a UNIX machine, you will now have a file called **sample.maps**. On a PC, it will be called **sample.map**. It will be one of these two on a Macintosh. Rename this output file to **sample.mps**, and use it along with the **sample.raw** file for the next part.

5.8.2 Using the MAPMAKER files

Create a new working subdirectory called **mm** in your **qwork** subdirectory. Copy the files **sample.mps** and **sample.raw** into it. The former is a genetic linkage map created by **Mapmaker/EXP**. The latter is **Mapmaker/QTL** raw file. You will now translate the data files into the ***QTL Cartographer*** format, and then analyze the data.

1. Start up **Rmap**. Select the option to change the filename stem. Change the filename stem to "sample" and set the proper working subdirectory. Then select the input file option and change it to `sample.mps`. Then go ahead with the analysis. Look at the output (**sample.map**).
2. Start up **Rcross**. Select the input file option and change it to `sample.raw`. Then go ahead with the analysis. Look at the output (**sample.cro**).
3. Proceed with the analysis programs as in the previous examples. Run **Qstats**, **LRmapqtl**, **SRmapqtl** and **Zmapqtl**. Look at the output after each run.
4. Start up **Preplot**. Don't change any parameters: Go ahead with the program.
5. View the results using**Gnuplot**.
6. Start up **Eqtl**. Go ahead with the analysis. Look at the output (**sample.eqt**).

The sample files should be in the usual place (**QTLCartUnix/example**). The UNIX equivalents of the above are

```
% cd qwork
% mkdir mm
% cd mm
% cp ~/QTLCartUnix/example/sample.* .
% Rmap -A -i sample.mps -X sample
% Rcross -A -i sample.raw
% Qstats -A
% LRmapqtl -A
% SRmapqtl -A
% Zmapqtl -A
% Zmapqtl -A -M 6
% Preplot -A
% gnuplot sample.plt
% Eqtl -A
```

5.9 Multiple Interval Mapping

This section is generally done during the QTL II module at the Summer Institute in Statistical Genetics. This lab will usually start with a twenty minute introduction by the instructor. Since the first part of the exercise takes about twenty minutes to run, it is a good idea to start it before, and let it run during, the introductory lecture.

We will work with the files **mletest.map** and **mletest.cro** for this section. We are going to analyze the data in three ways. The first will be to use **MImapqtl** to find a model *de novo*. Then, we will use composite interval mapping to find an initial model for multiple interval mapping. Finally, we will use stepwise regression to determine an initial model and multiple interval mapping to complete the analysis.

Create a new subdirectory called **mletest2** in your **qwork** subdirectory. Within that subdirectory, create three new subdirectories called **ci**, **mi** and **mr**. Place copies of the data files (**mletest.map** and **mletest.cro**) in each of these directories. In UNIX, you would do this:

```
% cd qwork
% mkdir mletest2
% cd mletest2
% mkdir ci
% mkdir mi
% mkdir mr
% cp ~/QTLCartUnix/example/mletest.* ci
% cp ~/QTLCartUnix/example/mletest.* mi
% cp ~/QTLCartUnix/example/mletest.* mr
```

5.9.1 Multiple interval mapping from scratch

Change into the **mi** subdirectory. Run **MImapqtl** as follows:

```
% cd mi
% MImapqtl -A -V -X mletest -p 1
```

When finished, note how much time it took for this run (you will need to look at the end of the **mletest.log** file. Also, look at the file *mletestPhase1.mqt* and compare the QTLs found to the true model, which is in the file *mletest.qtl*. How many of the true QTL were found by **MImapqtl**?

5.9.2 Composite interval mapping preceeds multiple interval mapping

Change into the **ci** subdirectory. Run the following set of commands:

```
% SRmapqtl -A -V -X mletest
% Zmapqtl -A -V -M 6
% Eqtl -A -V -S 12.0
% cp mletest.eqt mletestPhase0.mqt
% MImapqtl -A -V -I sMPrTseC -p 1
% MImapqtl -A -V -I sMPrtSeC
```

Compare the **mletestPhase*.mqt** files to the true model in **mletest.qtl**. The first is the model as estimated from composite interval mapping. The second is a reduced model in which **MImapqtl** has deleted non-significant QTL. The third (*mletestPhase2.mqt*) contains a model after **MImapqtl** has searched for more QTL. Is this final model closer to the true model than the model obtained by the analysis in the previous section? How long did this whole process take?

5.9.3 Multiple regression preceeds multiple interval mapping

Change into the **mr** subdirectory. Run the following set of commands:

```
% JZmapqtl -A -V -X mletest -M 9 -I 10
% MultiRegress -A -V
% Rqtl -A -V -i mletest.mr -o mletestPhase0.mqt
% MImapqtl -A -V -I sMPrTseC -p 1
% MImapqtl -A -V -I sMPrtSeC
```

As in the previous example, compare the **mletestPhase*.mqt** files to the true model in **mletest.qtl**. The first is the model as estimated from multiple regression of the trait on genotypic expected values every two centimorgans. The second is a reduced model in which **MImapqtl** has deleted non-significant QTL. The third (**mletestPhase2.mqt**) contains a model after **MImapqtl** has searched for more QTL. Is this final model closer to the true model than the model obtained by *de novo* analysis? How long did this set of commands take?

5.9.4 Real Data

You can do the same set of examples with the `sample` and `realdat` data sets.

5.10 Multiple Trait Mapping

Describe the data set.

```
% Rmap -i rootmassm.inp -X rootmass -A
% Rcross -i rootmassc.inp -A
% Qstats -A
% LRmapqtl -t 5 -A
% SRmapqtl -A
% Zmapqtl -t 5 -A
% Zmapqtl -t 5 -M 6 -A
% JZmapqtl -t 5 -M 3 -I 14 -A
% Eqtl -S 10.0 -H 14 -A
% mkdir model3
% mv rootmass.z0 rootmass.z1 rootmass.z2 rootmass.z3 rootmass.z4 model3
% cp rootmass.sr model3
% vi rootmass.sr
```

```
% JZmapqtl -t 5 -M 6 -I 14 -A
% Eqtl -S 10.0 -H 14 -A
```

The first six steps should be familiar. Look at the output in each step.

Step seven invokes multitrait mapping with a test for GxE interactions.

Steps 9-12 back up your analysis using interval mapping and prepare for composite interval mapping.

Step 12 requires you to edit the **rootmass.sr** file. The example uses **vi**, by you can use any text editor. You should decrease the number of ranked markers in the **rootmass.sr** file by deleting some of the lines. I would suggest deleting all lines with rank greater than three.

Chapter 6

Input File Formats

All of the input and output files in the ***QTL Cartographer*** system are plain text, and can thus be viewed by virtually any text editor or word processor on any platform. The input files for many of the programs will have embedded commands that start with a minus sign (-). Care should be taken not to have stray tokens such as '-Chromosome' in input files. Also, the case of commands is generally very important: When in doubt use the exact case that is specified here.

6.1 Genetic Linkage Maps

6.1.1 MAPMAKER output files

Rmap can translate the output of **Mapmaker/EXP** into the format required by the ***QTL Cartographer*** system. Use the .maps file that is the output of **Mapmaker/EXP** as the input to **Rmap** and it will be translated automatically. An alternate format has been designed for those who don't have the **Mapmaker/EXP** files.

6.1.2 Rmap input files

The general method of inputting data for this format is by tokens. Tokens are just collections of characters surrounded by whitespace (spaces, carriage returns, tabs, line feeds). The maximum length of any token must be less than 64 (and this may be increased in the future). An example of this format is given in Figure 6.1.

First line

It is critical that the first line start with a pound symbol, a space, a number, the word `bychromosome`, a `-filetype` token, and the `map.inp` token as shown below:

```
# 123456789    bychromosome      -filetype map.inp
 ^          ^                ^            ^
space      space           space        space
```

```
# 123456789    bychromosome  -filetype map.inp
# Look for documentation at the end
-type positions
-function     1
-Units       cM
-chromosomes  2
-maximum      5
-named      yes
-start
-Chromosome 1
   Marker1_1    0.0
   Marker1_2   10.2
   Marker1_3   34.1
   Marker1_4   43.3
   Marker1_5   52.1
-Chromosome 2
   Marker2_1    0.0
   Marker2_2   13.7
   Marker2_3   19.1
   Marker2_4   24.8
-stop
-end
```

Figure 6.1: Input format for a map.inp file

The spaces are necessary, because the input is token based. The program will read the second token in the file as a long integer and use it as an identifier for the file. Thus, each file should have a unique identifier following the pound symbol.

The `-filetype map.inp` tells the ***QTL Cartographer*** programs that this is a `map.inp` formatted file. The `bychromosome` word is a vestige of an idea that simply needs to hang around.

Tokens

A token is any string of non-whitespace surrounded by whitespace. Each time a token begins with a minus sign (-), it is read to determine what follows it. The next token is then read and processed. The program will ignore any token that it doesn't recognize. The tokens that are recognized as commands in a `map.inp` file are `-type`, `-function`, `-param`, `-Units`, `-chromosomes`, `-maximum`, `-end`, `-named`, `-start`, `-stop`, `-skip`, `-unskip` and `-Chromosome`.

Note that only the first letter of these tokens is actually necessary (except for `-start`, `-stop` and `-skip`, which require more letters to distinguish between them).

Case is important. `-Units` is different from `-units`.

Meanings of commands

The file in Figure 6.1 also has commands embedded into it. **Rmap** recognizes any token that begins with a minus sign (-) as an embedded command. Some commands require that the following token be a number or piece of information. The following table gives a list of tokens that the program recognizes, their purpose and what the next token should be.

Commands	Followed by	Means
`-type`	positions	Marker positions are given
`-type`	intervals	Marker intervals are given
`-function`	integer 1-8	Code for the map function: Details in manual
`-param`	real number	Extra parameter needed for some map functions
`-Units`	cM, M or r	Units are in centiMorgans (cM), Morgans (M) or recombination frequencies (r)
`-chromosomes`	integer > 0	Haploid number of chromosomes
`-maximum`	integer > 0	Maximum number of markers on any chromosome. Needed for array allocation.
`-named`	yes or no	Says whether markers will have names.
`-skip`		Tells **Rmap** to skip everything until an
`-unskip`		which is the end of the `-skip`. These cannot be within a `-start` to `-stop` block
`-start`		Start of a map block
`-stop`		End of a map block
`-end`		End of the file
`-Chromosome`	integer, string	Gives the number of the chromosome (if integer) or the name (if a string)

Map block

Between the `-start` token and the `-stop` token, you should have a repeating sequence of a `-Chromosome` token followed by an integer or sting, then markers ordered, with their names followed by the appropriate distances. You should be consistent with the tokens following the `-Chromosome` token: Either have them all be unique chromosome names or unique integers from 1 to the number of chromosomes indicted with the `-chromosomes` token.

This example has the markers followed by their positions in centiMorgans. Please give all markers unique names.

```
-named      yes
-start
-Chromosome 1
   Marker1_1    0.0
   Marker1_2   10.2
   Marker1_3   34.1
   Marker1_4   43.3
   Marker1_5   52.1
-Chromosome 2
   Marker2_1    0.0
   Marker2_2   13.7
   Marker2_3   19.1
   Marker2_4   24.8
-stop
```

In the above, the line `Marker1_2 10.2` means that Marker1_2 is on chromosome 1 at position 10.2 cM from the left telomere. If you want to give the chromosomes names, then the above could have been

```
-named      yes
-start
-Chromosome X
   Marker1_1    0.0
   Marker1_2   10.2
   Marker1_3   34.1
   Marker1_4   43.3
   Marker1_5   52.1
-Chromosome First
   Marker2_1    0.0
   Marker2_2   13.7
   Marker2_3   19.1
   Marker2_4   24.8
-stop
```

If `-named` had a value of `no` in the preamble, then the format of the distances would be:

```
-named     no
-start
-Chromosome 1      0.0 10.2 34.1 43.3 52.1
-Chromosome 2      0.0 13.7 19.1 24.8
-stop
```

That is, the names would be skipped.

You can also create a map where the markers are not on the telomeres. If you are inputting the map based on positions, then

```
-start
-Chromosome 1
   Marker1_1    5.0
   Marker1_2   10.2
   Marker1_3   34.1
   Marker1_4   43.3
   Marker1_5   52.1
   Telomere    55.3
-Chromosome 2
   Marker2_1    4.0
   Marker2_2   13.7
   Marker2_3   19.1
   Marker2_4   24.8
   Telomere    27.4
-stop
```

would mean that there is DNA outside of the first and last markers on a Chromosome. The keyword 'Telomere' is recognized by **Rmap** as a telomere and not as a marker.

If you are inputting intervals, then the same map would look like

```
-start
-Chromosome 1
   Telomere     5.0
   Marker1_1    5.2
   Marker1_2   23.9
   Marker1_3    9.2
   Marker1_4    8.8
   Marker1_5    3.2
-Chromosome 2
   Telomere     4.0
   Marker2_1    9.7
   Marker2_2    5.4
   Marker2_3    4.7
   Marker2_4    2.6
-stop
```

That is, you need to indicate that the telomeric DNA comes first. These two maps should give the same result when run through **Rmap**.

WARNING: do not use 'Telomere' as a marker name. This keyword is case sensitive, so 'telomere' is not the same as 'Telomere'.

Termination

The `-end` token will tell **Rmap** to stop reading the file and to the translation. The token `-quit` will do the same thing.

Annotation

You can annotate this file as much as you want. Just don't put in any extra stuff in the `-start` to `-stop` block. Everything after the `-end` token is ignored. Before the `-start` token, only the `-type, -function, -param, -units, -chromosomes` and `-maximum` tokens are processed. The token following each of these is read and the information used in the program.

The format of the information in `-start` to `-stop` block is unimportant. You just need whitespaces around each piece of information. All the marker names and their distances could be on one line.

Usage

If this file were called **map.inp**, then
`% Rmap -A -V -i map.inp`
would convert this file to the format required for the other programs in the ***QTL Cartographer*** system.

6.1.3 Rmap output files

Rmap overwrites any file that has the same name as specified as the output file. Be careful not to destroy any important files. The output file will contain the values of the parameters used, the names of chromosomes and markers (if a translation was made) and the linkage map.

6.2 QTL information

You can specify a genetic model and use it for simulation by translating it with **Rqtl**. This would be useful if you want to do some "what-if" experiments.

6.2.1 Rqtl input files

The input format is similar to that for **Rmap**. Figure 6.2 is an example of an input file for a QTL model.

First line

it is critical that the first line start with a pound symbol, a space, a number, a -filetype token and the qtls.inp filetype:

```
# 12345789    -filetype qtls.inp
 ^          ^           ^
space      space     space
```

The spaces are necessary, because the input is token based. The program will read the second token in the file as a long integer and use it as an identifier for the file. Thus, each file should have a unique identifier following the pound symbol.

The `-filetype qtls.inp` tells the QTL Cartographer programs that this is a `qtls.inp` formatted file.

Tokens

A token is any string of non-whitespace surrounded by whitespace. Each time a token begins with a -, it is read to determine what follows it. The next token is then read and processed. The program will ignore any token that it doesn't recognize. The tokens that are recognized in a *qtls.inp* file are `-Units, -end, -named, -start, -stop, -skip, -unskip, -interactions` and `-filetype`.

Usually, only the first letter of these tokens is actually necessary (except for `-start, -stop` and `-skip`, which require more letters to distinguish between them).

Case is important. `-Units` is correct and is different from `-units`, which will be ignored.

