(1) GENERAL INFORMATION ON MAPMAKER VERSION 3.0

MAPMAKER is a linkage analysis package designed to help construct primary linkage maps of markers segregating in both experimental crosses as well as simple natural populations (e.g. two and three generation nuclear families). MAPMAKER performs full multipoint linkage analyses (simultaneous estimation of all recombination fractions from the primary data) for dominant, recessive, and co-dominant (e.g. RFLP-like) markers.

A list of references can be found in the ’Release Notes’ section of the MAPMAKER manual and in the READ.ME file included with MAPMAKER.

ABOUT MAPMAKER Command

Summary: License and Contact Information for MAPMAKER

License to use this program for non-commercial purposes is provided free of charge. This software and documentation may be freely redistributed, although only under specific terms. This program is provided without any warranty of any kind. See the License Agreement included in the Release Notes and in the READ.ME file for details.

MAPMAKER 3.0 was written by Stephen E. Lincoln, Mark J. Daly, and Eric S. Lander of the Whitehead Institute for Biomedical Research and the M.I.T Center for Genome Research, Massachusetts Institute of Technology, Department of Biology. Further information is available from:

```
    MAPMAKER                                FAX: 617-258-6505
    c/o Eric Lander                      BITNET: mapm@mitwibr
    Whitehead Institute                INTERNET: mapmaker@genome.wi.mit.edu
    9 Cambridge Center
    Cambridge, Massachusetts 02142 USA
```

Source code, updates, and other information are available on the Internet by anonymous FTP from genome.wi.mit.edu.

RELEASE NOTES Information

Summary: Information on the Release Notes

Included with MAPMAKER Version 3.0 and MAPMAKER/QTL Version 1.1, you will find a Release Notes document describing:

    What MAPMAKER is
    How to get MAPMAKER
    How to Run MAPMAKER on a PC Compatible Running DOS
    How to Run MAPMAKER on a Sun SPARCStation Running SunOS
    How to Run MAPMAKER on an Apple Macintosh Running A/UX
    License Agreement
    GNU Readline and GhostScript License Information
    Where to Go from Here
    References

and other issues. The Release Notes are included in printed form in the manual, and as a READ.ME file with the program itself. Other important documentation includes:

    File   Manual Section

INSTALL.ME	Installation Guide
FEED.ME	Data Preparation Guide
CHANGE.ME	List of Changes Since Last Release, and To-Do List

STARTING MAPMAKER Information

Summary: How to Start MAPMAKER and Available Options

Exactly how you start MAPMAKER depends on how it was installed in your system -- see the Installation Guide included with MAPMAKER for details. Usually, you will want to first go into the directory which contains your data files (using the "cd" command). Then, you can start MAPMAKER by simply typing at a DOS or Unix prompt:

```
    mapmaker
```

On UNIX machines, you will be able get such a prompt by opening a terminal window (Sun’s Command-Tool or A/UX’s CommandShell). Alternatively, to run MAPMAKER on a Unix system under X-Windows (or OpenWindows), you may wish to type:

```
    xmapmaker &
```

As a third alternative, depending on your system, you may be able to start MAPMAKER instead by simply clicking on a particular icon or file. Again, see the Installation guide for details.

You may supply additional instructions to MAPMAKER on the command line. On a Unix system (including Suns and A/UX), these include:

-simple
Do not use any fancy terminal capabilities (e.g., highlighting, screen clearing, command editing, etc.). On DOS, this means that ANSI.SYS is not required. On Unix, this helps with unusual or buggy terminals and terminal emulators.

-nomore
MAPMAKER produces output one screenful at a time (with "Hit Return for More" breaks in between) from certain commands, including ’help’, ’translate’, and all ’list’ and ’show’ commands. This is useful on DOS systems or on Unix terminals which do not remember text that has scrolled off the screen. The ’-nomore’ option turns this off: "more" breaks never interrupt output. Regardless of whether you use this option, "more" breaks are never used by time-consuming analytical commands, regardless of the amount of output they produce.

-clear
Clearing the screen can make some output faster and prettier, although it wreaks havoc with some terminals or programs that try to remember text that has scrolled off the screen. By default, we allow the screen to be cleared on DOS systems, and not on Unix systems. The ’-clear’ option precisely reverses these assumptions.

-photo xxx
MAPMAKER output is copied to the named file (xxx). If the file exits, MAPMAKER output is appended onto the end of it.

-out xxx
MAPMAKER output is copied to the named file (xxx). If the file exists, it is first erased. Only one of "-photo" and "-out" can be used.

-load xxx
Data are loaded from the specified file (xxx) before other actions take place.

-prep xxx

Data are loaded from the named raw data file specified (xxx), as if the ’prepare data’ command had been used before other actions take place. Only one of "-load" or "-prep" can be used.

-run xxx
Commands are run from the named text file (xxx) before commands are accepted from the keyboard. If the file ends with a ’quit’ command, MAPMAKER quits without prompting for additional commands.

The options for the DOS version are identical, except that instead of dash characters (-) you should use forward slashes (/). For example, you might type:

```
    mapmaker -nomore -prep newfile.raw -run /scripts/dothis.in   (UNIX)
    mapmaker /nomore /prep newfile.raw /run \scripts\dothis.in   (DOS)
```

The Unix and DOS ’<’, ’>’, and ’|’ characters, indicating input and output redirection, as well as the Unix ’&’ argument for running a job in the background, all work with MAPMAKER as well.

A small number of DOS or Unix environment variables are also used. On a Unix system, you set these with a ’setenv’ command in your ’.cshrc’ file (assuming you use ’csh’ as your shell) or in a shell script (e.g., ’xmapmaker’). Under DOS, you set these with a SET command in your AUTOEXEC.BAT file or in some other batch file (e.g., MAPMAKER.BAT). The variables used include:

MAPM_LIB
The directory in which MAPMAKER will find it’s help files and other items it needs. This should usually be set to something like "/home/joe/mapmaker" or "/usr/local/mapmaker" on Unix, or "C:\MAPMAKER" on DOS. This variable must be set to use the on-line help feature: see the Installation Guide for details.

LINES
The number of lines the screen/terminal/window can display at once. MAPMAKER tries to figure this out on its own, although if the LINES variable is set, its value takes precedence. Set LINES if you use the DOS "mode con:lines=xx" command to change the number of lines of text displayed on EGA and VGA monitors or under Windows 3.1.

TERM
The type of terminal in use. MAPMAKER recognizes only a small number of terminal type names, even though many terminals are ANSI (and thus MAPMAKER) compatible. Try setting "TERM" to "ansi" or "vt100" if MAPMAKER is confused (for example, if it does not display "Hit return for more" in reverse-video when you type "help"). Usually set automatically on Unix systems and usually not needed on DOS systems.

SHELL (Unix) or COMSPEC (DOS)
The name of the program to be started by MAPMAKER’s "system" command. Usually set automatically by the system.

TUTORIAL Information

Summary: Information on the MAPMAKER Tutorial

MAPMAKER is provided with a printed tutorial, which illustrates many of the basic MAPMAKER commands which you will likely use often. The data sets used in the tutorial, sample.raw and mouse.raw, are provided with the MAPMAKER distribution so that you can work along with the tutorial. The tutorial itself is not available on-line, however. See the Release Notes (or the READ.ME file) for information.

ENTERING COMMANDS Information

Summary: How to Type Commands Into MAPMAKER

When started, MAPMAKER responds with its start-up banner and a prompt for the first command:

```
    1>
```

Whenever a prompt like this is displayed, any valid MAPMAKER command may be entered. All commands consist of one or more words separated by spaces, and may be followed by one or more "arguments" to the command. Arguments might include locus numbers, computer file names, or numeric values. Input is always terminated by hitting the RETURN key. In most cases, MAPMAKER ignores upper-case/lower-case distinctions in its input (the rare exceptions include arguments which need to be case-sensitive, such as file names under Unix). For example, the following are valid MAPMAKER command entries:

```
    1> load data newrflps.data
    1> error detection on
    1> group 3.0 0.30
```

To make typing commands easier, all commands may be abbreviated by only a few characters: see 'abbreviating commands' below for details. In addition, a keyboard-based command-line feature may be enabled on your version of MAPMAKER: see 'keyboard editing' below for more details. Also, MAPMAKER remembers commands you have already typed, allowing you to quickly execute the same commands again: see the discussion of 'previous commands' below.