Meanings of commands

The file in Figure 6.2 also has commands embedded into it. **Rqtl** recognizes any token that begins with a minus sign (-) as an embedded command. Some commands require that the following token be a number or piece of information. The following table gives a list of tokens that the program recognizes, their purpose and what the next token should be.

```
#   12345789    -filetype qtls.inp       # Documentation at end
-Units       cM
-named      yes

-start qtls 3
Trait_1 4
    1   9.1  0.75  0.0
    1  89.1  0.5   0.0
    3  68.4  0.22  0.0
    4  43.2  0.95  0.0
Trait_2 2
    2  93.4  0.42  0.0
    4  33.2  0.90  0.0
Trait_3 1
    1  33.4  0.84  0.2
-stop qtls

-interactions
-trait Trait_1
1 2  AA   0.01
1 2  AD   0.02
1 2  DA   0.031
1 2  DD   0.04
1 3  AA   0.01
1 3  AD   0.02
1 3  DA   0.032
1 3  DD   0.05
2 3  AA   0.01
2 3  AD   0.02
2 3  DA   0.03
2 3  DD   0.0
-trait Trait_2
1 2  AA   0.01
1 2  AD   0.02
1 2  DA   0.03
1 2  DD   0.04
-trait Trait_3
-stop interactions
-end
```

Figure 6.2: Example of a Model input file

Command	Followed by	Means
`-filetype`	qtls.inp	Tells **Rqtl** what to expect in the format.
`-Units`	cM or M	cM for centiMorgans and M for Morgans.
`-named`	yes or no	Answers whether the traits will have names.
`-start`	qtls	Begin data defining QTLs
`-interactions`		Begin data on defining interactions
`-skip`		Tells **Rqtl** to skip everything until an
`-unskip`		which is the end of the -skip. These cannot be within a -start to -stop block or a -interactions to -stop block
`-stop`		End of a qtls or interactions block
`-end`		End of the file

QTLs block

After the `-start` token, there should be the token `qtls` and a number to indicate the number of traits to be modelled. For example, `-start qtls 3` means that there will be QTLs for 3 traits. After this there should be a repeating sequence of a trait name, number of loci for that trait, then the chromosome, position, additive and dominance effects for each locus. This example has the loci followed by their positions in centiMorgans (from the telomere). Please give all traits unique names.

Here is an example of the repeating sequence:

```
Trait_Name   #loci
 chrom pos   a     d
 chrom pos   a     d
   .    .    .     .
   .    .    .     .
   .    .    .     .
 chrom pos   a     d
Trait_Name   #loci
 chrom pos   a     d
 chrom pos   a     d
   .    .    .     .
   .    .    .     .
   .    .    .     .
 chrom pos   a     d
```

Our specific example was

```
-start qtls 3
Trait_1 4
    1   9.1  0.75  0.0
    1  89.1  0.5   0.0
    3  68.4  0.22  0.0
```

```
    4  43.2  0.95  0.0
Trait_2 2
    2  93.4  0.42  0.0
    4  33.2  0.90  0.0
Trait_3 1
    1  33.4  0.84  0.2
-stop qtls
```

Now, consider the block

```
Trait_1 4
    1   9.1  0.75  0.0
    1  89.1  0.5   0.0
    3  68.4  0.22  0.0
    4  43.2  0.95  0.0
```

and think of them as being numbered consecutively:

```
Trait_1 4
1.    1   9.1  0.75  0.0
2.    1  89.1  0.5   0.0
3.    3  68.4  0.22  0.0
4.    4  43.2  0.95  0.0
```

and for the second trait, the numbering starts from 1 again:

```
Trait_2 2
1.    2  93.4  0.42  0.0
2.    4  33.2  0.90  0.0
```

These numberings will be used to define the interactions.

Interactions

For a QTL with k loci, there are $k(k-1)/2$ unordered pairs of loci. We can define epistatic interactions for each of these pairs. Epistatic interactions come in four types, thus there are $2k(k-1)$ possible interactions. The abbreviations for the types are:

	Interaction	Abbrev.
1.	Additive by additive	AA
2.	Additive by dominance	AD
3.	Dominance by additive	DA
4.	Dominance by dominance	DD

If an interaction is not defined in the `-interactions` block, then it will be set to zero. To define the interactions, we have a block that looks similar to that defining QTL main effects. The token `-interactions` is followed by the `-trait` token with a traitname and then a repeating

sequence of `QTL1 QTL2 Type Value` tokens. Thus, the line
`1 2 AA 0.01`
means that QTL 1 (chromosome 1 position 9.1) and QTL 2 (chromosome 1 position 89.1) have an additive by additive effect of 0.01. Note that the QTL with a smaller number ALWAYS comes before that with the larger number. Note that in the example file, there were no interactions between QTL 4 and any of the other QTL for the first trait. If an interaction term is missing, its value is zero.

Termination

The `-end` token will tell **Rqtl** to stop reading the file and to the translation. The token `-quit` will do the same thing.

Annotation

You can annotate the input file as much as you want. Be careful to not put annotation in the `-start` to `-stop` or `-interactions` to `-stop` blocks. Also, don't use words that begin with a minus sign, or put such words in a `-skip` to `-unskip` block. Finally, anything after an `-end` or `-quit` token will be ignored.

Usage

If this file were called `qtls.inp`, then
`% Rqtl -A -V -i qtls.inp`
would convert this file to the format required for the other programs in the ***QTL Cartographer*** system. The above assumes that an appropriate map file exists.

6.2.2 Rqtl output files

Rqtl overwrites any file that has the same name as specified as the output file. Be careful not to destroy any important files. The output file will contain the genetic model in a format suitable for input into **Rcross**.

6.3 Data files

These are files that contain marker and trait data. The output format of **Rcross** is rather difficult for the user to read and create manually. We have therefore provided ways to translate other formats.

6.3.1 MAPMAKER raw files

Rcross will convert **Mapmaker/QTL** raw files for use in the ***QTL Cartographer*** system. You will first need to use **Mapmaker/EXP** to create a genetic linkage map. Then convert the map into the "Rmap.out" format for use with **Rmap**. Then, use **Rcross** to convert the **Mapmaker/QTL** raw data file into the "Rcross.out" format.

6.3.2 Rcross input files

We have also defined a format for your data. It is similar to the input formats for **Rmap** and **Rqtl**. Input is token based, and the data file has embedded commands to indicate to **Rcross** what it is reading. An example of such a file is given in Figure 6.3.

First line

it is critical that the first line start with a pound symbol, a space, a number, a `-filetype` token, and the cross.inp filetype:

```
# 123456789    -filetype   cross.inp
 ^           ^            ^
space      space       space
```

The spaces are necessary, because the input is token based. The program will read the second token in the file as a long integer and use it as an identifier for the file. Thus, each file should have a unique identifier following the pound symbol.

The `-filetype cross.inp` tells the QTL Cartographer programs that this is a `cross.inp` formatted file.

Tokens

A token is any string of non-whitespace surrounded by whitespace. Each time a token begins with a minus sign (-), it is assumed to be a command and the token following it is processed. The program will ignore any token that it doesn't recognize. The tokens that are recognized as commands in a `cross.inp` file are `-traits, -SampleSize, -TranslationTable, -end, -Cross, -case, -skip, -unskip, -quit` and `-start`.

Note that only the first letter of these tokens is actually necessary (except for `-start`, `-stop` and `-skip`, which require more letters to distinguish between them).

Case is important. `-Units` is different from `-units`.

Extra whitespace is ignored, so you can put in spaces, tabs, carriage returns, etc, to format your data in an understandable fashion.

```
# 123456787        -filetype cross.inp
# Documentation at the end
-Cross       B1
-traits       2
-otraits      2
-SampleSize  10
-case        no
-TranslationTable
     AA    2    2
     Aa    1    1
     aa    0    0
     A-   12   12
     a-   10   10
     --   -1   -1
-start markers
Marker1_1 2 2 2 2 2 1 1 1 1 1
Marker1_2 2 2 2 1 1 1 1 1 1 1
Marker1_3 1 2 2 2 2 1 1 1 1 1
Marker1_4 1 1 2 2 2 1 1 1 1 1
Marker1_5 2 2 2 2 2 1 1 1 1 1
Marker2_1 2 1 1 2 2 1 1 1 1 1
Marker2_2 2 2 2 1 1 1 1 1 1 1
Marker2_3 2 2 1 1 1 1 2 2 2 2
Marker2_4 2 1 1 1 1 1 1 1 2 2
-stop markers

-missingtrait .

-start traits
Trait_1    5.0  5.3  6.2  4.1  5.5  5.8  6.7  6.1   .   6.4
Trait_2   15.0 15.3 16.2 24.1 25.5 25.8 16.7 26.1 33.2 16.4
-stop traits

-start otraits
Sex   M F M M M F F M F F
Brood 1 1 1 1 1 2 2 2 2 2
-stop otraits
-quit
```

Figure 6.3: Example of a cross.inp file

Meanings of commands

Token	Followed by	Means
`-Cross`	string	type of cross See (1.1.2)
`-traits`	integer	number of traits
`-otraits`	integer	number of categorical variables
`-SampleSize`	integer	sample size
`-case`	yes or no	whether comparisons are case sensitive
`-TranslationTable`	table	table for marker translation
`-skip`		Tells Rcross to skip everything until an
`-unskip`		which is the end of the -skip. These
		cannot be within a -start to -stop block
`-start`	string	Start of a block of markers, traits or otraits
`-stop`		End of a block
`-end`		End of the file

The `-case` token changes how strings are matched. Put a `yes` here if comparisons are case dependant. With `no`, all names of individuals, markers and traits are converted to lower case to make comparisons.

Marker Translation

You can define how markers are translated. Here is the default translation table

```
-TranslationTable
    AA    2    2
    Aa    1    1
    aa    0    0
    A-   12   12
    a-   10   10
    --   -1   -1
```

Note a few things in the above translation table. There are six rows and three columns. There must be a token in all 18 positions of the table. The first column is the genotype. The program assumes that the A allele is diagnostic for the High (parental 1) line and the a allele is diagnostic for the Low (parental 2) line. A minus sign (-) means the allele is unkown. Dominant as well as codominant markers can be encoded. The middle column is how the output of these genotypes will be encoded while the right (3rd) column is how you will code the input of this file. The above `TranslationTable` maps 2 to 2, 1 to 1, 0 to 0, etc. Just about any set of tokens can be used for the third column, but DO NOT change the first two columns. If you encoded your P_1 homozygotes as BB, heterozygotes as Bb, etc, your translation table might appear as

```
-TranslationTable
    AA    2   BB
    Aa    1   Bb
```

```
aa    0   bb
A-   12   B-
a-   10   b-
--   -1   --
```

Anything in the following data file that is not recognized (doesn't match something in column 3) will become unkown (-1) in the output.

REMEMBER: You need all 18 tokens following the `-TranslationTable` command and the first two columns can't be altered. You can only alter the last column.

Crosses

You need to define the type of cross. See (1.1.2) for more on the different crosses and the symbols used by the ***QTL Cartographer*** system.

Data Input by Markers and Trait

One way to organize the data is by markers. For each marker, you give the genotypes of the individuals. The order of the individuals has to be the same for each marker. Below is an example. After the `-start markers`, the program expects a repeating sequence of marker name, then n marker genotypes where n is the sample size. The marker names should match those in the Map file.

```
-start markers
Marker1_1 2 2 2 2 2 1 1 1 1 1
Marker1_2 2 2 2 1 1 1 1 1 1 1
Marker1_3 1 2 2 2 2 1 1 1 1 1
Marker1_4 1 1 2 2 2 1 1 1 1 1
Marker1_5 2 2 2 2 2 1 1 1 1 1
Marker2_1 2 1 1 2 2 1 1 1 1 1
Marker2_2 2 2 2 1 1 1 1 1 1 1
Marker2_3 2 2 1 1 1 1 2 2 2 2
Marker2_4 2 1 1 1 1 1 1 1 2 2
-stop markers
```

The traits are encoded in the same fashion. After the `-start traits` tokens, the program expects a repeating sequence of trait name and then n values for the sample. The order of the individuals has to be the same as in the markers. Also, you can specify the token for a missing trait with
`-missingtrait .`
If you want a lone period to indicate a missing datum.

```
-start traits
Trait_1    5.0  5.3  6.2  4.1  5.5  5.8  6.7  6.1   .   6.4
Trait_2   15.0 15.3 16.2 24.1 25.5 25.8 16.7 26.1 33.2 16.4
-stop traits ... indicates the end of the trait data.
```

Categorical (otraits) are other traits that will be stored as character strings. These will be things such as sex, brood, eye color, etc. Each token should be less than 64 characters in length. The `-missingtrait` token can be reused with a different token if desired.

```
-start otraits
Sex   M F M M M F F M F F
Brood 1 1 1 1 1 2 2 2 2 2
-stop otraits
```

Data Input by individuals

If the `-start` and `-stop` tokens are followed by the work 'individuals', then data will be read in by individuals.

Another way to organize the data is by individuals. The program expects that the markers are ordered from marker 1 on chromosome 1, marker 2 on chromosome 1, ..., to the last marker on the last chromosome. Since the individuals are named, they can be in any order.

```
-start individuals markers
Ind_1  2 2 1 1 2 2 2 2 2
Ind_2  2 2 2 1 2 1 2 2 1
Ind_3  2 2 2 2 2 1 2 1 1
Ind_4  2 1 2 2 2 2 1 1 1
Ind_5  2 1 2 2 2 2 1 1 1
Ind_6  1 1 1 1 1 1 1 1 1
Ind_7  1 1 1 1 1 1 1 2 1
Ind_8  1 1 1 1 1 1 1 2 1
Ind_9  1 1 1 1 1 1 1 2 2
Ind_10 1 1 1 1 1 1 1 2 2
-stop individuals markers
```

The traits are done similarly. All the traits have to be in this block. Each column is for a different trait. After the `-start` token, put individuals followed by traits, then the number of traits (2), then the names of the traits, then indicate whether the individuals are named. Here they are, but if they weren't, put an `notnamed` token where the 'named' token presently is. `otraits` below are done in the same fashion.

```
-start individuals traits 2 Trait_1 Trait_2 named
Ind_1  5.0 15.0
Ind_2  5.3 15.3
Ind_3  6.2 16.2
Ind_4  4.1 24.1
Ind_5  5.5 25.5
Ind_6  5.8 25.8
Ind_7  6.7 16.7
Ind_8  6.1 26.1
```

```
Ind_9   .  33.2
Ind_10 6.4 16.4
-stop individuals traits

-start individuals otraits 2 sex brood named
Ind_1   M 1
Ind_2   F 1
Ind_3   M 1
Ind_4   M 1
Ind_5   M 1
Ind_6   F 2
Ind_7   F 2
Ind_8   M 2
Ind_9   F 2
Ind_10  F 2
-stop  individuals otraits
```

Termination

The `-end` token will tell **Rcross** to stop reading the file and to the translation. The token `-quit` will do the same thing.

Annotation

You can annotate the input file as much as you want. Be careful to not put annotation in the `-start` to `-stop` blocks. Also, don't use words that begin with a minus sign, or put such words in a `-skip` to `-unskip` block. Finally, anything after an `-end` or `-quit` token will be ignored.

Usage

If this file were called `cross.inp`, then
`% Rcross -i cross.inp`
would convert this file to the format required for the other programs in the ***QTL Cartographer*** system. The above assumes that an appropriate map file exists.

Chapter 7

Benchmarks

Beginning with version 1.17 of ***QTL Cartographer***, we will only present timings for **MImapqtl**, since this is the most compute intensive program in the suite. For our data, we use the simulated data set (Zeng 1994) based on a genetic linkage map that has four chromosomes with 16 markers on each chromosome. The markers are evenly spaced at 10 cM and the simulated data has one trait. The entire genome was scanned at a walking speed of 2 cM. The programs were run in automatic mode, with no recourse to the interactive menus. They indicate the amount of time to read in the data, perform the analysis and write the output. Timings are presented in Table 7.1.

The analytic results were the same on all platforms. **MImapqtl** is sensitive to the number of parameters in the model it is analyzing. In this example, 10 QTL were identified. the first stages of QTL identification proceed quickly, but the program slows down significantly as more parameters are used. This is something to keep in mind if you decide to search for epistatic terms. If there are a lot of main effects, then a backward elimination method for finding epistatic interactions may take a prohibitavely long time.

With verson 1.17g of ***QTL Cartographer***, one can compile the programs to use single or double precision floating point arithmatic. In addition, we can compile for 32 or 64 bit binaries under Sun Solaris with the SUNWspro compilers.

Platform	Compiler	FPN	Time (seconds)
Macintosh G5	gcc 3.3 -fast	double	335
1.8GHz	gcc 3.3 -fast	float	371
Panther 10.3.3	Codewarrior 8	double	600
	Codewarrior 8	float	359
Dell Optiplex	gcc 3.3 -O3	double	340
3GHz, Pentium IV	gcc 3.3 -O3	float	291
Windows XP	Codewarrior 8	double	342
	Codewarrior 8	float	304
Dell 3GHz PIV	gcc 3.3 -O3	double	547
Redhat Linux	gcc 3.3 -O3	float	539
SunBlade 1000	gcc 2.95 -O3	double	1373
0.75GHz UltraSparc	gcc 2.95 -O3	float	947
Solaris 9	SUNWspro (32 bit)	double	1431
	SUNWspro (32 bit)	float	982
	SUNWspro (64 bit)	double	1301
	SUNWspro (64 bit)	float	1023

Table 7.1: Timings for Multiple Interval Mapping

Chapter 8

UNIX Man Pages

In the UNIX world, a standard way of providing online documentation of programs is to write man pages. These are ASCII text files with embedded troff commands. UNIX versions of ***QTL Cartographer*** have man pages for all the programs in the suite. On a UNIX system, if the man pages are in the correct subdirectory (in essence, if the subdirectory that contains the man pages is defined in the environmental variable MANPATH), then you can get the online help with a command such as

```
% man Rmap
```

We provide html versions of the man pages on the web server for Macintosh and Windows users. If you have World Wide Web access, first point your browser to our home page:

```
http://statgen.ncsu.edu/
```

Then, click on ***QTL Cartographer*** link under the **Software** menu. Follow it to the online man pages. You can also access the rest of the ***QTL Cartographer*** manual, the ftp server for the programs and other supplemental material. The manual is written in LaTeX2e and has been translated into HTML by the program **html2latex**.

The complete set of man pages are reprinted here for your benefit. Here follow the LaTeX formatted versions of the man pages.

Since the documentation will change regularly, it is a good idea to check the Web site for the current online manual. The web pages will always be updated with the manual updates

8.1 QTLCART

NAME

QTLcart — A rudimentary front end for the QTL Cartographer system.

SYNOPSIS

QTLcart [**-h**] [**-V**] [**-A**] [**-s** *seed*] [**-W** *workdir*] [**-X** *stem*] [**-e** *logfile*] [**-R** *resource*]

DESCRIPTION

QTLcart exists but does nothing at this time. It is intended to be the front end to a set of programs collectively known as QTL Cartographer. This man page explains the options that are valid in all the programs of the QTL Cartographer suite. It also outlines how to get started using the programs.

OPTIONS

The following options can be used with any of the programs in the QTL Cartographer suite.

-h Prints out the current values of all program options, and information on what the program does. It then exits.

-V Turns the verbosity mode off. The programs in the suite print out messages while running. This option turns off those messages. This is useful for batch files.

-A Skips the interactive screen for setting options. All programs start up with a menu that allows setting of options. This turns the menu off. It is also very useful for batch files.

-R The programs will read the default parameters from a file specified with this option. If a file called *qtlcart.rc* is in the current working directory, it will be opened by default and all parameter values read. If no such file exists, then default parameter values will be assumed, and the file will be created.

It is probably better to simply rename a resource file *qtlcart.rc* than to use this option.

-W This option allows one to set the work directory. This directory must exist. All the input files must be in this directory and the output files will be placed there.

-s This requires a long integer to act as the random number seed. By default, it is the value returned by the ANSI C function *time()*, which is usually the number of seconds since some arbitrary past date (often 1 January 1970). This number will also be used as a unique identifer on the first line of the output file.

This can be a useful option. It is recorded in the log file when any program is run. It is possible to recreate exactly what was done using the log file.

-e This requires a filename for the log file. It will be appended to if it exists and created if not. The default is *qtlcart.log*.

-X Give a filename stem. All output will start with this stem and have extensions indicating what is in them.

-D Is a debugging option. It can be followed by a number to indicate the level of debugging output. For example, **-D0** indicates no debugging output and is the default. **-D1** and **-D2** will cause some of the programs to output extra information to the log file. **-D3** will cause the programs to create a file *memacct.txt* and record all allocation and deallocation of memory.

EXAMPLES

For all the following examples, assume that **QTLCart** is just a wildcard for any of the programs in the suite.

```
% QTLcart -R resource.file
```

QTLcart will read option values from the file resource.file. The other programs do this, and except for **Preplot**, will regenerate the file upon exit.

```
% QTLcart -X corn
```

Will set the filename stem to *corn*. The output files will then have names beginning with *corn* and logical extensions. For example, the map file will be placed in *corn.map* and the file containing the data from a cross will be in *corn.cro*. Filenaming conventions should follow the old DOS 8+3 rule due to historical reasons.

GLOBAL COMMAND LINE OPTIONS

All the parameters for **QTLcart** are also parameters for the other programs in the QTL Cartographer system.

GLOBAL BEHAVIOR

All the programs in the QTL Cartographer suite behave in the same general way. They were originally UNIX programs and can be run as such (using command line options). More recently, we have added an interactive menu that allows the user to set parameters. Once inside any of the programs, all the parameters of the program are displayed with their current values. The user chooses whichever parameter he or she wishes to change by selecting a number. The menu is in a loop. Choosing 0 will end the loop and proceed with the current parameter values.

The menu is also where one can get online help. Online help will be a numbered option in the list of parameters. Choose it and specify the location of the help file if the program couldn't find it.

When the programs begin to run, they will print out their parameter values to a log file (*qtlcart.log* by default).

Here is an example of the **Rcross** menu:

```
=================================================================
No.          Options                                    Values:
-----------------------------------------------------------------
 0. Continue with these parameters
 1. Input File
 2. Output File                                         qtlcart.cro
 3. Error File                                          qtlcart.log
 4. Genetic Linkage Map File                            qtlcart.map
 5. QTL Data File                                       qtlcart.qtl
 6. Random Number Seed                                  1014739725
 7. Output format (0,1,2)                               0
 8. Cross (1,2,3) => (B1,B2,F2)                         B1
 9. Heritability                                        0.500000
10. Replications (Not yet active)                       0
11. Interactive Crosses? (0,1) => (no,yes)              0
12. Environmental Variance (used if > 0)                -1.000000
13. Sample Size                                         200
-----------------------------------------------------------------
14. Specify Resource File                               qtlcart.rc
15. Change Filename stem                                qtlcart
16. Change Working Directory:
17. Quit
18. Quit, but update the Resource File
=================================================================

    Please enter a number...
```

This menu is in a loop. To change a parameter, select its number and press return. You will be prompted for a new value or filename. You can clear out a filename or working directory by inputting a single period (.). When satisfied that the parameters are set correctly, you can select 0 to run the program. If you want to quit without changing the resource file, simply select 17. Selecting 18 will update the resource file with any parameter changes you have made.

Each program will have a different number of parameters, thus the last five options may not have the same numbering as in the **Rcross** example above. In addition, **Rmap**, **Rqtl** and **Rcross** have options that only make sense if you are simulating data. These options disappear if you set an input file to translate and thus the last five options are renumbered.

RESOURCE FILE

The resource file keeps track of the most current parameter values used in the programs. Each time the user runs a program, the program accepts new values for parameters and writes them to the resource file. This is unlike the log file which keeps track of the parameters used at the time of running each program. The resource file that is generated by the programs in the suite is self documenting. Look in the *qtlcart.rc* file.

WORKING DIRECTORY

You can specify a working directory (or folder) with the **-W** option. This directory (folder) must exist prior to running any of the programs. The directory can be relative or complete, and should have the standard directory delimiter appended to it. For example

```
-W /home/user/qtlcart/work/
```

would use */home/user/qtlcart/work* as the working directory. All input and ouput files would have to be in this directory. For a Windows system, the line might be

```
-W c:\qtlcart\work\
```

whereas a Macintosh would require

```
-W HardDrive:qtlcart:work:
```

The equivalent line in the resource file would have **-workdir** instead of just **-W**.

In UNIX, you can set a path variable pointing to the programs and simply set your current working directory to the working directory. For Mac, you double click the icons and should use a working directory variable. Relative paths are also possible. For example, if the programs reside in a *bin* folder in the *qtlcart* folder on a Macintosh, then you can have a *data* folder in the *qtlcart* folder and use

```
-W ::data:
```

as the working directory. The two colons mean go up one level and then go into the *data* folder.

On a Windows system, you can either open a command window and type in commands as you would under UNIX, or double click program icons as you would on a Macintosh. If you use the Macintosh mode, then you will need to set a working directory as the resource file is saved in the same directory where the binaries reside. If you use the command line mode, then you should have the binary directory in the PATH variable so that you can run the programs in the working directory and not have to set that variable.

Newer versions of the Macintosh have UNIX underneath the windowing system. QTL Cartographer can be compiled and used in the UNIX enviroment on the Macintosh as under any UNIX system. You will need to get the developer package and install it on your Macintosh to do this.

FILENAME STEM

The filename stem is an important concept in the usage of this package. Beginning with version 1.12, the programs utilize the filename stem *qtlcart*. All files are then named using this stem and filename extensions relevant to the filetype. For example, if the **-X** option is followed by *corn*, then when new files are created, they will have the stem *corn* followed by a logical extension. An example would be *corn.map* for a genetic linkage map. With some practice, you will be able to know the contents of a file by its extension.

USING THE INDIVIDUAL PROGRAMS

For now it is best to use the individual programs rather than the front end. If you have no data, then you would use the programs in the following order:

1. **Rmap**, to create a random map of markers.
2. **Rqtl**, to generate a random genetic model for the map.
3. **Rcross**, to create a random cross.
4. **LRmapqtl**, to do a simple linear regression of the data on the markers.
5. **SRmapqtl**, to do a stepwise linear regression of the data on the markers to rank the markers.
6. **Zmapqtl**, to do interval or composite interval mapping.
7. **Preplot**, to reformat the output of the analysis for **GNUPLOT**.
8. **GNUPLOT**, to see the results graphically.

If you have data, then you might use the programs in the following order:

1. **Rmap**, to reformat the output of MAPMAKER or a standard input file.
2. **Rcross**, to reformat your data.
3. **Qstats**, to summarize missing data and calculate some basic statistics on your quantitative traits.
4. **LRmapqtl**, to do a simple linear regression of the data on the markers.
5. **SRmapqtl**, to do a stepwise linear regression of the data on the markers to rank the markers. This should be run with model 2.
6. **Zmapqtl**, to do interval or composite interval mapping. This should be run twice, once with model 3 and a second time with model 6.
7. **Preplot**, to reformat the output of the analysis for Gnuplot.
8. **GNUPLOT**, to see the results graphically.

We recommend that the new user tries a simulation to gain an understanding of the programs

REFERENCES

1. **T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5**

BUGS

Many UNIX systems have been known to get upset when trying to run the QTL Cartographer programs from out of the front end. It has something to do with the memory management. Try running the individual programs one by one. A good test is to simply run each program without changing any parameters.

8.2 RMAP

NAME

Rmap — Simulate or reformat a map of molecular markers

SYNOPSIS

Rmap [**-o** *output*] [**-i** *input*] [**-g** *gmode*] [**-f** *mapfunc*] [**-p** *mapparam*] [**-c** *chroms*] [**-m** *MarkersPerChrom*] [**-vm** *sdMPC*] [**-d** *InterMarkerDist*] [**-vd** *sdIMD*] [**-t** *Tails*] [**-M** *Mode*]

DESCRIPTION

Rmap creates a random map of molecular markers. The user specifies the number of chromosomes, the number of markers per chromosome and the average intermarker distance. If one specifies standard deviations for the number of markers and the average intermarker distances, they will vary subject to the normal distribution. The output gives a table of markers by chromosomes, with the distances between consecutive markers (in centiMorgans) in the table.

If you specify an input file, **Rmap** will open it, determine if it is in the same format as **Rmap** outputs, and process it based in the value given to **-g**. If the input file is the output of **MAPMAKER**, then the map will be reformatted from **MAPMAKER** into the **Rmap** output format.

Finally, there is a standard input format that **Rmap** can translate, and is defined in the file *map.inp* that comes with the distribution of the programs. Note that if the user specifies an input file, no simulations will be done and the latter half of the command line options are ignored.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This should be used with a filename indicating where the output will be written. **Rmap** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Rmap** will use *qtlcart.map*.

-i You can use this option to specify an input filename. This file must exist and have one of three formats: *Rmap.out*, *map.inp* or *mapmaker.mps*. **Rmap** will attempt to identify the format of the file and translate it to another format. If you specify an input file, then the simulation parameters will be ignored.

-g Requires an integer to indicate the output format. You can use a 1 for the default ouput format, a 2 for **GNUPLOT** output or a 3 for both. If you use a 2 or a 3, then you can use **GNUPLOT** to see graphical version of the linkage map.

-f Requires an integer option to specify the mapping function. **Rmap** can use the Haldane, Kosambi, fixed or a number of other functions. The default is to use the Haldane function, which is specified with a 1. Using a 2 invokes the Kosambi mapping function. A 3 means that a fixed function is used and thus the distance in Morgans is the recombination fraction. The type of mapping function used would then be recorded in the ouput and all following analyses will use this function. One must edit the map file to change this if not using **Rmap**.

-p Requires a real number. Some map functions need an extra parameter, and this allows the user to specify it. See the manual for details.

-c This allows you to specify the number of chromosomes if you are simulating a genetic linkage map. It is 4 by default. If you are translating a file, then this will be ignored as will the remaining options.

-m This allows you to specify the average number of markers per chromosome in a simulation. The default is 16.

-vm This allows you to specify the standard deviation in the number of markers per chromosome. The number of markers per chromosome will have a normal distribution with mean given in the previous option, and the standard deviation specified here. If zero, then each chromosome will have the same number of markers.

-d **Rmap** uses the value given after this option as the average intermarker distance (in centiMorgans) for a simulation. It is 10 centiMorgans by default.

-vd The intermarker distance will have a normal distribution with mean set by the previous option and standard deviation specified with this option. It is 0.0 by default, which means that the intermarker distances between consecutive markers will all be the same. Set it to a positive value to have intermarker distances vary at random.

-t You can simulate maps where there are no markers on the telomeres with this option. Give this option a value of *tails* and **Rmap** puts an average of *tails* Morgans of genetic material on the ends of the chromosomes. By default, it is 0.0. If the standard deviation for intermarker distance is greater than 0.0, then then the amount of flanking DNA will have a normal distribution with mean given here and standard deviation proportional to that of the standard deviation of intermarker distances.

-M Allows you to specify an alternate simulation mode. If the **-M** option is used with a value of 1, then the intermarker distance will be used as the chromosome length (so you should make it longer), and the markers are placed on the chromosomes following the uniform distribution. The value of this option will be returned to 0 at the completion of the program.

INPUT FORMAT

Rmap recognizes four types of files. The first is the *Rmap.out* format that **Rmap** itself creates. The second is a special format defined in the example file *map.inp* included in the distribution. The third format is the output of **MAPMAKER**. If the input file is a **MAPMAKER** output file, **Rmap** translates this file into its own format. If the input file is already in the correct format, **Rmap** will output it dependant upon the flag given to the **-g** option. The units of intermarker distances will be in centiMorgans in the output.

Rmap also recognizes the input files for the Windows GUI version of **QTL Cartographer** (Wang, *et al.,* 2002).

EXAMPLES

```
% Rmap -o Map.out -c 23 -vm 3 -vd 1 -t 5
```

Simulates a random map where the number of markers on each of 23 chromosomes has a normal distribution with mean 16 and standard deviation 3. The intermaker distance is normally distributed with mean 10 cM and standard deviation 1. There will be some genetic material outside the flanking markers on each chromosome, with a mean length of 5 cM and standard deviation 0.5.

```
% Rmap -o Map.out -i map.mps
```

Opens the file *map.mps*, tries to determine its format, and translates it if possible. The output will be written to the file *Map.out*. The extension **.mps** should be used with **MAPMAKER** output files and the string *-filetype mapmaker.mps* should be put somewhere in the first twenty lines of the file.

```
% Rmap -i map.inp -g 3 -X test
```

This opens the file *map.inp* and translates it. Two output files are produced. The file *test.map* contains the genetic linkage map in *Rmap.out* format, while a file *testmap.plt* contains code for **GNUPLOT**. The next step would be to start **GNUPLOT** and load *testmap.plt*.

```
% gnuplot
gnuplot> load "testmap.plt"
Hit return to continue
gnuplot> quit
```

REFERENCES

1. Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln and L. Newburg (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* **1**, 174−181.