Note that MAPMAKER generally assumes that map distance values given as arguments greater than or equal to 0.50 are centimorgan distances (using the currently selected "centimorgan function"), while values less than 0.50 are assumed to be recombination fractions.

ABBREVIATING COMMANDS Information

Summary: How to Abbreviate Commands You Type

MAPMAKER allows you to abbreviate commands in a number of ways. In general, only enough of a command name needs to be entered in order to distinguish it from other MAPMAKER commands. Not all of the words in a multi-word command (such as "default linkage criteria") are needed, nor are all letters of any particular word, as long as the entry is not ambiguous. When more than one word of a two or three word command is used, the words must be separated by spaces. Any arguments to a command must also be separated by spaces.

For example, to turn MAPMAKER's 'auto save data' option on, you may type:

```
    1> auto on
```

Or to enter a sequence of loci, type (for example):

```
    1> seq 1 2 3 4 5 6
```

To turn 'error detection' on, you could type:

```
    1> err det on
```

Or to set the 'default linkage criteria' for pairwise analysis to a minimum LOD of 3.0 and a maximum distance of 50 cM, type:

```
    1> def link 3.0 50
```

Some frequently used commands also have one or two letter abbreviations which may be used to invoke the command very rapidly. For example, you can enter a sequence by typing:

```
    1> s 1 2 3 4
```

or load a data file by typing:

```
    1> ld myfile.data
```

On some systems, an elaborate keyboard-based editing may be enabled, making it even easier to type commands. See ’keyboard editing’ below for details.

KEYBOARD EDITING Information

Summary: Editing Commands and Sequences Using the Keyboard

On Unix systems (including Suns and A/UX), an interactive command editing facility may be optionally installed into MAPMAKER. If this feature is installed, you can recall previously typed commands using the up-arrow and down-arrow keys, edit them (e.g., to fix a mistake, add a locus to a sequence, etc.), and then hit RETURN to execute the command. The set of keys which work includes the standard ones (arrows, DELETE, BACKSPACE, etc.). Various control-key combinations also do useful thigs, although you will only find these familiar if you have used some version of the Emacs text editor. Some control keys include:

control-a	move to beginning of line
control-e	move to end of line
control-f	move forward one character
control-b	move backward one character
control-d	delete character
control-k	delete the rest of the line

The ’edit’ command can similarly be used to edit sequences: see the ’edit’ command below for information.

The keyboard editing functions are implemented using the GNU Readline package, published by the Free Software Foundation. See the Release Notes for details and licensing of this package.

The keyboard editing package (including the ’edit’ command) does not at present work on DOS systems. However, the simple default DOS 5.0 input editing features (e.g. F3 key, etc) do work, allowing you to at least edit and re-enter the last line you typed. DOSKEY has no effect on MAPMAKER’s keyboard editing (or lack thereof).

POSTSCRIPT OUTPUT Information

Summary: What to Do With MAPMAKER PostScript Graphic Output

MAPMAKER’s "draw chromosome" command produces graphic output as standard level 1 PostScipt files (text files with the extension ".ps") which can be displayed and printed in a number of ways. This issue is discussed in detail in the installation instructions in the manual (or the INSTALL.ME) file. Here we provide a brief summary:

These files print "as-is" simply by sending them to a PostScript printer (such as an Apple LaserWriter) in the usual way: On Unix (including Suns and A/UX) you can usually print these files using the ’lpr’ command (although you should not use lpr’s ’-p’ option with

PostScript files). For example:

```
    lpr mychroms.ps
```

On a DOS system, the "PRINT" command will usually work:

```
    print mychroms.ps
```

In either case, you can get a Unix or DOS prompt from which to issue this command using MAPMAKER’s ’system’ comand, described below. Both examples above assume that your software is configured correctly and in the default way. Ask your computer support people for details.

The "GhostScript" package included with MAPMAKER on some systems may be helpful for displaying PostScript files in a window under either X-Windows (including Sun’s OpenWindows), or under Microsoft Windows. Again, see the Installation Instructions for details.

(2) BASIC MAPMAKER COMMANDS

There are a few very basic MAPMAKER comands which you will likely find yourself using quite frequently. These include:

```
    'help' command, to read on-line help information
    'photo' command, to save MAPMAKER output to a text file
    'prepare data' command, to prepare a new data set for analysis
    'load data' command, to load an existing data set for analysis
    'quit' command, to save your data set and exit from MAPMAKER
```

Type 'help' followed by the command name above for more information on that command.

HELP Command (abbreviation '?')

```
Summary:     Read On-Line Help Information
Argument:    <command name or topic number>
Default:     with no arguments, display a list of all commands and topics
```

'Help' displays on line help information for MAPMAKER commands, options, parameters, and features. Type 'help' alone to see a list of available topics, commands and other information. For a general description of a numbered topic, type 'help <number>', where <number> is the displayed number of the topic. For example, for help on the MAPMAKER Error Detection feature, type:

```
    help 7
```

For help on a more specific command or feature, type 'help <name>', for example:

```
    help make chromosome
```

Abbreviations in these names are allowed in the same way as when typing a command. For example:

```
    help ma ch
```

The on-line help is an exact duplicate of the MAPMAKER reference manual.

PHOTO Command

```
Summary:     Begin Saving MAPMAKER Output to a Text File
Argument:    <file name>
Default:     with no arguments, display the current 'photo' status
```

The "photo" command is used to save a copy of the MAPMAKER session (input and output) in a text file. If you type "photo <file name>", for example,

```
    1> photo sample.out
```

all MAPMAKER input and output from that point on will be copied into the specified file (here, the file named "sample.out"). Typing "photo off" or quitting MAPMAKER terminates this process and closes the photo file. The default extension for a transcript file is ".out".
For example:

```
    2> photo june11.out
    photo is on: file is 'june11.out'

    2> try 11 12 13 14      <-      From this point on, all MAPMAKER input
            .                       and output is copied to the file
```

```
                .                        "june11.out".
                .

        5> photo
        photo is on: file is 'june11.out'

        6> photo off
        photo is off.

        7>                      <-        Input and output will no longer
                                          be copied.
```

Note that when MAPMAKER is photo-ing to a particular file, the transcript is APPENDED to the end of that file: none of the file's previous contents are lost. Successive MAPMAKER transcripts may be collected into single files in this way.

PREPARE DATA Command (abbreviation 'pd')

Summary:	Prepares a New Data Set for Analysis
Argument:	<file name>

The "prepare data" command loads a raw data set from a file you create into MAPMAKER for analysis. You may use a file name which specifies a directory in the manner supported by your operating system. You must specify the extension of the file (we sugest that you use ".raw" for raw data files). For example:

```
    1> prepare data newrflps.raw
    1> prepare data /users/joe/newrflps.raw          (UNIX)
    1> prepare data \mydata\newrflps.raw             (DOS)
```

Raw file formats are discussed in detail in the MAPMAKER manual.

When preparing a data set, MAPMAKER first reads the raw input file, and then (by default) writes out a set of "working" or prepared data files which later may be more quickly loaded into either MAPMAKER Version 3 or MAPMAKER/QTL Version 2 on later occasions. After preparing a file, you do NOT need to then load these files with the "load data" command - the "prepare data" command does this for you.

Analysis results and status information are also stored in these prepared files, so that whenever they are later saved and re-loaded (with the "load data" command), you begin using MAPMAKER or MAPMAKER/QTL right where you left off. However, if you again prepare the raw files, this information will need to be recalculated.

Whenever a data set is prepared, MAPMAKER looks for a file with the same name as the raw data file and the extension ".prep" (on UNIX) or ".pre" (on DOS). If present, commands from this file are run, serving as a useful way to setup MAPMAKER in the appropriate state for working with these particular data. This way, every time a file is prepared (e.g., if you change genotype data), it is relatively easy to start again where you left off. If you want MAPMAKER to save the working data files after these actions are completed, you should end this command file with a "save" command. If no such file is present, the default action is to simply save the working data files.