2. Wang, S., C. J. Basten and Z.-B. Zeng (2002) Windows QTL Cartographer: WinQtlCart V2.0.

3. T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5

BUGS

Note that if **MAPMAKER** outputs an intermarker distance of 0.00 cM, then **Rmap** will translate it to 0.0001 cM. In fact, all intermarker distances of 0.0 will be reset to 0.0001 cM.

Prior to version 1.17d, if you tried to simulate more than 50 linkage groups, you would crash the other programs in the QTL Cartographer Suite. This bug was fixed with version 1.17d.

8.3 RQTL

NAME

Rqtl — Place a set of estimated or randomly generated QTLs on a molecular map.

SYNOPSIS

Rqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-b** *beta*] [**-t** *Traits*] [**-q** *QTLperTrait*] [**-d** *dominance*] [**-1** *beta1*] [**-2** *beta2*] [**-E** *prop*] [**-M** Rmode]

DESCRIPTION

Rqtl will translate a genetic model or simulate a random model for use by **Rcross** to simulate a data set. It places a specified number of QTLs (Quantitative Trait Loci) on the molecular map created or translated by **Rmap**. For simulations, they are placed randomly on the map, and the additive, dominace and epistatic effects are also determined. The molecular map could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Rqtl** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Rqtl** will use *qtlcart.qtl*.

-i This requires an input filename. This file must exist. **Rqtl** will attempt to identify the format of the file and translate it to another format. This file should contain a genetic model defining a set of QTL and including their positions and effects. See the file *qtls.inp* for the format.

-m This requires a filename that must exist. **Rqtl** will read the genetic linkage map from this file.

-t This allows the user to specify the number of traits to simulate. It is 1 by default.

-q This requires an integer argument. It allows the user to specify the number of QTL that affect the trait. If one trait is simulated, then exactly this number of QTL will be created. If more than one trait are simulated, then the number of QTL per trait will vary but have mean value specified here. The default is 9.

-d You can specify the type of dominance at the trait loci. If we assume inbred parental lines with line one marker trait alleles all Q and line two trait alleles all q, then use a 1 for no dominance, a 2 for complete dominance of Q over q, a 3 for complete dominance of q over Q, and a 4 for dominance that is random in direction and magnitude for each locus. It is 1 by default, that is no dominance.

-b Specifies the parameter needed to determine the additive effect of a QTL. It is 0.5 by default. See Zeng (1992) equation (12) and accompanying text for a discussion of this parameter. Itis not the allelic effect of a QTL allele, rather it is the shape parameter in the beta distribution.

-1, -2
Allows you to specify the two parameters used to determine the dominance effect of a QTL. The effect is simulated from a beta distribution. See the manual for more details.

-E For a k QTL model, there will be *2k(k+1)* potential epistatic terms. This option sets the proportion of epistatic interactions that will be non-zero in a simulated model. The effects are generated with the same beta funtion used for the dominance effects.

-M By default, **Rqtl** will not place a new QTL on the same interval or an adjacent interval. If you use this option with a value of 1, then it will allow QTL in adjacent intervals. The value of this option will be returned to 0 at the completion of the program.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**.

If a file is specified with the **-i** option, then that file will be read for the positions and effects of the QTLs. The format of this file should be identical to that of the output of **Rqtl**, or of a special format defined in the file *qtls.inp* included with the distribution.

EXAMPLES

```
% Rqtl   -d 2
```

Places 9 QTLs on the map in *Rmap.out*. There is complete dominance of A over a.

```
% Rqtl   -i qtls.inp -o test.qtl
```

Reads the file *qtls.inp* and translates it into the output format of **Rqtl**. The output is written to the file *test.qtl*, which is overwritten if it exists.

REFERENCES

1. Zeng, Zhao-Bang (1992) Correcting the bias of Wright's estimates of the number of genes affecting a quantitative trait: A further improved method. *Genetics* **132**, 823−839.

BUGS

The **-t** option for the number of traits is rather primitive at this time. The number of QTLs and their effects are randomly determined, with means given in the other options.

8.4 RCROSS

NAME

Rcross — Simulate or reformat a data set.

SYNOPSIS

Rcross [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-q** *modelfile*] [**-g** *Output*] [**-r** *repetitions*] [**-c** *Cross*] [**-n** *SampleSize*] [**-H** *heredity*] [**-E** *Ve*] [**-I** *Interactive*]

DESCRIPTION

Rcross performs a random cross or reformats a data set. Cross types include F1 backcrosses to the P1 or P2, F2 crosses produced by selfing or random mating, recombinant inbred lines as well as a few others. It simulates marker and trait data. The markers simulated come from a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The QTL model could be a random set produced by **Rqtl** or an estimated set in the same format as the output of **Rqtl**.

Rcross can also translate files from three different formats. If the user chooses to translate a file, then the simulation options are ignored.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Rcross** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Rcross** will use *qtlcart.cro*. This output is in a format suitable for any of the mapping programs.

-i This requires an input filename. This file must exist. **Rcross** will attempt to identify the format of the file and translate it to another format. Specifying a file with this option turns off the simulation parameters below.

-m **Rcross** requires a genetic linkage map. This option require the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-q **Rcross** needs a genetic model to simulate a data set. It will read from the file specified by this option. The file specified should contain a genetic model in the same format as the output of **Rqtl**. The default file is *qtlcart.qtl*.

-H Allows the user to specify the heritability for the trait. If used, it requires a value in the range 0.0 to 1.0. It is 0.5 by default.

-E Allows the user to specify an environmental variance for the trait. If used, it requires a positive value and will disable the heritability. This is ignored by default.

-I is the flag to turn on interactive crosses. By default, it has a value of 0. To do interactive crosses, use this option with the value 1.

-c Allows the user to specify the type of cross. It requires a string such as B1, SF2 or RI1. See below for more on the values of the cross.

-g This should be used with an integer in the range 0 to 6. It specifies the format of the output. The default is 0 and the other options are defined below.

-n This is the sample size of the offspring. It is 200 by default and requires some integer value greater than 0 if used.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input form of the QTL data should be that of the output format from **Rqtl**. If an input file for the data is used, then it can have one of two formats. The first is identical to the *raw* files required by **MAPMAKER**. You must first use **MAPMAKER** to create a genetic map, then run the map through **Rmap** to reformat it, then use the map and the original raw file to reformat the data for subsequent use.

An alternative format is defined in a file *cross.inp* that is included with the distribution. The file can be annotated freely. Look at the *cross.inp* file and use it as a template for your data. In addition, you can use the *qtlcart.mcd* files that were formatted for the Windows version of **QTL Cartographer**.

Rcross can read the input files formatted for use with **PLABQTL**. You will need to add the phrase *-filetype plabqtl0.inp* (matrix format) or *-filetype plabqtl1.inp* (vector format) to the first line of the **PLABQTL** formatted input file. Also, be sure that there are no map files in the current working directory: You want **Rcross** to read the map that is in the **PLABQTL** input file. If your **PLABQTL** input file has measurements for your traits in different environments, you need to add the phrase *-environments x* at the end of the first line, where x is the number of environments. If you have t traits, then **Rcross** will output $(x+1)t$ traits. Generally, the main block of data will have the means of each trait over environments while the raw data are appended to the end of the **PLABQTL** file. The raw measurements will be named *TrYEnvX* in the output, where Y and X are the trait and environment numbers. For example, if weight (trait 1) and height (trait 2) are the two traits and they are measured in three environments, then there will be eight traits in the output file. The raw data for weight will be named *Tr1Env1*, *Tr1Env2* and *Tr1Env3*, while for height they will be *Tr2Env1*, *Tr2Env2* and *Tr2Env3*.

OUTPUT

Rcross can produce eight different types of output files. The output formats are specified by an integer from 0 to 7 used with the *-g* command line option or set in the text menu. The numbers correspond to the following output formats:

Value 0.
qtlcart.cro: This is the default and standard format for the analysis programs in the **QTL Cartographer** package.

Value 1.
cross.inp: This is a standard input format designed for **Rcross**. The manual has more details.

Value 2.
mapmaker.raw: This is the **MAPMAKER** raw file format.

Value 3.
qtlcart.r: This format is suitable for import into the programs **R** or **Splus**. All the data will be written into the file and embedded in commands that allow **R/Splus** to read it. In addition, **Rcross** writes a set of commands to do ANOVA analyses of each trait on each marker and categorical trait. For a file named *qtlcart.r*, use the command **source("qtlcart.r")** in **R/Splus** to import the data. Note that names of traits, markers and categorical traits must conform to **R/Splus** usage.

Value 4.
qtlcart.sas: This format is a SAS program that has all the data and a set of **PROC GLM** commands to do ANOVA and Means analyses similiar to the **R/Splus** option above. Note that names of traits, markers and categorical traits must conform to **SAS** usage. Thanks to Emilio Carbonell for the suggestion.

Value 5.
plabqtl0.qdt: This will produce a file suitable for input into **PLABQTL**. The data will be in the matrix format. See the **PLABQTL** manual for more details.

Value 6.
plabqtl1.qdt: This will produce a file suitable for input into **PLABQTL**. The data will be in the vector format. See the **PLABQTL** manual for more details.

Value 7.
qtlcart.mcd: This will produce a file suitable for input into the Windows GUI version of **QTL Cartographer** (Wang, *et al.,* 2002). This is a single file with the genetic linkage map and the data set. The format is similar to the *map.inp* and *cross.inp* formats.

CROSSES

A pair of inbred parental lines (P1 and P2) that differ in the trait of interest and marker genotypes are crossed to produce an F1 generation. All crosses are then derived from these lines.

Backcrossing to P1 is encoded by B1, and to P2 by B2. Selfed intercrosses of generation i are encoded by SFi. Randomly mated intercrosses of generation i are encoded by RFi. Recombinant inbreds created by selfing have the code RI1, while those by sib-mating are RI2. Doubled haploids have the code RI0. A test cross of an SFi line to a Pj line is encoded by T(Bj)SFi. The QTL Cartographer manual explains some other crosses that are possible. Note that the UNIX shell may interpret (and) so they should either be quoted, or the cross entered into the interactive menu.

Rcross uses the general genetic model developed by Cockerham (1954).

EXAMPLES

```
% Rcross  -A -V -c SF2 -n 1000
```

Does a selfed F2 cross with 1000 offspring using the linkage map in *qtlcart.map* and the model in *qtlcart.qtl*. The command line options **-A** and **-V** turn off the interactive menu and the verbosity mode, respectively.

```
% Rcross  -i cross.raw
```

Reads from the file *cross.raw*, tries to determine its format, and translates it if possible. The file *cross.raw* could be a **MAPMAKER/QTL** formatted file, a *cross.inp* formatted file or one that is already in the *Rcross.out* format.

REFERENCES

1. Cockerham, C. C. (1954) An extension of the concept of partitioning hereditary variance for analysis of covariances among relatives when epistasis is present. *Genetics* **39**, 859−882.

2. Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln and L. Newburg (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* **1**, 174−181.

3. Utz, H.F. and A.E. Melchinger (1996) PLABQTL: Aprogram for composite interval mapping of QTL. *J. Agric. Genomics* **2**(1).

4. Wang, S., C. J. Basten and Z.-B. Zeng (2002) Windows QTL Cartographer: WinQtlCart V2.0.

BUGS

If you use the interactive mode, you can print out the results of crosses. The analysis of these arbitrary crosses has not been fully integrated into the other programs.

The input subroutines for importing **PLABQTL** files has not been extensively tested. Check your output for correctness.

The Windows version of QTL Cartographer (Wang, *et al.,* 2002) requires files with DOS line endings. Either do the conversions on a Windows machine, or ftp the files to the Windows machine as text.

8.5 PRUNE

NAME

Prune — Prune or resample the data set.

SYNOPSIS

Prune [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-I** *interactive*] [**-M** *Model*] [**-b** *simflag*]

DESCRIPTION

Prune allows one to eliminate markers or traits. It removes the data from the file containing the cross and reconstructs the molecular map. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.

Prune also does bootstraps, permutations and simulations of missing or dominant markers.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename stem for output. **Prune** will overwrite the file ending in *.crb* if it exists, and create a new file if it does not. If not used, then **Prune** will use *qtlcart.crb*. If the map is recreated, then a new map file will be written to *qtlcart.mpb* by default or a file ending in mpb with the specified stem.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **Prune** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-I Sets the interactive level. A zero means that Prune will do what it needs to without asking (the default for bootstraps, permutations or missing data simulations). A one means that the user will be put into a repeating loop to manipulate the data set. It has a value 1 by default, but using the **-b** option disables it.

-M This sets a level for the elimination of individuals with this much missing marker data, or for the simulation of missing or dominant markers when used with the **-b** option.

-b Prune will read in the map and data file and do one of **AUTOMATIC ACTIONS** described in the section of the same name below. A value of zero means that this option is ignored.

-t Set the trait to process if using **-b 7**.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

AUTOMATIC ACTIONS

There are a number of automatic actions that can be performed using the **-b** option. You will use one of the numbers below with the option to tell **Prune** to do that action. A new dataset is then printed to a file *stem.crb*, where *stem* is the filename stem. Note that if you give a nonzero value to this option, the interactive flag is turned off.

1. Perform a bootstrap resampling of the data. Sampling of individuals is done with replacement to create a sample of the same size as the original.

2. Permute the the traits against genotype arrays. If there are multiple traits in the data set, then each trait will be shuffled against the genotype arrays.

3. Simulate missing markers. The percent of missing marker data should be specified with the **-M** option, and it should be an number in the range of 0 to 100 percent.

4. Simulate dominant markers. The percent of dominant marker data should be specified with the **-M** option, and it should be an number in the range of 0 to 100 percent. The direction of dominance is random.

5. Simulate selective genotyping. The percent of typed individuals should be specified with the **-M** option, and it should be an number in the range of 0 to 100 percent. This will print out individuals with trait values in the tails of the overall distribution. The value specified will be the sum of these tails: Each tail will have half of the total. This will apply to whichever trait was last analyzed, or trait 1 if all the traits had been analyzed. It is probably best to do this with single trait data sets.

6. Permute the the traits against genotype arrays. A value of 12 does this as well. If there are multiple traits in the data set, then entire trait arrays will be shuffled against the genotype arrays. This contrasts with option 2 above which permutes the traits independently. If you think the traits are correlated and you want to maintain that correlation, use this option. Otherwise, use option 2.

7. Prune the data back to one trait. Use the **-t** option with a trait number to select the trait. The output will have one trait: All individuals with missing values for this trait will also be deleted.

8. Prune the data to specified traits. Use the **-t** option with a trait number to select the trait. If the original data has t traits, then an integer in the range *[1,t]* will eliminate all but the specified trait, that is it will do exactly the same thing as option 7 above. If an integer less than one is used, then only traits whose names begin with a plus sign will remain in the output. If greater than the number of traits, then any trait whose name begins with a minus sign will be eliminated. Once the traits are eliminated, all individuals with missing data for any of the surviving traits will also be deleted.

9. Remove all categorical trait information. This is for compatibility with R/QTL, which can not read categorical trait information as of 8 June 2004.

10. Collapse the genetic linkage map and data. If you have a large number of markers and infer a genetic linkage map from a small sample, then there will be a lot of intermarker distances of 0.0. For all neighboring markers A and B with zero intermarker distance, A will be replaced with B and all data combined. (The marker arrays should be identical except for missing data).

EXAMPLES

```
% Prune -m example.map -i example.cross -o exout
```

Puts the user into an interactive menu for eliminating traits, markers, etc.

```
% Prune -m example.map -i example.cross -o exout -b 1
```

The **-b** option creates a new sample from the old. The new sample is created by resampling the original sample with replacement. Phenotypes and genotypes are kept together. The new sample will have the same sample size as the old one. It will be written to *exout.crb*. No new map will be written.

```
% Prune -m example.map -i example.cross -o exout -b 5 -M 20.0
```

Here, the **-b** option tells **Prune** to selectively genotype. We specify 20.0 percent with the **-M** option meaning that those individuals with trait values falling in the lower and upper 10 percent tails are retained, and the middle 80 percent are removed.