MAPMAKER will always create a working data file with the same first name as the raw file, but with a different extension instead. (For example, preparing the file "newrflps.raw" will create a file named "newrflps.data" on a Unix computer, or the file "newrflps.dat" on a PC). Many extensions are also used by MAPMAKER and MAPMAKER/QTL, including:

```
UNIX:  .data .traits .2pt .3pt .maps .qtls .out
DOS:   .dat  .tra    .2pt .3pt .map  .qtl  .out
```

We recommend that you never create files yourself for use with MAPMAKER with any of these extensions. Note that if you specify a directory with the raw file name, the prepared files will be written there also.

LOAD DATA Command (abbreviation 'ld')

Summary:	Load Data Set for Analysis
Argument:	<file name>
Default:	with no arguments, display the current data file name

The "load data" command loads a previously prepared data set into MAPMAKER for analysis. Only one data set may be loaded at any time: any data previously loaded will be forgotten. Data sets must be prepared using the current version of MAPMAKER.

The name of the data file should be given as an argument. A directory may optionally be specified in the manner supported by your system. The data file(s) MUST use the ordinary MAPMAKER extensions, as written by 'prepare data' (i.e., the genotype data file must have the extension ".data" on Unix or ".dat" on DOS, etc). For example:

```
    1> load data example
    1> load data /home/boingo/joe/chromosome7          (UNIX)
or  1> load data \joe\expt1\maize                      (DOS)
```

When a data set is loaded, MAPMAKER may also load in a number of its option settings as well as some saved results from the associated files (such data include all two and three point maps computed). This feature lets you quit and restart MAPMAKER, resuming your analysis right where you left off. Because of this however, some of MAPMAKER's option settings may change from their previous values after loading data. If you want to set any of MAPMAKER's options before running your analyses, we recommend that you do so only after loading your data set. A list of the items which are saved in this manner is provided under the description of the "auto save data" option.

SAVE DATA Command

Summary:	Save Status, Mapping Results, etc.
Argument:	<file name>
Default:	current data file name

The 'save data' command instructs MAPMAKER to immediately update all of your data files with the current option settings, any new two-point and three-point data, mapping results, etc.

This behavior is precisely the same as that used when quitting MAPMAKER with the "auto save data" option on.

QUIT Command (abbreviation 'q')

Summary: Quit from a MAPMAKER Session
No Arguments

The "quit" command is used to end a MAPMAKER session. If you are using MAPMAKER interactively (i.e., not through a batch script), and you have loaded a data file, and the "auto save" option is "on" (as it usually is), then MAPMAKER will first ask you if you want to save your data before it quits.

(3) SEQUENCE COMMAND AND RELATED FEATURES

Before using most of MAPMAKER's analysis functions, one often needs first to specify the loci and (in some cases) order(s) of those loci which should be considered in the analysis. This is usually done using MAPMAKER's 'sequence' command. This facility is actually quite powerful, and by allowing many orders to be specified with a single "sequence" command, can help you perform complex analyses simply. Commands related to the use of sequences include:

```
    'Sequence' Command
    'Expand Sequence' Command
    'History' Command (Sequence History Feature)
    'Insert', 'Delete', and 'Append' Commands
    'Edit' Command
    'Translate' Command
```

MAPMAKER also allows you to assign names to particular sequences of loci, and to refer to these sequences by name subsequently. Commands which help you do this include:

```
    'Names' Command
    'Let' Command
    'Edit' Command
    'Forget Named Sequence' Command
```

SEQUENCE Command (abbreviation 's')

Summary:	Select the Loci and Order(s) You Wish to Analyze
Argument:	<sequence>
Default:	displays the current sequence

MAPMAKER's 'sequence' command is used to select the current set and order of markers that MAPMAKER should use when executing data analysis commands. Many commands will perform their operation on all orders specified by the sequence (e.g., "map", "try", and "compare"), while a few others (such as "two point") ignore the orders specified and care only about which markers are listed in the sequence.

For example, typing

```
    1> sequence 1 2 3 4
```

will set the current sequence for analysis to be the (ordered) list of loci "1 2 3 4". We might then type an analysis command such as "map", which in this case would instruct MAPMAKER to compute the maximum likelihood map for the markers numbered 1, 2, 3, and 4 in that order.

The "sequence" command recognizes locus numbers (assigned by MAPMAKER), locus names (assigned by the user), references to previous sequences (see the "history" command for details), user defined sequence names (set with the "let" and "edit" commands), and a variety of special MAPMAKER defined names (see the "names" command for details).

Sequential locus ranges may also be specified (the range is sequential in the data file, not sequential in the map). For example:

```
    1> sequence 1-10
```

is equivalent to:

```
    1> sequence 1 2 3 4 5 6 7 8 9 10
```

SPECIFYING MANY ORDERS USING THE "SEQUENCE" COMMAND

Set brackets ("{" and "}") in a sequence indicate that MAPMAKER should consider all permutations of the set of markers contained inside the braces. For example, if the current sequence is "{1 2 3} 4" and the map command is given, maps for the six orders:

```
    1 2 3  4     2 1 3  4     2 3 1  4
    1 3 2  4     3 2 1  4     3 1 2  4
```

will be generated. Braces are useful when the exact order of markers within a linkage group is unknown.

Square brackets ("[" and "]") indicate that no permutations need be considered which alter the placing of any markers contained within the brackets. No loci will be placed among those in the brackets, and the order of loci in the brackets will not be permuted. For example, if the current sequence is "{1 [2 3 4]} 5 6", only the orders

```
    1 2 3 4  5 6      2 3 4 1  5 6
```

are considered. Square brackets are useful when several markers are known to be very close. Note however that the "try" command will attempt to place tested markers in between loci in the sequence in square brackets.

Angle brackets ("<" and ">") work similarly to megalocus brackets (described above) except that both the order of markers within the brackets as well as that order REVERSED are considered. For example, if the current sequence is "<1 2 3> <4 5>", the four orders:

```
    1 2 3  4 5        3 2 1  4 5
    3 2 1  5 4        1 2 3  5 4
```

will be examined. Angle brackets may be useful when one has mapped several linkage groups and wishes to try them together, without knowing their configuration relative to each other.

Brackets in a sequence may be deeply nested. For example:

```
    1> sequence <{[1 2] [3 4 5]} 6 7 8> 9 10
```

specifies the four orders:

```
    1 2  3 4 5  6 7 8  9 10        3 4 5  1 2  6 7 8  9 10
    8 7 6  5 4 3  2 1  9 10        8 7 6  2 1  5 4 3  9 10
```

HISTORY Command (abbreviation 'h')

Summary:	List Previous Sequences
Argument:	<number of previous sequences to display>
Default:	20

MAPMAKER allows you to easily recall sequences previously entered. These sequences are remembered and are collectively referred to as the 'sequence history'. In MAPMAKER's "sequence" command, the syntax "#n", where n is a number, instructs MAPMAKER to recall and insert the n-th sequence used.

```
    5> sequence 2 6 9 11 8
    sequence #1= 2 6 9 11 8

    6> sequence 13 15 9 1
    sequence #2= 13 15 9 1

    7> sequence #1
    sequence #3= 2 6 9 11 8
```

To list the sequence history, simply type:

```
    4> history
    Previous Sequences:
    #1=  2 6 9 11 8
    #2=  13 15 9 1
    #3=  2 6 9 11 8
```

Sequences from the MAPMAKER sequence history may be used as pieces of new commands by including the history reference(s) with other sequence elements. Continuing the example above, we could type

```
    5> sequence 4 {#3} 12
    sequence #5= 4 {2 6 9 11 8} 12
```

EXPAND SEQUENCE Command (abbreviation 'x')

Summary:	Set the Sequence, Expanding Names
Argument:	<sequence>
Default:	expand the current sequence

Named sequences are stored in MAPMAKER sequences as are loci. Before any command is run, MAPMAKER looks up the value of each name, and builds a new "expanded" sequence accordingly. This is convenient, except with names which change frequently (for example, "order1").