Suppose you have a large set of markers (say 5,000) and a data set of 100. The sample size is too low for you to observe any recombinant events between many pairs of markers. You can reestimate the genetic linkage map with **Emap** and then use **Prune** to trim out markers that appear to be redundant. Suppose the data are in *cross.inp* and the genetic linkage map is in *map.inp*. The map file must have the correct marker order, but need not have the correct intermarker distances.

```
%  Emap -i cross.inp -m map.inp -S 0.0
%  Prune -b 10
```

will produce files *qtlcart.mpb* and *qtlcart.crb* that have a reduced number of markers but the maximal amount of marker data.

BUGS

You can eliminate multiple markers in the interactive loop. You should be aware that the order marker elimination is important. If all the markers to be eliminated are on separate chromosomes, the order is unimportant. If two markers from the same chromosome are to be eliminated, order should be to eliminate the highest numbered marker. The same concept holds for traits: eliminate them in the order of highest to lowest.

Do not try to eliminate any markers or traits AND do a bootstrap, permutation or simulation of missing markers in the same run.

8.6 EMAP

NAME

Emap — Infer a genetic linkage map from data

SYNOPSIS

Emap [**-i** *data input*] [**-o** *data output*] [**-m** *map input*] [**-l** *map output*] [**-S** *s size*] [**-L** *l size*] [**-M** *method* [**-r** *permutations*] [**-f** *map function*] [**-p** *parameter*] [**-O** *obj. function*]

DESCRIPTION

Emap infers a genetic linkage map from a data set. It uses the rapid chain delineation method of Doerge and Weir.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o Use this to specify the data output file. **Emap** will overwrite the file if it exists, and create a new file if it does not. The default value is *qtlcart.cro*.

-i You need to specify a data input file with this option. The data should be **Rcross.out**, **cross.inp** or **mapmaker.raw** format.

-m Use this to specify the map input file. **Emap** will read the map from this file and try to rearrange markers to improve their ordering.

-l The completed map will be written to a file specified with this option. The default is *qtlcart.map*.

-M Requires an integer to indicate the linkage map method. At present, the only options are 10, 11, 12 or 13.

-f Requires an integer option to specify the mapping function. See **Rmap(1)** for more information on mapping functions.

-p Requires a real number. Some map functions need an extra parameter, and this allows the user to specify it. See the manual for details.

-S This allows you to specify the significance level for declaring segregation distortion between a pair of markers.

-L This allows you to specify the significance level for declaring linkage between a pair of markers.

-r This allows you to specify the number of permutations. It is not an active option at this time.

-O This allows you to specify the objective function. Use 0 for SAL (sum of adjacent likelihoods) or 1 for SAR (sum of adjacent recombination fractions).

INPUT FORMAT

Emap recognizes three types of files. The first is the *Rcross.out* format. The second is a special format defined in the example file *cross.inp* included in the distribution. Finally, **MAPMAKER** raw files can be read by **Emap**.

EXAMPLES

```
% Emap -i sample.raw
```

Will attempt to create a genetic linkage map for the data in the *sample.raw* file.

REFERENCES

1. Doerge, R.W. and B. S. Weir (1999) . *XXX* **1**, 174−181.

BUGS

This is an initial version and needs some work. It does fine on simulated data, but could use some testing with real data.

8.7 QSTATS

NAME

Qstats — Calculate basic statistics for a QTL dataset.

SYNOPSIS

Qstats [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-p** *yes*]

DESCRIPTION

Qstats does some basic statistics on a dataset of quantitative traits. It plots a histogram and calculates the sample size, mean, variance standard deviation, skewness, kurtosis, and average deviation for a quantitative trait. The program also summarizes missing marker and trait data, as will as determining the marker types (dominant or codominant). Finally, **Qstats** will test whether markers are segregating at random. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Qstats** will append the file if it exists, and create a new file if it does not. If not used, then **Qstats** will use *qtlcart.qst*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **Qstats** requires a genetic linkage map. This option require the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-p requires an argument *yes* or *no*. By default it is *no*. If used with *yes*, then the probability distribution for each marker and each individual will be written in a section at the end of the output file.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

OUTPUT

As mentioned above, the output will have sections for basic statistics on the traits, missing data summaries and segregation statistics. If the **-p** option is used with an argument *yes*, then marker distributions are printed in a section at the end. For each marker, there will be a table that has individuals as rows and the probability distribution and expected values of the marker as columns. An example of the output is

```
-begin markerprobs
-markername BD267                  -chromosome     1 -marker     1
-h   ind.   p(QQ)   p(Qq)    p(qq)   E(a)     E(d)
        1  0.3304  0.5242  0.1454   0.1851   0.0242
        2  0.3441  0.4850  0.1709   0.1731  -0.0150
        3  0.3747  0.4958  0.1295   0.2452  -0.0042
        4  0.0696  0.3884  0.5421  -0.4725  -0.1116
        5  0.2597  0.5696  0.1707   0.0889   0.0696
        6  0.2903  0.5500  0.1598   0.1305   0.0500
        7  0.2903  0.5500  0.1598   0.1305   0.0500
        8  0       0       1       -1.0     -0.5
```

The section begins with *-begin markerprobs* and ends with *-end markerprobs*. Each marker is then listed and a header line is followed by the output. The first column is the individual number. The next three are the probabilities of QQ, Qq and qq genotypes, respectively. Then, the expected values for additive and dominance effects are calculated via

```
E(a) = p(QQ) - p(qq)
E(d) = [p(Qq)-p(QQ)-p(qq)]/2
```

One could use the final two values as replacements for the marker values in a regression analysis. The additive expectation is calculated assuming that QQ genotypes have value 1, while Qq and qq genotypes have values 0 and -1, respectively. The dominance expectation sets heterozygotes equal to $1/2$ and homozygotes to $-1/2$.

This option produces a lot of output: It is off by default. The probabilities follow from Jiang and Zeng (1997).

Note that the line for individual 8 has $p(\mathrm{QQ}) = p(\mathrm{Qq}) = 0$, while $p(\mathrm{qq}) = 1$. This is a dominant marker and individual 8 had the unambiguous genotype qq. You would see similar output for codominant markers.

EXAMPLES

```
% Qstats -i corn.cro -m corn.map
```

Calculates basic statistics on the dataset in *corn.cro* using the genetic linkage map in *corn.map*. The program will display and interactive menu for setting options and print out messages to the screen while running. These can be turned off with **-A** and **-V**, respectively. If the dataset in *corn.cro* has more than one trait, then all traits will be analyzed.

REFERENCES

1. M. Lynch and B. Walsh (1998) Genetics and Analysis of Quantitative Traits. Sinauer Associates, Sunderland, MA.

2. C. Jiang and Z.-B. Zeng (1997) Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* **101**, 47−58.

BUGS

Are there any other statistics that we can do? Your suggestions are welcome.

8.8 LRMAPQTL

NAME

LRmapqtl — Single marker QTL analysis.

SYNOPSIS

LRmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-r** *reps*] [**-t** *trait*]

DESCRIPTION

LRmapqtl uses simple linear regression to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **LRmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **LRmapqtl** will use *qtlcart.lr*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **LRmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-r **LRmapqtl** will do a permutation test *a la* Churchill and Doerge (1994). This option specifies the number of permutions to do. It is zero by default, which means no permuation test is done. If used, you must specify a positive integer. Usually, 1,000 is sufficient.

-t Use this to specify which trait **LRmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed. The default is to analyze trait 1 only.

MODEL

The basic linear model is

```
Trait = Mean + Slope x Marker + Error
```

The marker value will be in the range [-1, 1] inclusive. Two hypotheses are compared. The null hypothesis is that the Slope is zero. The alternate is that the Slope is non-zero. A p-value for the likelihood ratio of these to hypotheses is calculated for each marker-trait combination. **LRmapqtl** outputs a table with parameter estimates, F statistics, Likelihood ratios and p-values.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% LRmapqtl -i corn.cro -m corn.map
```

Calculates the regression coefficients for each marker on the dataset in *corn.cro* using the genetic linkage map in *corn.map*.

REFERENCES

1. Churchill, G. A. and R. W. Doerge (1994) Empirical threshold values for quantitative trait mapping. *Genetics* **138**, 963−971.

BUGS

8.9 SRMAPQTL

NAME

SRmapqtl — Map quantitative traits on a molecular map.

SYNOPSIS

SRmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-t** *trait*] [**-M** *Model*] [**-F** *pFin*] [**-B** *pFout*] [**-u** *MaxSteps*]

DESCRIPTION

SRmapqtl uses stepwise regression to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**.

This program should be run before .**Zmapqtl** if you want to use composite interval mapping. The results will be used to pick markers background control in composite interval mapping. The main result from using this program is to rank the markers in terms of their influence on the trait of interest.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **SRmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **SRmapqtl** will use *qtlcart.sr*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **SRmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **SRmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed. The default is to analyze trait 1 only.

-M This tells **SRmapqtl** what type of analysis to perform. Use a 0 for forward stepwise (FS) regression, a 1 for backward elimination (BE) and a 2 for forward regression with a backward elimination step at the end (FB). It is probably best to use Model 2 here.

-F Requires a real number in the range 0.0 to 1.0. This is a threshold p value for adding markers in model 2 during the forward stepwise regression step. The default is 0.05.

-B Requires a real number in the range 0.0 to 1.0. This is a threshold p value for deleting markers in model 2 during the backward elimination step. It should probably be the same as the previous option. The default is 0.05.

-u Requires an integer valued argument. This allows you to specify a hard limit to the number of steps in a forward regression analysis. It is valid for models 0 and 2. By default, it is 100.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% SRmapqtl -i corn.cro -m corn.map -M 2
```

Does a forward stepwise regression with a backward elimination step for the dataset in *corn.cro* using the genetic linkage map in *corn.map*.

REFERENCES

BUGS

Forward and backward regression should probably use the thresholds for adding and deleting markers from the model. When that feature is added, the **-F** and **-B** options will have more use.

8.10 ZMAPQTL

NAME

Zmapqtl — Composite interval mapping module

SYNOPSIS

Zmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-l** *lrfile*] [**-S** *srfile*] [**-t** *trait*] [**-M** *Model*] [**-c** *chrom*] [**-d** *walk*] [**-n** *nbp*] [**-w** *window*] [**-r** *perms*] [**-r** *boots*]

DESCRIPTION

Zmapqtl uses composite interval mapping to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**. In addition, the program requires the results of the stepwise linear regression analysis of **SRmapqtl** for composite interval mapping.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Zmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **Zmapqtl** will use *qtlcart.z*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **Zmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **Zmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed. The default is to analyze trait 1 only.

-l Allows the user to specify the name of the file containing results from **LRmapqtl**. **Zmapqtl** reads those results and uses the information to choose cofactors for some of the analysis methods.

-S Allows the user to specify the name of the file containing results from **SRmapqtl**. **Zmapqtl** reads the results and uses the information to choose cofactors for composite interval mapping model 6.

-M **Zmapqtl** assumes the specified model (see below) in the analysis. Model 3 is default.

-c The user can specify a specific chromosome for **Zmapqtl** to analyze. If zero, then all will be analyzed.

-d **Zmapqtl** walks along the chromosome at a rate that can be specified with this option. The default is to do an analysis every 2 centiMorgans along the chromosome.

-n Use this to indicate how many background parameters **Zmapqtl** uses in composite interval mapping. This is used only with model 6, and gives an upper bound. If fewer than this number of markers are ranked in the *SRmapqtl.out* file, then less than the specified number of markers will be used.

-w **Zmapqtl** blocks out a region of this many centiMorgans on either side of the markers flanking the test position when picking background markers. It is 10 by default and is only used in models 5 and 6. We refer to it as the *window size*.

-r **Zmapqtl** can do a permutation test to determine the threshold for rejecting the null hypothesis of no QTL at a site. By default, this option sets the number of permutations equal to 0, which means no permutation test is run. You can set it to a number ¡ 10000 to do the test. See Churchill and Doerge (1994) for more details. The results are in an interim file. Use **Eqtl** to summarize them when enough repetitions have been done. You need to run **Zmapqtl** without permutations or bootstraps at least once before you can do the permutation tests. This option only allows for interval mapping (Model 3) or composite interval mapping (Model 6).

-b When used with argument 1, **Zmapqtl** will do a single bootstrap. You need to run **Prune** to actually create the bootstrapped data set: This option merely analyzes it and stores summary statistics in an interim file *qtlcart.z3b* by default, for model 3. You should also run **Zmapqtl** without bootstraps or permutation tests before doing a bootstrap analysis.

When used with an argument 2, **Zmapqtl** will do a jackknife analysis. Again, **Zmapqtl** should be run without this argument prior to doing a jackknife.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% Zmapqtl
```

Calculates the likelihood ratio test statistics of the dataset in *qtlcart.cro* using the map in *qtlcart.map*.

```
% nice Zmapqtl -A -V -i corn.cro -m corn.map -M 6 -r 500 &
```

Calculates the likelihood ratio test statistics of the dataset in *corn.cro* using the map in *corn.map*. Model 6 is used for analysis and a permutation test with 500 replications is performed. The program is nice'd as a courtesy to other users, and run in the background so that the user can logout and relax.

MODELS

Different parameters for the **-M** option allow for the analysis of the data assuming different models. Models 1−3 were described in Zeng (1993, 1994).

1. Fit all the background markers. This was meant for illustration of the method in the original paper (Zeng, 1993), and should not be used for analysis.
2. Fit all unlinked background markers. This is another illustrative example and also should not be used.
3. Fit only the mean (Lander and Botstein (1989) method)
4. Fit a subset of the other markers, namely those unlinked markers with the highest correlation with the trait on each chromosome. This is an *ad hoc* model programmed in anticipation of model 6 below.
5. This model uses a pair of markers from each other chromosome and all linked markers that fall outside a window around the flanking markers. This window extends to 10 cM beyond the markers immediately flanking the test position. The window size can be changed with the **-w** option. This is another *ad hoc* model programmed in anticipation of model 6 below.
6. This model uses a specified number of markers that fall outside a window around the flanking markers. This window extends to 10 cM beyond the markers immediately flanking the test position. The number of markers are set by the **-n** option. You need to run **SRmapqtl** to rank the markers before using model 6. You should use this for composite interval mapping when using markers ranked by **SRmapqtl**.
7. This model requires that you have already run **Zmapqtl** and **Eqtl**. It reads in the estimated QTL from the *Eqtl.out* file and uses them as virtual markers to control for the genetic background. All identified markers are used that do not fall within the window. This has not been extensively tested: Use it at your own risk.
8. Like model 7, this requires a prior run of **Zmapqtl** and **Eqtl**. Instead of using virtual markers, **Zmapqtl** uses the closest flanking markers to identified QTL. Again, all of these markers outside the window are used. This is good model. You can iterate this process to see if a stable set of cofactors can be identified. See the script below and the **Model8.pl(1)** man page for more details.

The default is to fit only the mean, that is to use interval mapping.

PERMUTATION TESTS

Churchill and Doerge (1994) describe a method to calculate the threshold values for quantitative trait mapping that we have implemented in this program. Basically, it does a permutation of the trait values and the genotypes and redoes the analysis. Over the number of replicates, two types of thresholds are defined: "experimentwise" and "comparisonwise". We calculate the experimentwise thresholds, but only give p values for the comparisonwise values to save on storage space. The p values give the proportion of permuted replicates that have loglikelihood ratios larger than the observed ratios.

If you choose to do permutation tests, you need to run **Zmapqtl** with the model of choice prior to doing the permutation test. Also, if the program terminates prematurely, you can restart it from where it left off to complete the permutation test.

REFERENCES

1. Churchill, G. A. and R. W. Doerge (1994) Empirical threshold values for quantitative trait mapping. *Genetics* **138**, 963−971.

2. Lander, E. S. and D. Botstein (1989) Mapping Mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics* **121**, 185−199.

3. Zeng, Zhao-Bang (1993) Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc. Natl. Acad. Sci., USA* **90**, 10972−10976.

4. Zeng, Zhao-Bang (1994) Precision mapping of quantitative trait loci. *Genetics* **136**, 1457−1468.

CAVEATS

Model 7 has not been extensively tested. Remember that you will need to run **Zmapqtl** with some other model (say 3 or 6) and then run **Eqtl** prior to using model 7.

Model 8 requires a prior run of **Zmapqtl** and **Eqtl** as well. There is a shell script called **Model8.csh** in the **scripts** subdirectory that allows you to easily iterate using model 8. Try using it to see whether your set of cofactors (or estimated QTL) becomes stable after a few iterations. A low threshold may lead to adding and dropping putative QTL over consecutive analyses using Model 8. Here is the shell script:

```
#!/usr/bin/csh
#
#  Run Model 8 iteration
#           Copyright 2001 Christopher J. Basten
#   Usage:
#   Model8  bindir  stem   siglevel  iterations  max_nbp
#     bindir is the binary subdirectory
```

```
#     stem is the filename stem
#     siglevel  is the significance level to declare a QTL
#     iterations is the number of iterations
#     max_nbp is the maximal number of background parameters.
#
if ( $1 == '-h' ) then
echo "    Usage:  Model8.csh bindir  stem   siglevel  iterations  max_nbp"
echo "Where"
echo "        bindir  = QTL Cart. binary directory"
echo "          stem  = filename stem"
echo "      siglevel  = Significance level to declare a QTL"
echo "    iterations  = number of iterations"
echo "       max_nbp  = maximal number of background parameters"
echo " "
echo "Now exiting"
exit
endif
set bindir=$1
set stem=$2
set siglevel=$3
set iterations=$4
set maxnbp=$5
$bindir/Qstats -X $stem -A -V
$bindir/Zmapqtl -A -V -M 3
$bindir/Eqtl -A -V -S $siglevel
#
#  Save the original files
#
/usr/bin/mv $stem.eqt $stem.eqt.0
/usr/bin/mv $stem.z $stem.z.0
/usr/bin/cp $stem.sr $stem.sr.0
#
#  Use model 8 iteratively with cofactors from previous run.
#
set i=1
while ( $i < $iterations )
echo "Doing iteration $i"
$bindir/Zmapqtl -A -V -M 8 -n $maxnbp
/usr/bin/rm $stem.sr
$bindir/Eqtl -A -V -S $siglevel
/usr/bin/cp $stem.sr $stem.sr.$i
/usr/bin/mv $stem.eqt $stem.eqt.$i
/usr/bin/mv $stem.z $stem.z.$i
@ i++
```

```
end
/usr/bin/rm $stem.sr
echo "Finished"
```

The above script has been translated into **Perl** to make it more useful. The **Model8.pl** script can take command line parameters and is self-documenting.

BUGS

It is likely that we will abandon the internal permutation tests in **Zmapqtl**. It is more efficient to use **Prune** and a batch file to do the same job. This paradigm will allow users to do permutation tests with any of the programs. Of course, you will need access to a UNIX platform to do this.

8.11 JZMAPQTL

NAME

JZmapqtl — Multitrait mapping module

SYNOPSIS

JZmapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-E** *eqtfile*] [**-S** *srfile*] [**-t** *trait*] [**-M** *Model*] [**-c** *chrom*] [**-d** *walk*] [**-n** *nbp*] [**-w** *window*] [**-I** *hypo*]

DESCRIPTION

JZmapqtl uses (composite) interval mapping to map quantitative trait loci to a map of molecular markers and can analyze multiple traits simultaneously. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**. In addition, the program requires the results of the stepwise linear regression analysis of **SRmapqtl** for composite interval mapping.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **JZmapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **JZmapqtl** will use *qtlcart.zj*, where the *j* indicates the trait analyzed and the zero'th file contains joint mapping.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **JZmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **JZmapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed unless the trait name begins with a minus sign. If a negative number is given, then only traits beginning with a plus sign will be analyzed. The default is to analyze trait 1 only.

-E Allows the user to specify the name of the file containing results from **Eqtl**. **JZmapqtl** reads those results and uses the information to choose cofactors for some of the analysis methods.

-S Allows the user to specify the name of the file containing results from **SRmapqtl**. **JZmapqtl** reads the results and uses the information to choose cofactors for composite interval mapping model 6.

-M **JZmapqtl** assumes the specified model (see below) in the analysis. Model 3 is default.

-c The user can specify a specific chromosome for **Zmapqtl** to analyze. If zero, then all will be analyzed.

-d **Zmapqtl** walks along the chromosome at this rate. The default is to do an analysis every 2 centiMorgans along the chromosome.

-n Use this to limit the number of background parameters that **JZmapqtl** uses in composite interval mapping. This is used only with model 6. It tells **JZmapqtl** to use markers with rank no higher than specified with this option. Markers are ranked by *SRmapqtl.out* and only those markers for traits in the analysis with sufficient rank are used.

-w **JZmapqtl** blocks out a region of this many centiMorgans on either side of the markers flanking the test position when picking background markers. It is 10 by default and is only used in models 5 and 6. We refer to it as the *window size*.

-I **JZmapqtl** requires the user to specify which hypotheses to test. For backcrosses, there are two hypotheses numbered 1 and 0. Use 10 for backcrosses or a 14 to do GxE tests as well. For crosses in which there are three genotypic classes, there are hypotheses 0, 1, 2, and 3. Use 30, 31, 32 in that case or 34 to do GxE. These are explained in greater detail in the manual.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**.

EXAMPLES

```
% JZmapqtl
```

Calculates the likelihood ratio test statistics of the dataset in *qtlcart.cro* using the map in *qtlcart.map*.

```
% nice JZmapqtl -A -V -i corn.cro -m corn.map -M 6 -t 3 -I 34 &
```

Calculates the likelihood ratio test statistics of the dataset in *corn.cro* using the map in *corn.map*. Model 6 is used for analysis. This file has two traits, so specifying trait 3 means that both traits are analyzed. Hypothesis 34 means that GxE interactions are also analyzed. The program is nice'd as a courtesy to other users, and run in the background so that the user can logout and relax.

MODELS

Different parameters for the **-M** option allow for the analysis of the data assuming different models. See the **Zmapqtl** man page for explanations of models 3 and 6. These are the main analysis models available in **JZmapqtl**. You can also use model 9, which prepares an input file for use in **MultiRegress**. Mainly, it calculates the expected genotypes at the sites where it would have done analyses. The expected genotypes are calculated according to table 3.7 from the **QTL Cartographer** manual.

REFERENCES

1. Lander, E. S. and D. Botstein (1989) Mapping Mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics* **121**, 185−199.

2. Zeng, Zhao-Bang (1993) Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc. Natl. Acad. Sci., USA* **90**, 10972−10976.

3. Zeng, Zhao-Bang (1994) Precision mapping of quantitative trait loci. *Genetics* **136**, 1457−1468.

4. Jiang, Changjian and Zhao-Bang Zeng (1995) Multiple trait analysis of genetic mapping for quantitative trait loci. *Genetics* **140**, 1111−1127.

BUGS

Preplot ignores the output at present. So far, the program only does joint mapping and one form of GxE. Tests for close linkage, pleiotopic effects and other environmental effects will be added in the future.

HINTS

You can select traits to include in the analysis in three ways:

a. Set the trait to analyze at 0, so that no traits except those beginning with a `[+]` (plus sign) are analyzed. You would need to edit the .cro file first to prepend a + to all traits you wanted in the analysis.

b. Set the trait to a value in the range **[1-t]** inclusive, where t is the number of traits in the .cro file. You will then get single trait results.

c. Set the trait to a value greater than t. Then all traits will be put in the analysis, unless they begin with a minus sign `[-]`. As in a. above, you would need to edit the .cro file to minus out some traits.

You need to set the hypothesis test for SFx and RFx crosses. The default of 10 is ok for crosses in which there are only two marker genotypic classes (BCx, RIx). To test GxE, use 14. For SFx and RFx, values of 30, 31 or 32 are valid, and a 34 invokes the GxE test. Recall that we have the following hypotheses:

1. H0: a = d = 0
2. H1: a !=0 , d = 0
3. H2: a = 0 , d != 0
4. H3: a != 0, d != 0

For 30, we test H3:H0. For 31, we test H3:H0, H3:H1 and H1:H0. For 32, we test H3:H0, H3:H2 and H2:H0. 30 is probably fine for initial scans. Hypothesis 34 does a test for H3:H0 as well as the GxE.

For Model 6, be sure to run **SRmapqtl** first. Once done, **JZmapqtl** will use all markers that are significant for any of the traits in the analysis. We need to work out a better way to select the cofactors. Presently we use any markers that are significant for any trait. Also, be sure to use FB regression (Model 2 in **SRmapqtl**), or else you will end up using all markers as cofactors.

8.12 MULTIREGRESS

NAME

MultiRegress — Multiple Regression analysis of QTL data

SYNOPSIS

MultiRegress [**-o** *output*] [**-i** *input*] [**-t** *trait*] [**-c** *cat*] [**-S** *size*] [**-w** *window*] [**-u** *MaxSteps*] [**-I** *hypo*]

DESCRIPTION

MultiRegress uses stepwise regression to map quantitative trait loci. The data consist of trait values that will be mapped onto expected genotypes. The standard data set can be translated by **JZmapqtl** using model 9. Map information is encoded in the data file and thus a separate map is not needed.

You might rightly ask "What does this program add to the QTL Cartographer system?" First, it doesn't require a map like the other programs in the **QTL Cartographer** system. Second, since all of the genotypic expected values have been calculated, just about any type of cross could be analyzed. The user could write a program to calculate expected genotypes at specified sites. Finally, it can speed up the process of finding QTL when using **MImapqtl** (see the EXAMPLE section).

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **MultiRegress** will append the file if it exists, and create a new file if it does not. If not used, then **MultiRegress** will use *qtlcart.mr*.

-i This requires an input filename. This file must exist. The format is defined below. The default file is *qtlcart.zr*.

-t Use this to specify which trait **MultiRegress** will analyze. If this number is greater than the number of traits, then all traits whose names do not begin with a minus sign will be analyzed. If 0, then no traits except those beginning with a plus sign will be analyzed. The default is to analyze trait 1 only.

-c This tells **MultiRegress** whether to use the categorical traits in the analysis. Use a 1 to include categorical traits and a 0 to exclude them.

-S Requires a real number in the range 0.0 to 1.0. This is a threshold p value for adding or deleting sites from the model. The default is 0.05.

-w Requires a non-negative real number. This defines a window around a site already in a regression model to block from further analysis.

-u Requires an integer valued argument. This allows you to specify a hard limit to the number parameters in the regression analysis. By default, it is 100.

-I Requires a value of 10 or 30. The value 10 means just analyze additive effects, while the value 30 means analyze for dominance and additive effects.

INPUT FORMAT

Here is an example of the data input file:

```
#     1002909319   -filetype JZmapqtl.zr
#
#       QTL Cartographer v. 1.15e, October 2001
#       This output file (qtlcart.zr) was
#       created by JZmapqtl...
#
#       It is 13:55:19 on Friday, 12 October 2001
#
#
#  This output of JZmapqtl is meant to be used
#  with MultiRegress
-walk         2.00      Interval distance in cM
-cross          B1      Cross
-otraits         1      Number of explanatory variables
-traits          2      Number of Traits
-positions 39           Number of positions
-n               9      Sample Size
-Trait 1   Trait.1
                        5             5.3           6.2
                        4.1           5.5
                    5.8           6.7           6.1
                    6.4
-Trait 2   Trait.2
                   15            15.3          16.2
                   24.1          25.5
                   25.8          16.7          26.1
                   16.4
-Otrait 1   Sex
        1     2     1     1     1     2     2
        1     2
```

```
-Site 1 -parameter additive -chromosome 1
-marker 1 -name c1m1  -position 0.000100  -values
                  0.5           0.5           0.5
                  0.499         0.499
                 -0.5          -0.5          -0.5
                 -0.5
-Site 2 -parameter additive -chromosome 1
-marker 1 -name c1m1  -position 0.020100  -values
           0.4984        0.4984        0.4984
           0.3026        0.3026
          -0.4984       -0.4984       -0.4984
          -0.4984
-Site 3  ......
```

The data file above was created by **JZmapqtl** with model 9. The header of the file is similar to the *qtlcart.cro* format: The first line has a long integer and specifies the filetype as *JZmapqtl.zr*. Some header information is followed by parameter definitions that include the distances between sites (same as the walking speed in **Zmapqtl**, **JZmapqtl** and **MImapqtl**), the cross, numbers of categorical traits and quantitative traits, positions (or sites) and sample size (n). The data set above has a sample size of 9 for two traits, one categorical trait and 39 genotype sites. The cross and *walk* parameters are not needed by **MultiRegress**: They are provided as a reminder of how the data set was created. The genotypes are expected QTL types based on flanking marker information.

After the parameters, the traits are listed. For each trait, there will be a token *-Trait* followed by the trait number, trait name and n real values. After the traits come the categorical traits in the same format: The token *-Otrait* is followed by the categorical trait number, name and then n integer values.

Finally, data for each of the sites are presented. Site data start with the token *-Site* followed by information about the site. The token *-parameter* is followed by the word *additive* or *dominance* indicating what expected value is calculated. The other tokens indicate which chromosome and left-flanking marker define the site, and the position is from the left telomere of the chromosome. The token *-values* is followed by n expected values of the QTL genotype at the site. This structure is repeated for each site.

EXAMPLES

Suppose we have a data set for an SF3 population in *qtlcart.zr* with three traits and the filename stem has already been set to *qtlcart*.

```
% MultiRegress -I 30 -t 4
```

Does a stepwise regression with backward elimination steps for the dataset. All three traits are analyzed and both additive and dominance effects are estimated.

One can also speed up the process of finding QTL using multiple interval mapping. The core algorithms of **MImapqtl** are very compute intensive. As an example, using **MImapqt**

to search for QTL *de novo* takes 934 seconds on a Macintosh G4 with an 867 MHz processor. Contrast this with the following sequence:

```
% JZmapqtl -X mletest -M 9 -A -V
% MultiRegress  -A -V
% Rqtl -i mletest.mr -o mletestPhase0.mqt
% MImapqtl -p 1  -IsMPrtseC
```

Converting the data with **JZmapqtl** and searching for putative QTL with **MultiRegress** yields a starting point for **MImapqtl**. **Rqtl** translates the output of **MultiRegress** so that **MImapqtl** can use it as an initial model. The *-p 1* option tells **MImapqtl** to set the phase variable to one, and thus the program expects the input model to be in *mletestPhase0.mqt*. This method takes about 25 seconds and comes up with a very similar set of QTL as using **MImapqtl** to search from scratch.

BUGS

If you have a multitrait data set, then use all of the traits. Convert them all with **JZmapqtl** by using a trait value greater than the number of traits, and be sure that none of the traits have names beginning with a minus sign.

8.13 MIMAPQTL

NAME

MImapqtl — Multiple Interval mapping module

SYNOPSIS

MImapqtl [**-o** *output*] [**-i** *input*] [**-m** *mapfile*] [**-E** *inputmodel*] [**-O** *outputmodel*] [**-t** *trait*] [**-q** *QTL*] [**-k** *Epi*] [**-d** *walk*] [**-S** *stop*] [**-L** *threshold*] [**-I** *workcode*] [**-p** *phase*]

DESCRIPTION

MImapqtl uses multiple interval mapping to map quantitative trait loci to a map of molecular markers. It requires a molecular map that could be a random one produced by **Rmap**, or a real one in the same format as the output of **Rmap**. The sample could be a randomly generated one from **Rcross** or a real one in the same format as the output of **Rcross**. In addition, the program can use an initial genetic model. This model will most likely be produced by running **Eqtl** on the results of a **Zmapqtl** run, but could be the results of a prior run of **MImapqtl**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **MImapqtl** will append the file if it exists, and create a new file if it does not. If not used, then **MImapqtl** will use *qtlcart.mim*.

-i This requires an input filename. This file must exist. It should be in the same format as the output of **Rcross**. The default file is *qtlcart.cro*.

-m **MImapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-t Use this to specify which trait **MImapqtl** will analyze. If this number is greater than the number of traits, then all traits will be analyzed unless the trait name begins with a minus sign. If a negative number is given, then only traits beginning with a plus sign will be analyzed. The default is to analyze trait 1 only.

-E Allows the user to specify the name of the file containing the genetic model for input. This file should be in the format of *Rqtl.out* and produced by **Rqtl**, **Eqtl** or **MImapqtl**. A

new model will be placed in the file specified with the **-O** option. For an initial analysis (phase 0), this will default to *qtlcarti.mqt*.

-O Allows the user to specify the name of the file containing the genetic model for output. For an initial analysis (phase 0), this will default to *qtlcarto.mqt*.

-q **MImapqtl** has a limit to the number of QTL it can analyze. For 32 bit machines, this is 19 QTL. For 64 bit machines, this can be 31. If you set this to a number higher than that allowed, it will be reset to the maximum allowed for the machine type.

-k The user can specify the maximum number of epistatic terms allowed in the model.

-d **MImapqtl** walks along an interval at this rate during the refinement of QTL positions and the search for more QTL.

-S Requires an integer value to indicate the information criterion for declaring the presence or absence of a parameter. Information criteria are explained below.

-L Requires a real value to indicate the threshold for adding or deleting parameters to a model. Comparisons are made based on the information criterion function specified with the **-S** option above. The default is 3.84. If this is set too low, the program will continue to find QTL until it hits the upper limit. If set too high, it will not find any QTL.

-I Requires an eight character string that codes for what the program should do. See below for more explanation.

-p is used with an integer to specify the *phase* of the analysis. See below for more explanation.

FILES

You will need a map of molecular markers, *qtlcart.map* and a data set, *qtlcart.cro*. You can also specify an initial genetic model, *qtlcart.qtl*.

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**. If you use an initial genetic model, it should be of the same format as an **Rqtl** output file. The output of **Eqtl** will also have such a model, as will the output of **MImapqtl** itself.

MImapqtl can produce three types of output files. Most results will be put in the *qtlcart.mim* file, while the **Rqtl** formatted output file will be *qtlcart.mqt*. If you choose to calculate residuals, they will be placed in *qtlcart.res*, which will have the same format as an **Rcross** output file.

NOTES

ANALYSIS

If **MImapqtl** is invoked and an initial model is provided, it will do the following seven steps:

Step 1.
Initial Model

Read in the initial model and convert it to a usable format. The model is read from the file specified by the **-E** option and must exist.

Step 2.
Parameter Estimation

Estimate all parameters in the initial model. If used, then the initial model simply specifies the positions of the QTL.

Step 3.
Refine Positions

Refine the estimates of the positions of all QTL in the initial model. This refinement occurs in the interval where the QTL resides. This is not an option to search other intervals for QTL.

Step 4.
Test parameters

Test each parameter in the initial model for significance. This follows a backward elimination procedure, and those parameters that do not lead to a significant improvement in fit are dropped. The threshold for dropping parameters is specified by the **-L** option. The information criterion is calculated for the model with and without the tested parameter, and the difference must be greater than the threshold for the effect to be retained.

Step 5.
Search for QTL

Search for more QTL. This follows a forward stepwise procedure, whereby the genome is scanned, the most likely place for a new QTL is determined, and if it results in a significant improvement, is retained.

Step 6.
Epistasis search

Search for epistatic interactions between the QTL in the model. This will do all pairwise combinations of the QTL that survive steps 4 and 5.

Step 7.
Calculate predictions

Calculate breeding values, the Variance-Covariance matrix and R2 values for the parameters.

If the user specifies that no initial model is to be used, then the analysis starts with step 5 above.

WORK CODE

The **Work Code** must be specified with an 8 letter string. Each letter in the string is a flag to tell the program whether to do a certain step. Some of the flags have options to modify the behavior of that step. The 8 letter string starts from position 0. The remaining positions (1−7) correspond to the steps given in the previous section.

Position 0.
Scan flag

This can take on values **S** or **s**. If **S**, then **MImapqtl** will go into scan mode. It will do one pass in the search for QTL phase, and print out positions and a likelihood profile to the output file. The user can then plot the values and decide where to place a new QTL.

Position 1.
Model flag

Tells **MImapqtl** whether to use the initial model specified with the **-E** option. If **M**, then use the model, and if **m**, don't use it. If you use **m**, then you should also specify **prt** in positions 2, 3 and 4. For example, **smprtSEC** would make sense: It would search for QTL *de novo*.

Position 2.
Parameter flag

Use a **P** here if you want **MImapqtl** to re-estimate the paramters in the initial model. Use a **p** if you want to skip this step. The case of this position should almost always match that of position 1.

Position 3.
Refine postion flag

Use an **R** here if you want **MImapqtl** to refine the position estimates in the initial model. Use an **r** if you want to skip this step. If you don't have an initial model, then this should be **r**. You can also extend the refinement of position to the immediate adjacent intervals by using **A** in this position.

Position 4.
Test flag

Use a **T** here if you want **MImapqtl** to test the significance of the parameters in the initial model. Use a **t** if you want to skip this step. If you don't have an initial model, then this should be **t**. You can use **D** in place of **T** in order to test dominance effects only, but t his is only relevant with three marker classes. Finally, if you want to test any existing epistatic interactions, then use **E**.

Position 5.
Search flag

Use an **S** here if you want **MImapqtl** to search for more QTL. Use an **s** if you want to skip this step. You can also specify a **A** if you only want to search for the additive effects

of putative QTL (that is, don't search for dominance effects in Fx lines). Finally, if you use a **D** here, **MImapqtl** will only search for dominance effects at QTL locations that don't already have them.

Position 6.

Epistasis flag

Use an **E** here if you want **MImapqtl** to search for epistatic effects. Use an **e** if you want to skip this step. By default, the **MImapqtl** does a forward stepwise search for epistatic terms. If you want to try a backward elimination approach, use a **B** in this position, but be aware that if there are too many epistatic terms, the request will be ignored in favor of a forward search. Finally, a **U** in this postion will do a backward elimination approach but the limit to the number of parameters will be the sample size minus one.

Position 7.

Covariance flag

Use a **C** here if you want **MImapqtl** to calculate the variance-covariance matrix, R2 values and breeding values for the final model. Use a **c** if you want to skip this step. If you specify an **R** in this spot, then for the current model, the residuals for the trait being analyzed are calculated and used as the new trait values. These residuals are written to a file *stem.res* where *stem* is the filename stem.

The default string is **smprtSeC**, which tells **MImapqtl** to scan for QTL without an initial model, where the the additive and dominance effects are treated as a unit.

INFORMATION CRITERIA

See Kao, Zeng and Basten (1999) for more detailed information on the information criteria. We use

```
IC(k) = -2(log(L) - k c(n) / 2)
```

where L is the likelihood for a k-parameter model and *log* is the natural log function. The penalty function *c(n)* takes one of six forms:

1. c(n) = log(n)

2. c(n) = 2

3. c(n) = 2 log(log(n))

4. c(n) = 2 log(n)

5. c(n) = 3 log(n)

6. c(n) = 0

Use the numbers above with the *-S* option to indicate which information criterion you want to use. If you use penalty functions 1 through 5 above, then you should also specify a threshold of 0.0 with the *-L* function. Penalty function 6 is equivalent to no penalty function and requires an experimentwise threshold value that might be obtained via a permutation test.

ANALYSIS PHASE

MImapqtl can read a genetic model and proced with various tasks as explained above. This leads to the idea of repeating the analysis with the results of a previous run of the program. One can think of doing the analysis in steps or *phases*. The default is to set the phase to zero. If the phase is set to zero, then the default input file for a genetic model is *qtlcart.eqt*, the output is *qtlcart.mqt* and the general output file is *qtlcart.mim*. At the end of the analysis, the phase remains 0.

If one sets the phase to a positive integer (generally starting with 1), then the default input is to assume that input and output files follow a rule. Assume the filename stem is *qtlcart* and the phase is i. The input genetic model will be set to *qtlcartPhase*i-1.*mqt*, the output genetic model will be written to *qtlcartPhase*i.*mqt*, and the general output file will be *qtlcartPhase*i.*mim*. At the end of the analysis, the phase variable i will be incremented by one and recorded in the *qtlcart.rc* file. This makes it easier for the program (and the user) to keep track of previous and current results. Also note that if you used an **R** in position 7 of the workcode, then the output file containing a new dataset with the residuals replacing the trait values will be put in *qtlcartPhase*i.*res*.

EXAMPLES

```
% MImapqtl -I smprtSeC
```

Calculates the best model for the dataset in *qtlcart.cro* using the map in *qtlcart.map* and the model in *qtlcart.eqt*, but only searches for main effects (additive and dominance).

Here is a sequence using the example dataset *mletest.cro* along with its map file *mletest.map*, both of which come with the programs. Assume that these two files have been placed in an empty subdirectory which is now the current working directory.

```
% MImapqtl -A -V -I smprtSeC  -L 0.0 -S 1 -p 1 -X mletest &
% MImapqtl -A -V -I sMPrTseC    &
% MImapqtl -A -V -I sMPRtseC    &
% MImapqtl -A -V -I sMPrtSeC    &
% MImapqtl -A -V -I sMPrtsBC    &
```

The first invocation sets the filename stem, the information criterion and threshold for adding parameters and indicates that it is phase 1. The **-I** option tells **MImapqtl** to search for additive QTL. The second invocation tests each QTL found in the first phase. The third step refines the positions of all remaining QTL. The fourth step searches for more QTL (and probably won't find any). The fifth step searches for interactions between the identified putative QTL. The phase variable is updated after each step, so **MImapqtl** knows where to find the results from the previous step.

REFERENCES

1. Kao, Chen-Hung and Zhao-Bang Zeng, (1997) General formulae for obtaining the MLEs and the asymptotic variance-covariance matrix in mapping quantitative trait loci when using the EM algorithm. *Biometrics* **53**, 653−665.

2. Kao, Chen-Hung and Zhao-Bang Zeng, (2000) Modeling epistasis of quantitative trait loci using Cockerham's model. *Theoret. Pop. Biol.* in press.

3. Kao, Chen-Hung, Zhao-Bang Zeng and R. Teasdale (1999) Multiple interval mapping for quantitative trait loci. *Genetics* **152**, 1203−1216.

4. Zeng, Zhao-Bang, Chen-Hung Kao and Christopher J. Basten (1999) Estimating the genetic architecture of quantitative traits. *Genetical Research, Camb.* **74**, 279−289.

CAVEATS

We are still doing some simulations to determine the best information criterion to use. At present, the defaut of 1 with a threshold of 0.0 seems to work well for a variety of data sets. If the defaut detects no QTL, then you might try information criteria 2, 3 or 6.

BUGS

Still under development: We hope to add the joint analysis of multiple traits in multiple environments. We are also working on the output formats.

The **A** option for refining positions in the work code does not yet behave correctly. It tends to place all putative QTL at the left flanking marker of an interval. If you use this option, then re-run **MImapqtl** with an **R** in position four to better refine the position estimate of the QTL.

8.14 PREPLOT

NAME

Preplot — Process results of LRmapqtl and Zmapqtl for input to gnuplot

SYNOPSIS

Preplot [**-o** *output*] [**-m** *mapfile*] [**-l** *lrfile*] [**-z** *zfile*] [**-q** *qtlfile*] [**-S** *threshold*] [**-T** *terminal*] [**-H** *hypo*] [**-L** *lod*]

DESCRIPTION

Preplot reformats the output of **LRmapqtl** and **Zmapqtl** so that it can be plotted by **GNUPLOT**. It requires a molecular map that was used in the analysis of the data with **LRmapqtl** and **Zmapqtl**.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename stem for output. **Preplot** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Preplot** will use *qtlcart*. The **GNUPLOT** file will be *qtlcart.plt* in that case.

-m **Zmapqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-l This requires an input filename. This file must exist. It should be in the same format as the output of **LRmapqtl**. The default file is *qtlcart.lr*.

-q This requires an input filename. This file may or may not exist. It should be in the same format as the output of **Rqtl**. The default file is *qtlcart.qtl*.

-z This requires an input filename. This file must exist. It should be in the same format as the output of **Zmapqtl**. The default file is *qtlcart.z*.

-T Allows the user to set the output terminal. Valid options can be found in the **GNUPLOT** manual. The default is *x11* on UNIX, *mac* for Macintosh and *windows* for MS-Windows.

-S When given an argument, **Preplot** will use this significance threshold. It is 3.84 by default.

-H **Preplot** will get results for this hypothesis test from the **Zmapqtl** outputfile. Test 1 is the default, which is the only value for a backcross.

-L If given an argument of 1, **Preplot** will output LOD scores instead of the LR test statistics.

EXAMPLES

```
% Preplot -L 1
```

Preplot will automagically reformat your results to be plotted by **GNUPLOT**, converting the likelihood ratio test statistics into LOD scores along the way.

REFERENCES

1. T. Williams and C. Kelley (1993) GNUPLOT: An Interactive Plotting Program. Version 3.5

CAVEATS

Preplot will search for the ouput files from **Rqtl**, **Eqtl** and **MImapqtl** and try to process them. If you don't want them processed, rename or move the files. Note that **Preplot** only searches for files with names *stem.eqt, stem.qtl* and *stem.mqt*, where *stem* is the filename stem.

BUGS

Preplot ignores **JZmapqtl** output.

8.15 EQTL

NAME

Eqtl — Summarize the output of **Zmapqtl**

SYNOPSIS

Eqtl [**-o** *output*] [**-z** *zmapfile*] [**-m** *mapfile*] [**-t** *trait*] [**-M** *Model*] [**-a** *size*] [**-S** *threshold*] [**-L** *lod*] [**-I** *workcode*]

DESCRIPTION

Eqtl reformats the prodigous output of **Zmapqtl**. The output file has a section that is suitable for input to **Rcross**. There are other sections to the output that are more readable. **Eqtl** can also detect whether a bootstrap, permutation or jackknife analysis was performed and process the interim files produced by those analyses.

OPTIONS

See **QTLcart(1)** for more information on the global options **-h** for help, **-A** for automatic, **-V** for non-Verbose **-W path** for a working directory, **-R file** to specify a resource file, **-e** to specify the log file, **-s** to specify a seed for the random number generator and **-X stem** to specify a filename stem. The options below are specific to this program.

If you use this program without specifying any options, then you will get into a menu that allows you to set them interactively.

-o This requires a filename for output. **Eqtl** will overwrite the file if it exists, and create a new file if it does not. If not used, then **Eqtl** will use *qtlcart.eqt*.

-z This requires an input filename. This file must exist. It should be in the same format as the output of **Zmapqtl**. The default file is *qtlcart.z*.

-m **Eqtl** requires a genetic linkage map. This option requires the name of a file containing the map. It should be in the same format that **Rmap** outputs. The default file is *qtlcart.map*.

-H Allows the user to specify which hypothesis test results to process. Use values 10 or 14 for data with two marker classes, and 30, 31, 32, 34 for those with three marker classes.

-S Tells **Eqtl** the significance threshold. It assumes that the test statistic is significant if greater than this value. It is 3.84 by default.

-a **Eqtl** uses the specified size (alpha) to determine the significance threshold from the experiment-wise permutation results. If used, the **-S** option is ignored, and the significance threshold is set and saved from the experiment-wise permutation test results. The size is 0.05 by default.

-L If used with argument 1, it causes LOD scores to be output rather than the LR statistics. It is 0 by default.

-I This should be followed by a string of no more than eight characters. It allows the user to control which files **Eqtl** will process. The default is *ZM*, which means process the results of **Zmapqtl** and **JZmapqtl** (the *M* is for multitrait). You can also add the letters *P* for permutations, *B* for bootstraps or *J* for jackknives. Prior to implementing this option, the default was *PBJZM*. Note that the order of the letters indicates the order of processing: If you want to use the threshold from a permutation test, then the *P* should precede the *Z* and *M* values. Also, you need to have at least one action.

INPUT FORMAT

The input format of the molecular map should be the same as that of the output format from the program **Rmap**. The input format of the individual data should be the same as the output format of the program **Rcross**. The other files should have been created by **Zmapqtl**. Take care that **Zmapqtl** completed its analysis: An incomplete *qtlcart.z* file can cause **Eqtl** to crash.

EXAMPLES