The "expand sequence" command sets the current sequence, expanding all named sequences first, so that only locus names appear. With no arguments, "expand sequence" expands, and thus changes, the current sequence.

INSERT Command (abbreviation 'i')

Summary:	Insert a Marker into the Sequence
Arguments:	<marker before poisition to add> : <new marker>

The "insert" command is used to add a marker or set of markers to the current sequence. To use this command, type "insert", followed by the marker in the current sequence after which the new marker(s) should be added, a colon, and then the new markers to be added. For example,

```
    3> sequence 1 2 3
    sequence #1= 1 2 3

    4> insert 2: 4 5
    sequence #2= 1 2 4 5 3
```

To insert marker(s) at the leftmost end of the sequence, give no marker before the colon, for example:

```
    5> insert :6 7
    sequence #3= 6 7 1 2 4 5 3
```

APPEND Command (abbreviation 'a')

Summary:	Append Marker(s) to the End of the Sequence
Argument:	<marker(s) to append>

The "append" command will apend the specified marker(s) to the current sequence. For example,

```
    6> sequence 1 2 3 {4 5 6 7} 8
    sequence #1= 1 2 3 {4 5 6 7} 8

    7> append 9 10 11
    sequence #2= 1 2 {4 6 7} 8 9 10 11
```

DELETE Command (abbreviation ’d’)

Summary: Deletes a Marker or Markers from the Sequence
Argument: <marker(s) to delete>

The "delete" command will remove the specified marker(s) from the current sequence. For example,

```
    6> sequence 1 2 3 {4 5 6 7} 8
    sequence #1= 1 2 3 {4 5 6 7} 8

    7> delete 3 5
    sequence #2= 1 2 {4 6 7} 8
```

EDIT SEQUENCE Command (abbreviation ’e’)

Summary: Edit a Sequence Using the Keyboard
Argument: <sequence name or number to edit>

The ’edit’ command, when used with no arguents, allows you to edit the current sequence using the arrow keys (and others) on the keyboard. The keys used are the same as for editing commands -- type ’help keyboard editing’ for details.

If you type ’edit name’ where name is a sequence you defined using the ’let’ command, then you will instead edit the definition of that name (the current sequence will NOT be affected).

Similarly, if you type ’edit number’, where number is a sequence history number (type ’help history’ for details), then you will edit that sequence, although it WILL become the current sequence when you are done.

The ’edit’ command currently works only on certain UNIX systems. See the installation instructions for details.

LET Command (abbreviation ’l’)

Summary: Name a Sequence
Argument: <name> = <sequence>

The "let" command assigns a name to any valid MAPMAKER sequence. The sequence may any allowed types of brackets, locus names or numbers, as well as other names set using the "let" command. To refer to a previously named sequence, you simply use the assigned sequence name. Names must start with an alphabetic character, are limited to 10 characters, and may not conflict with any locus names listed in the data set. For example:

```
    3> let new7q= 14 12 [6 25] 21

    3> sequence {new7q} 13
    sequence #1= {14 12 {6 25} 21} 13
```

Note that the matching of sequence names is not sensitive to alphabetic case (e.g., "sevenq" will match "SevenQ"). Moreover, only enough characters of a name need be given in order to specify that name unambiguously. For example, the construct "pdg" will match the sequence named "PDGregion" as long as no other locus or sequence name begins with the letters "pdg".

MAPMAKER recognizes the names of individual loci in the same manner. Locus names, however, do not have to be set with the "let" command: they are set automatically when a data set is loaded. You can mix the use of locus and sequence names. For example, if "PDGregion" is set to "11 3 16", and "L1063" is locus number 1, then we may type:

```
    4> sequence PDGregion {24 L1063} 12
    The current sequence is now ’11 3 16 {24 L1063} 7’.
```

MAPMAKER automatically assigns the name "all" upon loading data to refer to a sequence which lists all of the loci in the loaded data set in order. Other commands which can set names include "group", "orders", and the chromosome related commands.

NAMES Command (abbreviation ’n’)

Summary: List the Named Sequences
No Arguments

The "names" command displays a list of all of the sequence names which have been set. For example:

```
    1> let favorite= 9 16 4 6 18
    ok

    2> names
    Special Names:
     new=       none
     anchors=   1 4 13
     assigned=  1-12 14-22
     framework= none
     unassign=  15
     group1=    1-3 7 8 11 12
     group2=    4-6 9 10 16-18
     group3=    13 14 20-22
     unlinked=  15 19
     order1=    9 4 6 18
     others:    all, none, <chromosome-name>, etc.
    User Defined Names:
     class1=    1 2 4 8 11
     class2=    20-22
     no_class=  3 5-7 8-10 12-19
     favorite=  9 16 4 6 18
```

Note that MAPMAKER lists both "special" names (names it defined itself), as well as user defined names (set either using the "let", "expanded let", or "class" commands). Some names (such as "anchors", "assigned", "framework", etc.) take on special meanings when used with the "genome analysis" features depending on which chromosome is currently selected. See the discussion of this below for details.

FORGET NAMED SEQUENCE Command

Summary: Erase a Named Sequence
Argument: <name>

This command instructs MAPMAKER to remove a name from the current list of names remembered. For example,

```
    1> let p_arm= 1 2 3
    p_arm= 1 2 3

    2> let q_arm= 4 5 6
    q_arm= 4 5 6

    3> forget p_arm
    ok
```

TRANSLATE Command (abbreviation ’t’)

Summary: Show the Names and Numbers of Loci in the Sequence
No Arguments

When given locus names and/or numbers as argument(s), the "translate" command displays both the name and number of all the specified markers. Otherwise, the "translate" command displays the marker number and name for each marker in the current sequence. We could type:

```
    7> trans 1 2 3 4 5 6 7 8
       1 TG_24         2 TG125         3 cd_15         4 tg175
       5 cd35          6 tg93          7 tg71          8 tg83
```

The "translate" command recognizes sequence names in its arguments. Therefore, you can type:

```
    8> translate all
```

to list the names and numbers of all loci in your data set.

(4) TWO-POINT (PAIRWISE) ANALYSIS COMMANDS

MAPMAKER provides a number of commands which analyze data by calculating the pairwise recombination fractions and maximum LOD scores between markers, and examining these results alone. MAPMAKER’s two-point analysis facilities are distinct both algorithmically and operationally from its multipoint capabilities. While two-point analysis is useful for finding linkage groups and determining rough proximity of markers, it is often problematic to use two-point analysis alone for determining correct genetic order.

In general two markers are declared linked only if (1) their LOD score is greater than a specified minimum LOD, and (2) if the maximally likely map distance between them is less than a specified maximum. By default, MAPMAKER uses the criteria specified by the "default linkage criteria" command, unless told otherwise by the command’s arguments. Generally, MAPMAKER assumes that map distance values given as arguments greater than or equal to 0.50 are centimorgan distances (using the currently selected "centimorgan function"), while values less than 0.50 are assumed to be recombination fractions.

TWO POINT Command

```
Summary:     Compute Pairwise Distances and LOD Scores
No Arguments
```

The command "two point" instructs MAPMAKER to compute the two-point map distances and LOD scores for all pairwise combinations of the markers in the current sequence. This information is stored by MAPMAKER and may be accessed by any of several commands, including "group", "lod table", "big lods", and others. It is NOT necessary to ever type the "two point" command: MAPMAKER will compute the two-point values as needed. Precomputing these numbers is simply a convenience for speeding later analyses (note that this is NOT true of three-point analysis however).