```
% Eqtl -m example.map -z example.z -S 13.2
```

reprocesses the results of example.z based on the map in example.map using a significance threshold of 13.2. If you had run **JZmapqtl**, then those results would also have been processed.

BOOTSTRAPS, JACKKNIVES AND PERMUTATIONS

If **Zmapqtl** was used to do a bootstrap experiment or a permutation test, then there will be interim results files. With the default filename stem and model 3, there will be files *qtlcart.z3c* and *qtlcart.z3e* if a permutation test was done, and *qtlcart.z3a* if a bootstrap was done. Using the work code *PBZM* will direct **Eqtl** to detect these files and processes their results. It will open a qtlcart.z3e file and determine an experimentwise threshold based on the size specified with the **-a** option.

If the *qtlcart.z3a* file exists, then **Eqtl** opens it and computes the means and standard deviations, at each test site, of the likelihood ratio test statistic, additive effect and dominance effect. The results are printed to *qtlcart.z3b*.

The jackknife procedure produces a *qtlcart.z3i* file. Adding a *J* to the work code directs **Eqtl** to computes the means and standard deviations of the likelihood ratio test statistic, additive effect and dominance effect at each test site. The results are printed to *qtlcart.z3j*.

REFERENCES

BUGS

If the resource file indicates that there are more than one trait, then Eqtl will try to estimate positions and additive effects for all the traits. This will happen even if no analysis was done on the extra traits. The output file will then have some null estimates.

When doing a jackknife with **Zmapqtl**, the user should check that the file ending in the letter i is truely the last version of the interim jackknife file. **Zmapqtl** switches between a file ending in i and another ending in j, so check both and move the j file onto the i file if required.

If you set the significance threshold too high, then **Eqtl** may find no QTL in the *qtlcart.z* output. If this is the case, then **Eqtl** will crash.

8.16 EXAMPLES

NAME

Examples — Example data sets for **QTL Cartographer**

DESCRIPTION

The files in the *example* subdirectory are example data sets that can be used with **QTL Cartographer**. Some of these data sets have been published, so you can compare your results with those in the publications.

If you are on a Unix system, you can use make to create a test subdirectory, with a different subdirectory for each set of example files. Be sure to check the *Makefile* for proper **rm** and **cp** commands. You will also want to set the target location for the test directory.

COPYING

The file COPYING contains the GNU GENERAL PUBLIC LICENSE (Version 2, June 1991) that governs the usage of QTL Cartographer. It does not cover the usage of the data sets in this directory. You can use these data sets to learn about or teach QTL Cartographer. Contact the people who generated the data if you want to use their data sets for other purposes. We are grateful to the individuals who created the data for allowing us to distribute their data sets.

INPUT FORMAT

The genetic linkage map files will generally have to be converted using **Rmap**, and the data sets should be run through **Rcross**. For example, the *sample* subdirectory contains files *sample.mps* and *sample.raw*. You will convert them with the commands

```
% Rmap -i sample.map -X sample
% Rcross -i sample.raw
```

The above commands will output files *sample.map* and *sample.cro*.

The only exception to this general rule is the *mletest* files, which are already in **QTL Cartographer** format.

EXAMPLES

Here is a list of the example data files contained in this directory.

1. *mletest.map* and *mletest.cro* contain the simulated data generated and used by Zhao Bang Zeng in his original paper on composite interval mapping (Zeng, 1994).

   These are ready for use with QTL Cartographer. There is also a file mletestq.inp which contains the true positions and effects of the QTL used in the simulation. It can be translated with Rqtl

   ```
   %  Rqtl -i mletestq.inp -X mletest
   ```

and then you can proceed with the analysis

```
% Qstats
% LRmapqtl
% SRmapqtl
% Zmapqtl
% Zmapqtl -M 6
% Eqtl -S 12.0
% Preplot
% gnuplot mletest.plt
```

You can modify the above commands to explore how to use **QTL Cartographer**.

2. The files *realdatc.inp* and *realdatm.inp* contain real data from Horvat and Medrano (1995). These should be converted with Rmap and Rcross before use with **QTL Cartographer**.

```
% Rmap -i realdatm.inp -X realdat
% Rcross -i realdatc.inp
% Qstats
% LRmapqtl
% SRmapqtl
% Zmapqtl
% Zmapqtl -M 6
% Eqtl -S 12.0
% Preplot
% gnuplot realdat.plt
```

3. **MAPMAKER/EXP** and **MAPMAKER/QTL** (Lander *et al* 1987) come with a sample data set which we have included in the **QTL Cartographer** distribution. *sample.raw* is the sample data set and *sample2.raw* is essentially the same data but the trait foo=*log*(weight) has been calculated and put in the file. The *sample.raw* file was run through **MAPMAKER/EXP** to create the *sample.mps* file in preparation to use with **QTL Cartographer**.

   These should be converted with Rmap and Rcross before use with **QTL Cartographer**.

```
% Rmap -i sample.mps -X sample
% Rcross -i sample2.raw
```

   or

```
% Rmap -i sample.mps -X sample
% Rcross -i sample2.raw
```

   and then you can run the rest of the analyses

```
% Qstats
% LRmapqtl
% SRmapqtl
% Zmapqtl
% Zmapqtl -M 6
% Eqtl -S 12.0
% Preplot
% gnuplot sample.plt
```

4. Nuhzdin *et al* (1997) present an analysis of sex-specific QTL Data coming from Trudy MacKay's lab. These data are contained in nuzhdinm.inp (the map) and nuzhdinc.inp (the data). They should be converted with **Rmap** and **Rcross** before use with **QTL Cartographer**.

   ```
   % Rmap -i nuzhdinm.inp -X nuzhdin
   % Rcross -i nuzhdinc.inp
   ```

5. The data from Weber *et al* (1999) and Weber *et al* (2001) are in the files *weber519c.inp* and *weber701c.inp* and there is a map file for each. The numbers in the file name indicate the sample sizes. For the sample of size 519, convert the files prior to analysis:

   ```
   % Rmap -i weber519m.inp -X weber519
   % Rcross -i weber519c.inp
   ```

   For sample of size 701, use *weber701m.inp* and *weber701c.inp*

   ```
   % Rmap -i weber701m.inp -X weber701
   % Rcross -i weber701c.inp
   ```

   and proceed with the analyses.

6. Another data set from Trudy Mackay's lab was presented by Vieira *et al* (2000). The data are in *vieiram.inp* and *vieirac.inp*. The by-now usual conversion should preceed the analysis.

   ```
   % Rmap -i vieiram.inp -X vieira
   % Rcross -i vieirac.inp
   ```

7. There are four data sets from Cathy Laurie's lab with analyses presented by Zeng *et al* (2000).

   The map file (*lauriem.inp*) should be used with all four of the data sets. The data sets include two backcrosses between *Drosophila simulans* and *D. mauritiana*. Each backcross has two independent samples. Think of *D. mauritiana* as parental line 1 (P1) and *D. simulans* as parental line 2 (P2). Then, we have B1 crosses (F1 x P1) and B2 crosses (F1

x P2). The first B1 cross has a sample size of 192 and is in the file *lauriem4c.inp*, while the second (*lauriem6c.inp*) has a sample size of 299. The first B2 cross has a sample size of 184 (*lauries4c.inp*), while the second (*lauries6c.inp*) has a sample size of 287.

All need to be converted.

```
% Rmap -i lauriem.inp -X lauriem4
% Rcross -i lauriem4c.inp

% Rmap -i lauriem.inp -X lauries4
% Rcross -i lauries4c.inp

% Rmap -i lauriem.inp -X lauriem6
% Rcross -i lauriem6c.inp

% Rmap -i lauriem.inp -X lauries6
% Rcross -i lauries6c.inp
```

You might want to create a separate subdirectory for each data.

8. Arabidopsis data from Rauh *et al* (2002) are in *rauhall.inp* with the map in *rauhmap.inp*. There are four different traits raised in four different environments. Each of the traits has been split into a file of its own. Rootmass data are in *rauhrm.inp*, rootlength data in *rauhrl.inp*, arialmass data in *rauham.inp* and the ratio of root to arial mass in *rauhratio.inp*. Again, they need to be converted, and each dataset should its own subdirectory. For example, if you have all the files in one subdirectory, you could analyze the rootmass data as follows:

```
% mkdir rootmass
% cp rauhmap.inp rauhrm.inp rootmass
% cd rootmass
% Rmap -i rauhmap.inp -X rootmass
% Rcross -i rauhrm.inp
% Qstats
% LRmapqtl -t 5
% SRmapqtl -t 5
% Zmapqtl -t 5
% Zmapqtl -t 5 -M 6
% JZmapqtl -t 5 -I 14
% Eqtl -S 12.0 -H 14
% Preplot
% gnuplot rootmass.plt
```

REFERENCES

1. Horvat, S. and J. F. Medrano (1995) Interval mapping of *high growth (hg)*, a major locus that increases weight gain in mice. *Genetics* **139:**1737−1748.

2. Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln and L. Newburg (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* **1:** 174−181.

3. Nuzhdin, S. V., E. G. Pasyukova, C. L. Dilda, Z. B. Zeng and T. F. C. Mackay (1997) Sex-specific quantitative trait loci affecting longevity in *Drosophila melanogaster*. *Proceedings of the National Academy of Science USA* **94:** 9734−9739.

4. Rauh, B. L., C. Basten, and E. S. Buckler IV (2002) Quantitative trait loci analysis of growth response to varying nitrogen sources in *Arabidopsis thaliana*. *Theor Appl Genet* **104:** 743−750.

5. Vieira, C., E. G. Pasyukova, Z. B. Zeng, J. B. Hackett, R. F. Lyman and T. F. C. Mackay (2000) Genotype-environment interaction for quantitative trait loci affecting lifespan in *Drosophila melanogaster*. *Genetics* **154:** 213−227.

6. Weber, K., R. Eisman, S. Higgins, L. Kuhl, A. Patty, J. Sparks and Z. B. Zeng (1999) An analysis of polygenes affecting wing shape on chromosome three in *Drosophila melanogaster*. *Genetics* **153:** 773−786.

7. Weber, K., R. Eisman, S. Higgins, L. Morey, A. Patty, M. Tausek and Z. B. Zeng (2001) An analysis of polygenes affecting wing shape on chromosome 2 in *Drosophila melanogaster*. *Genetics* **159:** 1045−1057.

8. Zeng, Z. B. (1994) Precision mapping of quantitative trait loci. *Genetics* **136:** 1457−1468.

9. Zeng, Z. B., J. Liu, L. F. Stam, C. H. Kao, J. M. Mercer and C.C. Laurie (2000) Genetic architecture of a morphological shape difference between two Drosophila species. *Genetics* **154:** 299−310.

Bibliography

Akaike, H. (1969). Fitting autoregressive models for prediction. *Ann. Institute Stat. Math. 21*, 243–247.

Basten, C. J., B. S. Weir, and Z.-B. Zeng (1994). Zmap–a QTL cartographer. In C. Smith, J. S. Gavora, B. B. J. Chesnais, W. Fairfull, J. P. Gibson, B. W. Kennedy, and E. B. Burnside (Eds.), *Proceedings of the 5th World Congress on Genetics Applied to Livestock Production: Computing Strategies and Software*, Volume 22, Guelph, Ontario, Canada, pp. 65–66. Organizing Committee, 5th World Congress on Genetics Applied to Livestock Production.

Broman, K. W. (1997). *Identifying quantitative trait loci in experimental crosses*. Ph. D. thesis, UC Berkeley, Department of Statistics.

Carter, T. C. and D. S. Falconer (1951). Stocks for detecting linkage in the mouse and the theory of their design. *J. Genet. 50*, 307–323.

Churchill, G. A. and R. W. Doerge (1994). Empirical threshold values for quantitative trait mapping. *Genetics 138*, 963–971.

Cockerham, C. C. (1954). An extension of the concept of partitioning hereditary variance for analysis of covariances among relatives when epistasis is present. *Genetics 39*, 859–882.

Cockerham, C. C. and Z. Zeng (1996). Design III with marker loci. *Genetics 143*, 1437–1456.

Doerge, R. W. and G. A. Churchill (1996). Permutation tests for multiple loci affecting a quantitative character. *Genetics 142*, 285–294.

Doerge, R. W., Z. Zeng, and B. S. Weir (1997). Statistical issues in the search for genes affecting quantitative traits in experimental populations. *Stat. Sci. 12*, 195–219.

Dongarra, J. J., C. B. Moler, J. R. Bunch, and G. W. Stewart (1979). *LINPACK Users' Guide*. Philadelphia, PA: SIAM.

Falconer, D. S. and T. F. C. MacKay (1996). *Introduction to Quantitative Genetics*. Essex, UK: Longman Group Limited.

Felsenstein, J. (1979). A mathematically tractable family of genetic mapping functions with different amounts of interference. *Genetics 91*, 769–775.

Fisch, R. D., M. Ragot, and G. Gay (1996). A generalization of the mixture model in the mapping of quantitative trait loci for progeny from a bi-parental cross of inbred lines. *Genetics 143*, 571–577.

Haldane, J. B. S. (1919). The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet. 8*, 299–309.

Hannan, E. J. and B. G. Quinn (1979). The determination of the order of an autoregression. *J. Royal Stat. Soc., Series B 41*, 190–195.

Horvat, S. and J. F. Medrano (1995). Interval mapping of *high growth (hg)*, a major locus that increases weight gain in mice. *Genetics 139*, 1737–1748.

Jiang, C. and Z. Zeng (1995). Multiple trait analysis of genetic mapping for quantitative trait loci. *Genetics 140*, 1111–1127.

Jiang, C. and Z. Zeng (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica 101*, 47–58.

Kao, C. and Z. Zeng (1997). General formulae for obtaining the MLEs and the asymptotic variance-covariance matrix in maping quantitative trait loci. *Biometrics 53*, 653–665.

Kao, C., Z. Zeng, and R. Teasdale (1999). Multiple interval mapping for quantitative trait loci. *Genetics 152*, 1203–1216.

Karlin, S. (1984). Theoretical aspects of genetic map functions in recombination processes. In A. Chakravarti (Ed.), *Human Population Genetics: The Pittsburgh Symposium*, New York, pp. 209–228. Van Nostrand Reinhold.

Kosambi, D. D. (1944). The estimation of map distances from recombination values. *Ann. Eugen. 12*, 172–175.

Lander, E. S. and D. Botstein (1989). Mapping mendelian factors underlying quantitative traits using rflp linkage maps. *Genetics 121*, 185–199.

Lander, E. S., P. Green, J. Abrahamson, A. Barlow, M. Daley, S. Lincoln, and L. Newburg (1987). MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics 1*, 174–181.

Lincoln, S., M. Daly, and E. S. Lander (1992). Constructing genetic maps with MAPMAKER/EXP 3.0. Technical report, Whitehead Institute Technical Report.

Liu, B. (1998). *Statistical Genomics: Linkage, Mapping and QTL Analysis*. Boca Raton, FL: CRC PRess LLC.

Lynch, M. and B. Walsh (1998). *Genetics and Analysis of Quantitative Traits*. Sunderland, MA: Sinauer Associates, Inc.

Meng, X. and D. B. Rubin (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika 80*, 267–268.

Morgan, T. H. (1994). *The Theory of Genes*. New Haven, CN: Yale University Press.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1988). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, UK: Cambridge University Press.

Rao, D. C., B. J. Keats, J. M. Lalouel, N. E. Morton, and S. Lee (1979). A maximum likelihood map of chromosome 1. *A. J. Hum. Genet. 31*, 680–696.

Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Stat. 6*, 461–464.

Sturt, E. (1976). A mapping function for human chromosomes. *Ann. Hum. Genet., Lond. 40*, 147–147.

Utz, H. F. and A. Melchinger (1996). PLABQTL: A program for composite interval mapping of QTL. *J. Agric. Genomics 2*, 1.

Williams, T. and C. Kelley (1993). *GNUPLOT: An Interactive Plotting Program*. Version 3.5.

Zeng, Z. (1992). Correcting the bias of wright's estimates of the number of genes affecting a quantitative trait: A further improved method. *Genetics 131*, 987–1001.

Zeng, Z. (1993). Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc. Natl. Acad. Sci. USA 90*, 10972–10976.

Zeng, Z. (1994). Precision mapping of quantitative trait loci. *Genetics 136*, 1457–1468.

Zeng, Z., C. Kao, and C. J. Basten (1999). Estimating the genetic architecture of quantitative traits. *Genetical Research, Camb 74*, 279–289.

Index