LOD TABLE Command

```
Summary:     Print All Two-Point Data for the Current Sequence
Argument:    <half or full>
Default:     half
```

The "lod table" command displays a table of map distances and LOD scores for all pairwise combinations of markers listed in the current sequence. The map distances are displayed either as centimorgan distances or recombination fractions, depending on the setting of the "units" option. The optional argument indicates whether MAPMAKER should print a half matrix or full-matrix.

```
    7> seq 4 6 9 16 17 18
    sequence #2= 4 6 9 16 17 18

    8> lod table
    Bottom number is LOD score, top number is centimorgan distance:

          4     6     9     16    17

    6     19.3
          6.69

    9     58.2  80.5
          0.84  0.47
    16    21.4  56.0  22.2
          5.43  1.23  4.11

    17    32.9   7.9  78.0   -
          2.91 14.57  0.49

    18    42.2  18.7   -     -    14.1
          1.77  7.21              9.00
```

Pairs where a dash replaces the LOD and distance are considered completely unlinked, defined as having a LOD less than 0.5 AND a distance greater than 40% recombination.

BIG LODS Command

Summary:	List Linked Pairs of Markers in Sequence
Argument:	<minimum LOD> <maximum distance>
Default:	default linkage criteria

The "big lods" command displays the map distance and LOD score for all pairwise combinations of loci in the current sequence which have a LOD score greater than or equal to the specified minimum, and a distance less than or equal to the specified maximum. If not supplied as arguments, the LOD and map distance thresholds are obtained from the "default linkage criteria" parameter. For example, to see the two-point maps for all pairs with LOD scores of at least 4.0, type:

```
    biglods 4.0
```

If no threshold is given as an argument, those specified by the "default linkage criteria" setting are used.

NEAR Command

Summary:	List Markers in Sequence Near Other Marker(s)
Argument:	<list of markers> <: <minimum LOD> <maximum distance>>

The "near" command prints a list of which markers in the current sequence are linked to markers in a list (given as the command’s argument(s)). Each such pairwise combination that has a LOD score above the minimum LOD threshold and that lies at a distance less than or equal to the maximum distance is reported. If not supplied as arguments, the LOD and map distance thresholds are obtained from the "default linkage criteria" parameter.

```
    6> seq 4 6 9 16 17 18
    sequence #19= 4 6 9 16 17 18

    7> near 5 10
    Linked Markers in Sequence at min LOD 3.00, max Distance 50.0

     Marker-1   Marker-2   Theta    LOD     cM
     5          4           0.05   15.32    5.05
     5          6           0.12    9.80   14.25
     5          16          0.24    3.14   33.08
     5          17          0.22    4.35   28.46
     5          18          0.25    3.12   34.46
     10         4           0.18    5.33   22.94
     10         9           0.14    5.46   16.01
     10         16          0.03   17.49    2.78
```

By using a colon in the argument list, you can optionally specify LOD and distance thresholds other than the defaults to use. For example:

```
    8> near 5 10: 4.0 30
```

prints the nearby markers at a LOD of at least 4.0, and a maximum distance of 30 centimorgans.

LINKS Command

Summary: Find Any Markers Near Given Marker(s)
Arguments: <chromosome> <minimum LOD> <maximum distance>

The "links" command prints a list of which markers in the current sequence are linked to either (i) any markers in the entire data set (if either no chromosome is specified, or if the specified chromosome is "any"), or (ii) any markers assigned to the specified chromosome. A chromosome can be specified either as an argument, or by using the "sequence" command beforehand. Each such pairwise combination that has a LOD score above the minimum LOD threshold and that lies at a distance less than or equal to the maximum distance is reported. If not supplied as arguments, the LOD and map distance thresholds are obtained from the "default linkage criteria" parameter.

```
    6> seq 5 10
    sequence #25= 5 10

    7> links
    Linked Markers in Data Set at min LOD 2.00, max Distance 50.0

     Marker-1   Marker-2   Theta    LOD     cM
     5          4           0.05   15.32    5.05   (c2)
     5          6           0.12    9.80   14.25
     5          10          0.26    2.82   36.08
     5          16          0.24    3.14   33.08
     5          17          0.22    4.35   28.46
     5          18          0.25    3.12   34.46
     10         4           0.18    5.33   22.94   (c2)
     10         5           0.26    2.82   36.08
     10         9           0.14    5.46   16.01
     10         16          0.03   17.49    2.78
```

PAIRWISE Command

Summary: Print Two-Point Data Between Sequence and Other Loci
Argument: <list of markers>

The "pairwise" command displays the two-point distances and LOD scores between the specified marker (or markers) and the loci listed in the current sequence. The output produced by this command is similar to that of the "lod table" command. For example:

```
    5> seq 4 6 9 16 17 18
    sequence #19= 4 6 9 16 17 18

    6> pair 5 10
    Bottom number is LOD score, top number is centimorgan distance:

          5     10

    4      5.0  22.9
         15.32  5.33

    6     14.3  73.5
          9.80  0.52
    9     76.6  16.0
          0.41  5.46

    16    33.1   2.8
          3.14 17.49

    17    28.5   -
          4.35

    18    34.5   -
          3.12
```

GROUP Command

```
Summary:     Separate Markers in Sequence into Linkage Groups
Arguments:   <mimimum LOD> <maximum distance>
Defaults:    default linkage criteria
```

The "group" command displays possible linkage group assignments of the markers listed in the current sequence, based on two-point analysis. All pairs of loci that have a distance less than or equal to the specified maximum and a LOD score exceeding a specified minimum are considered linked. For the purpose of computing linkage groups, linkage is considered fully transitive (i.e., if A is linked to B, and B is linked to C, then A, B, and C will be included in the same linkage group). For example, typing:

```
    9> group 4.0 30
```

displays linkage groups of markers in the current sequence, where all loci within any linkage group will have a minimum LOD 4.0 linkage with a maximally likely map distance of no more than 30 cM to at least one other locus in the group.

```
    8> seq all
    sequence #6= all

    9> group 4.0 30
    Linkage Groups at min LOD 4.00, max Distance 30.0

    group1= 1 2 3
    -------
    group2= 4 5 6 9 10 16 17 18
    -------
    group3= 7 8 11
    -------
    group4= 13 14 20 21
    -------
    unlinked= 12 15 19 22
```

If not supplied as arguments, the LOD and map distance thresholds are obtained from the "default linkage criteria" parameter.

DEFAULT LINKAGE CRITERIA Parameter

```
Summary:     LOD and Distance Thresholds for Two-Point Linkage
Arguments:   <minimum LOD> <maximum distance>
Defaults:    with no argument, displays current setting
```

This parameter controls how MAPMAKER defines two-point linkage by default. These default parameters are used to define linkage for many MAPMAKER functions (including "group", "biglods", "near", etc.) unless specific parameters are provided to those functions.

For example, typing:

```
    1> default linkage criteria 3.0 0.30
```

sets the default LOD threshold to 3.0 and the default maximum map distance to a recombination fraction of 0.30. When MAPMAKER begins, the default linkage criteria, are LOD 3.0 and distance 50 Haldane cM. As usual, MAPMAKER assumes that map distance values greater than or equal to 0.50 are centimorgan distances, while values less than 0.50 are assumed to be recombination fractions.

SUGGEST SUBSET Command

Summary:	Find Highly Informative Well Spaced Markers
Arguments:	<minimum LOD> <maximum distance>
Defaults:	default linkage criteria

The "suggest subset" command first breaks the markers in the current sequence into linkage groups at the specified criteria, in the same way as the "group" command. For each group, MAPMAKER then tries to find a subset of the markers which are both highly informative individually, and for which there exist no pairs closer than a specified MINIMUM distance. These two thresholds are set by the "informativeness criteria" command. For example:

```
    90> seq all
    sequence #27= all

    8> inf crit 2 50
    Informativeness Criteria: min Distance 2.0, min #Individuals 50

    9> suggest subset
    Informative Subgroups at min LOD 3.00, max Distance 50.0
    Informativeness: min #Individuals 50, min Distance 2.0

    Linkage group 1: 1 2 3 7 8 11 12 13 14 20 21
    8 Markers in informative subset:
    order1= 1 2 3 7 8 12 14 21
    other1= 11 13 20
    -------
    Linkage group 2: 4 5 6 9 10 16 17 18
    All markers are informative.
    order2= 4 5 6 9 10 16 17 18
    other2=
```

For your convenience, the names "orderN" and "otherN" are set to specify the informative subset and remaining markers, respectively, for the Nth linkage group.

INFORMATIVENESS CRITERIA Parameter

Summary:	Criteria for Finding Highly-Informative Markers
Arguments:	<minimum distance> <minimum individuals> <codominant>
Defaults:	with no arguments, displays the current values

Markers will be included in informative subsets (as in the "suggest subset" and "order" commands) if they meet the criteria: (i) No distance between any pair of markers is less than the specified minimum distance; (ii) at least the specified mimimum number of individuals are informative for each marker, and (iii) if the data set is for F2 or F3 intercross, and if the word "codominant" is specified as the third argument, then the markers must be scored codominantly (e.g. no C/D scores in raw data file).

When MAPMAKER perpares a data file, the informativeness criteria are set to a minimum distance of 1cM, and a minimum of 1 informative individual (essentially no test). You need to set these criteria to apply a better test.

(5) MULTIPOINT ANALYSIS COMMANDS

Fast multipoint analysis is the distinguishing point of MAPMAKER as compared to some of the other linkage analysis methods available. Multipoint analysis takes into account the primary genotype data for all loci simultaneously, when determining map orders, distances, and map likelihoods. We contrast this approach with so-called "Multi-Two-Point" methods, which take into account all pairwise recombination fractions simultaneously, but not the raw data. Multipoint analysis enforces greater consistency of the available data, and thus extracts more information than can be captured by even the complete set of pairwise LODs.

MAP Command (abbreviation ’m’)

```
Summary:     Compute and Print a Maximum Likelihood Map
No Arguments
```

The "map" command calculates and displays the maximum likelihood map for the order (or orders) of markers specified by the current sequence. Locus names and numbers are displayed, with distances in cM or recombination fraction (see "units"), as are any candidate errors if "error detection" is on (see the discussion of MAPMAKER’s error detection below).

DRAW MAP Command

```
Summary:     Compute and Draw a Maximum Likelihood Map
Arguments:   <file name> <scale>
Defaults:    map.ps
```

The "map" command calculates and draws as a PostScript graphic file the maximum likelihood map for the order of markers specified by the current sequence. These files may be viewed or printed if you have the appropriate software and/or printer. If a file name is given, the graphics are placed there, otherwise the name of the data set (with the extension ".ps") is used. A map is drawn on one page, with the marker names shown and interval distances to scale. If a numeric argument is given after the file name, this will be the scale (in dots per centimorgan) used to draw the map. Most printers will display 72 dots per inch. If this argument is omitted, the map will be drawn to a scale that covers the full length of the page.

COMPARE Command (abbreviation ’c’)

```
Summary:     Compare Likelihoods of Many Map Orders
Arguments:   <number of maps to remember> <log-likelihood threshold>
Defaults:    20 infinite (no thresholding)
```

The "compare" command calculates the maximum likelihood map for all orders of markers specified by the current sequence. (Recall that one sequence can specify more than one order of loci. For example, the sequence "{1 2 3}" specifies the three orders: "1 2 3", "1 3 2", and "2 1 3"). Unlike the "map" command however, the orders are diplayed sorted by the likelihoods of their maps. For example:

```
    67> seq {9 6 18 4}
    sequence #34= {9 4 6 18}

    70> compare
    Best 12 orders:
```

```
    1:    9 4 6 18   Like:  0.00
    2:    4 6 18 9   Like: -3.49
    3:    6 18 4 9   Like: -4.23
    4:    9 6 18 4   Like: -4.59
    5:    9 6 4 18   Like: -5.05
    6:    6 4 18 9   Like: -5.26
    7:    4 9 6 18   Like: -5.99
    8:    6 18 9 4   Like: -6.23
    9:    6 4 9 18   Like: -6.84
    10:   4 6 9 18   Like: -7.07
    11:   6 9 4 18   Like: -11.38
    12:   6 9 18 4   Like: -11.96
    order1 is set
```

This test indicates that of the 12 orders specified by the sequence "{9 6 18 4}", one of them (the order "9 4 6 18") is preferred over all others by at least a likelihood difference of 10 raised to the 3.49 power, indicating that this map is about 3100-fold more likely than the others. The best map of course has a log-likelihood of 0.0 relative to the best (itself). For your convenience, the name "order1" is always set to the most likely map order found.

The arguments to the "compare" command specify how many of the most likely orders will be displayed. The first number indicates how many (maximum) will be displayed. The second is a log-likelihood threshold (a positive real number) -- only orders within this much of the best will be displayed. By default the best 20 orders are displayed with no log-likelihood cutoff.

When ’use three point’ is on, some orders may be excluded by three-point tests, in which case they are never assigned a multipoint log-likelihood at all, and they are not displayed (although the total number of orders excluded is). Note tnat the ’compare’ command takes into account "error detection", if it is on, although the computed error LODs are never actually printed out. Use the "map" or "genotypes" command to do this for the most likely order or orders.

TRY Command

Summary: Insert Markers into an Order and Compare Likelihoods
Argument: <loci to try>

The "try" command is used to place an unmapped marker or markers relative to a known order of loci to which it is known to be linked. MAPMAKER computes the maximum likelihood maps for the current sequence (which specifies the known order) with the new marker(s) inserted into each interval, and will display each map’s log-likelihood relative to the highest likelihood found (thus, the best placement receives a log-likelihood 0.0, and all others are less than 0.0, indicating how much less likely they are, given the available data). Note that while only this restricted set of orders is tried, for each order all map distances are recomputed de novo for each map calculation using full multipoint linkage analysis.

The "try" command should be given as arguments the marker(s) by name or number to place (for example, "try 25 28"). If more than one marker is to be tried, they are each tried separately, yielding exactly the same results as if "try" command is used separately for each marker. If the current sequence specifies more than one order (for example "10 11 {12 13} 14") the marker(s) are tried into each such order separately.

```
    2> seq 9 4 6 18
    sequence #1= 9 4 6 18
    3> try 16 5 10 17

              16      5       10      17
             -------------------------------
            | -4.59  -17.68   -3.45  -15.75 |
    9       |                               |
            |  0.00   -2.69    0.00  -13.00 |
    4       |                               |
            | -9.58    0.00  -10.68   -5.22 |
    6       |                               |
            |-15.40  -12.54  -16.46    0.00 |
    18      |                               |
            | -9.26  -14.74   -9.83   -6.30 |
            |-------------------------------|
    INF     | -9.62  -18.54   -9.90  -16.37 |
             -------------------------------
    BEST     -91.47  -82.55  -90.28  -85.93
```

In this example, MAPMAKER has displayed the relative likelihoods of
the maps containing markerd 16, 5, 10, and 17 inserted into all
intervals in the order "9 4 6 18". For marker 16, the best map
corresponds to the order "9 16 4 6 18", with a log-likelihood of
-91.47. The next best order is "16 9 4 6 18", with a log likelihood
of -4.59 relative to the best, making it about 40,000-fold less
likely. The likelihood that the marker is unlinked (as opposed to
being placed into the map) is much lower still, -9.62 on a log scale.

The "try" command can also handle the insertion of two or more markers
together in the order. Square and angle brackets can be used in the
list of markers to try in the same way they are used in the "sequence"
command. For example, typing:

```
    try [24 25]
```

will instruct MAPMAKER to insert the pair of markers [24 25] together
into each of the selected intervals in the current sequence. Typing:

```
    try <7 8 9>
```

yields two results, one for trying [7 8 9], and one for trying [9 8 7].
This feature greatly expands the number and types of tests which
can be run using the "try" command. Particularly, this allows one to
try recombinationally inseparable loci, using all of the available data
to place them.

When 'use three point' is on, some orders may be excluded by
three-point tests, in which case they are never assigned a multipoint
log-likelihood at all. An "x" is printed as the likelihood for these
placements, for example:

```
              16
             -------
            | -4.59 |
    9       |       |
            |  0.00 |
    4       |       |
            |   x   |
    6       |       |
            |   x   |
    18      |       |
            |   x   |
            |-------|
    INF     | -9.62 |
             -------
    BEST     -91.47
```

indicates that the orders "9 4 16 6 18", "9 4 6 16 18", "9 4 6 18 16", were all excluded by three-point analysis.

Because the "try" command is usually only used to examine a limited number of orders, it is most useful only when the linkage group in question has been thoroughly analyzed and its map order is well trusted. Note that the ’try’ command takes into account "error detection", if it is on, although the computed error LODs are never actually printed out. Use the "map" or "genotypes" command to do this for the most likely order or orders.

RIPPLE Command

```
Summary:     Test a Map Order by Permuting Local Orders
Arguments:   <window size> <log-likelihood threshold>
Defaults:    3 2.0
```

The ripple command tests a map order by comparing the likelihood of the original map order to those found when the order of neighboring loci are permuted. Analytically, this is much like using a ’sequence’ with set brackets (to indicate ambiguity) along with the ’compare’ command. ’Ripple’ is convenient because is automates many such analyses for you.

Before using the ripple command, you should set the sequence to a (single) order of loci to test. Ripple’s two optional arguments specify (i) the number of neighboring markers to permute at one time (i.e., the number to put within set brackets), and (ii) the log-likelihood difference threshold for declaring when an alternative order should be displayed. Note that the ripple command uses ALL flanking markers in every computation, and that it takes advantage of ’error detection’ and ’use three point’, if enabled.

For example:

```
    29> sequence r45s_dr tg1b tg175 cd35 tg93 cd66 tg50b
    sequence= r45s_dr tg1b tg175 cd35 tg93 cd66 tg50b

    27> ripple 4 4.0
    ========================================================================
    Map To Test:
      Markers          Distance
        9  r45s_dr      16.3 cM
       10  tg1b         24.2 cM
        4  tg175         5.1 cM
        5  cd35         13.9 cM
        6  tg93          7.9 cM
       17  cd66         14.1 cM
       18  tg50b      ----------
                        81.5 cM   7 markers   log-likelihood= -105.42
    ========================================================================
    Window-size: 4  Log-likelihood Threshold: 4.00
    Comparing maps with ALL flanking markers...

    compare    {r45s_dr tg1b tg175 cd35}...
    best orders:
    1:    r45s_dr tg1b tg175 cd35 tg93 cd66 tg50b   Like:  0.00
    2:    tg1b r45s_dr tg175 cd35 tg93 cd66 tg50b   Like: -3.40

    compare ...{tg1b tg175 cd35 tg93}... ok
    compare ...{tg175 cd35 tg93 cd66}... ok
    compare ...{cd35 tg93 cd66 tg50b}    ok
    ========================================================================
```

In this example, all orders where 4 neighboring loci are permuted are supported by a high log-likelihood ratio of 4.0 (a 10,000:1 ratio) with the sole exception of the order which inverts the two most proximal markers. In this case, the log-likelihood ratio is 3.4, about 2500-1 odds, in favor of the original map order.

ORDER Command

```
Summary:     Automatically Build Map Orders from Scratch
Arguments:   <min LOD> <max distance> <start size> <threshold> <num to try>
Defaults:    <default LOD> <default distance> 5 3.0 10
```

The "order" command provides a fast and powerful automated tool for mapping markers using full multipoint analysis, including ’error detection’ and ’three point’ analysis, if enabled. (Be forwarned, due to the number of tests performed, using the ’order’ command WITHOUT three-point analysis can be very slow).

The ’order’ command essentially automates the use of many other MAPMAKER commands in the following fashion:

(i) MAPMAKER first subdivides the markers listed in the current sequence into linkage groups using two-point analysis (in the same manner used by the "group" command). If given, the <min LOD> and <max distance> arguments to the ’order’ command are used as the criteria for declaring linkage groups, otherwise the ’default linkage criteria’ are used.

(ii) For each group of adequate size, MAPMAKER attempts to find a starting map order of highly informative markers supported by a high log-likelihood ratio. This analysis is similar to first (a) using the ’suggest subset’ command to select informative well spaced markers, then (b) using the ’compare’ command repeatedly until you find a subset of the highly informative markers that has only one plausible map order with high log-likelihood. The <starting size> and <threshold> arguments specify the number of markers desired in the starting order and the log-likelihood ratio which must support a single map order of those markers. The <number to try> specifies the number of such tests to make before giving up for this group. The ’informativeness criteria’ command, described below, sets the thresholds for including markers in the highly informative subset.

(iii) Having found a seed order, MAPMAKER then incrementally adds markers to the order, one at a time, in the same manner used by the ’build’ command. To accept a placement in the order as unique, MAPMAKER uses the thresholds specified by the ’multipoint criteria’ command. When ’error detection’ is on, the criteria set using the ’error thresholds’ command are also used to exclude certain orders. MAPMAKER stops when it can no longer add any markers to the order, and reports the final unique map order as well as the possible relative positions of any remaining markers.

For your convenience, MAPMAKER sets certain named sequences with the results of this operation. The name "orderN", where N is a number, is set to the unique order accepted for the Nth linkage group tested. The name "unorderN" is set to the remaining markers in the Nth group, if any.

Examples of the use of the ’order’ command are provided in the tutorial. Note that the ’order’ command is non-deterministic, in the sense that it randomly chooses starting subsets and markers to try. Thus, if you run the ’order’ command multiple times, you may get slightly different results each time. However, this provides a convenient qualitative measure of the support for a map order: if the ’orders’ command produces substantially different results with each run (not including small local ordering differences where few co-informative meioses are available), then there are likely problems in the data set. The ’error detection’ mechanism may help in such cases.

BUILD Command

Summary: Sequentially Add New Markers into a Known Map Order
Argument: <markers to add>

The ’build’ command sequentially adds markers to a known map order, which should be set using the ’sequence’ command prior to typing the ’build’ command. Criteria for placing markers into the order are specified by the ’multipoint criteria’ command. When ’error detection’ is on, the criteria set using the ’error thresholds’ command are also used to exclude certain orders. MAPMAKER stops when it can no longer add any markers to the order in unique positions, and it reports the final unique map order as well as the possible relative positions of any remaining markers.

The algorithm used works essentially as follows:

(i) The markers which need to be added to the starting order are sorted by a heuristic measure of informativeness, in order to first try to place the markers most likely to fall into single intervals in the order. Some degree of randomness is used, meaning that successive executions of the ’build’ command may produce slightly different answers. As with the ’order’ command, vastly different answers are considered indicative of problems in the raw data.

(ii) If ’use three point’ is on, three point analysis is used to determine which intervals each marker needs to be tried in. If this step places a marker uniquely in one interval, it is added to the order and MAPMAKER starts again with step (i) to place the remaining markers.

(iii) Multipoint analysis is used, as with the ’try’ command, to calculate the log-likelihoods for placing each marker into each interval. As in step (ii), if MAPMAKER places any marker uniquely in one interval (at the given threshold), it is added to the order and MAPMAKER starts again with step (i) to place the remaining markers. Certain rules are followed when adding markers in this way: (a) MAPMAKER adds any marker off the end of an order only as a last resort, as markers often place off the end if they have serious genotyping errors or have extremely few informative meioses. (b) Similarly, if ’error detection’ is on, markers which have very high error LODs in their most likely position (given the criteria set using the ’error thresholds’ command) are placed only as a last resort, to avoid having their errors confound the placement of other markers. (c) Markers which place at zero distance from a known marker are placed (arbitrarily) on one side of that marker.

(iv) When all markers are added, or when steps (ii) and (iii) are no longer able to add markers to the data set, MAPMAKER stops. (If the ’multipoint criteria’ specify two placement thresholds, then MAPMAKER first starts again from step (i) at the lower threshold). The "order1" and "other1" names are set in the same way as they are for the ’order’ command.

For example:

```
    5> seq
    sequence #3= 9 4 6 18

    6> build 16 5 10 17
    Placement Threshold-1 3.00, Window 7
    ==============================================================
    8 Markers to order:
       4 tg175         5 cd35          6 tg93          9 r45s_dr
      10 tg1b         16 tg165        17 cd66         18 tg50b

    Placing at log-likelihood threshold 3.00...
    Start:   9 4 6 18
    3pt:     9 (16) 4 6 18
    Npt:     9 16 4 6 (17) 18
    Npt:     9 16 4 (5) 6 17 18
    No unique placements for 1 remaining marker

    Map:
      Markers          Distance
        9  r45s_dr      22.1 cM
       16  tg165        20.7 cM
        4  tg175         4.8 cM
        5  cd35         13.5 cM
        6  tg93          7.9 cM
       17  cd66         14.1 cM
       18  tg50b      ----------
                        83.1 cM

    Markers placed relative to above map:
              9   16    4    5    6   17   18
              :-22-:-21-:--5-:-14-:--8-:-14-:
      10 2 ...:.**.:..*.:....:....:....:....:...

    order1= 9 16 4 5 6 17 18
    other1= 10
    ==========================================================
```

MULTIPOINT CRITERIA Parameter

Summary: Mapping Criteria for ’Order’, ’Build’, etc.
Arguments: <log-likelihood threshold> <window size> <strict threshold>

The "multipoint criteria" determine when the order-building commands ("order", "build", "place", "together") consider a marker uniquely placed into a single interval.

As is described above for the "try" command, a marker is considered uniquely placed when the log-likelihood for placement in all intervals (besides the best one) is significantly lower than that for the best interval -- the minimum such difference is specified by the <log-likelihod threshold> argument.

The <window size> argument indicates the minimum number of flanking markers which should be used to place a locus (more may be used if needed). A window size of 9 indicates that at least 4 flanking markers will be used on either side of the marker to place.

If a separate <strict threshold> is given, the order building commands will add markers in two passes: (i) first, at a strict threshold, then (ii) at the "ordinary" threshold. This is mostly to help keep the sequential placement algorithm from making a mistake early on, which propagates through the whole process.

(6) GENOME ANALYSIS FEATURES

MAPMAKER is not a database. However, MAPMAKER has the ability to, in a very simple manner, keep track of many facts about markers which make it easier to manage large Genome mapping projects in MAPMAKER. These facts include:

```
    assignments - which marker(s) are assigned to which chromosome and why
    frameworks  - a known and well trusted order of markers for a chromosome
    placements  - map position of other markers relative to a framework
    age         - whether a marker is "new" or "old"
    class       - a user-defined classification of markers
```

These data are saved between MAPMAKER sessions, although they are reset whenever a data file is prepared. The ideal way to maintain some of these data for a working data set in a state of flux is to place MAPMAKER commands in your ".prep" (or ".pre") file which set some of these values, as described above for the "prepare data" command.

Commands used include:

```
    make chromosome, assign, attach, unassign, anchor, framework
    place, together
    list loci, list status, list chromosomes
    draw chromosome, draw all chromosomes
    age, class, make class
```

The Genome Analysis features described in this section need not be used. If you do use them, the only major feature not mentioned here which they affect is the "sequence" command (particularly with regard to named sequences and chromosome restrictions).

In particular, the "sequence" command allows you specify not only a set of markers, but also a chromosome for analysis. The syntax for this is <chromosome name> : <sequence>. For example, typing:

```
    1> sequence chrom1: 1 2 3 4 5 7 8 11 12
```

instructs MAPMAKER to consider the ordered list of loci from 1...12 on the chromosome named "chrom1" (note that you must have already defined a chromosome with this name using the "make chromosome" command). With most commands, it is an error to use markers in this manner which are assigned to a different chromosome (they may be unassigned however).

Certain named sequences are changed based on the currently selected chromosome. The include:

```
    assigned  - all markers assigned to this chromosome
    framework - the framework order for this chromosome
    anchors   - anchor loci for this chromosome
    new, old  - new and old loci on this chromosome
```

Once a chromosome has been selected, ALL following sequences are assumed to specify marker(s) on that chromosome ONLY, until either another chromosome is selected, or the chromosome restriction is removed. (That is, MAPMAKER maintains a notion of the "current chromosome", in addition to its notion of the "current sequence".) Removing the restriction is done by using the word "any" in place of a chromosome name, for example:

```
    1> sequence any: 1 2 3 4 5
```

in which case MAPMAKER would no longer restrict analyses to a particular chromosome. Alternatively, typing:

```
    2> sequence all
```

Removes the chromosome restriction and sets the sequence to specify all loci in the data set. As shorthand, you can type things like:

```
    1> sequence chrom1
```

to abbreviate:

```
    1> sequence chrom1:assigned
```

Commands that make use of the selected chromosome include "place", "together", and various other commands described in the "automation" section. Also see the "make chromosome", "assign", and "names" commands for details.

MAKE CHROMOSOME Command

```
Summary:     Specify the Name(s) or Chromosome(s)
Argument:    <chromosome names>
```

The ’make chromosome’ command declares one or more named chromosomes to exist. (Upon creation, no markers are assigned to the new chromosome). There is presently no way to completely undo the ’make chromosome’ command -- once declared, a chromosme exists until the data file is re-prepared. Luckily, "extra" chromosomes may be safely ignored if no loci are assigned to them.

ASSIGN Command

```
Summary:     Assign Markers to a Chromosome by Linkage
Arguments:   <min LOD> <max distance> <maximum unlinked LOD> <borderline min LOD>
Defaults:    3.0 30 (haldane cM) 2.0 3.0
```

Using simple pairwise linkage data, markers in the current sequence are assigned to linkage groups which are anchored to chromosomes. Linkage tests are first performed at the standard lod and distance thresholds, and any unplaced are then tested at the "borderline" threshold. As with the ’group command, pairwise linkage is considered fully transitive. MAPMAKER remembers all such chromosome assignments and uses them for commands such as ’place’. Assignments are saved between MAPMAKER sessions and may be viewed with the ’list loci’ or ’list assignments’ command. Note that with large data sets, you will likely need to use more stringent criteria than the defaults, because of the incresed cumulative chance of spurious linkage.

ATTACH Command

```
Summary:     Assign Markers to a Chromosome, Regardless
Argument:    <chromosome name>
Default:     <the current chromosome, selected with the ’sequence’ command>
```

The ’attach’ command assigns the loci in the current sequence to the specified chromosome, WITHOUT performing any linkage tests. It is assumed that you know what you are doing.

UNASSIGN Command

```
Summary:     Unassign Markers from All Chromosomes
No Arguments
```

The ’unassign’ command unassigns loci in the current sequence from all chromosomes. No linkage analysis is performed -- it is assumed that you know what you are doing.

ANCHOR Command

```
Summary:     Specify the Anchor Loci for a Chromosome
Argument:    <chromosome name>
Default:     the currently seleted chromosome
```

Anchor loci are used by the ’assign’ command to determine which chromosome any particular linkage group belongs to -- if any locus in the group is an ’anchor’ on some chromosome, then the entire group is assigned to that chromosome (contradictions notwithstanding).

The anchor loci for a chromosome may be changed by re-issuing the "anchor" command. Of course, this can have implications for the saved assignment and placement data. You should reassign all other markers after changing anchors.

FRAMEWORK Command

```
Summary:     Set the Framework Map Order for a Chromosome
Argument:    <chromosome name>
Default:     <the current chromosome, specified using the ’sequence’ command>
```

The ’framework’ command declares the framework order of the specified chromosome to be that specified by the current sequence. For the sequence, a map is computed exactly as if the ’map’ command had been invoked, and this map is remembered. Changing the framework of a chromosome will cause all placement information for that chromosome to be cleared. A sequence of "none" may be used to clear the framework. All loci in the framework must already be assigned to the specified chromosome.

PLACE Command

```
Summary:     Place Markers Relative to a Chromosome Framework
Argument:    <log-likelihood threshold>
Default:     2.0
```

The place command maps all markers in the current sequence (which have been assigned to a chromosome) relative to the framework order of their chromosome. ’Place’ is thus similar to ’try’, however place makes a judgement as to whether the marker mapped to a single interval or not (and if not, why not) and records this information inside MAPMAKER. Possible outcomes for a marker include:

```
    UNIQUE   mapped to a single interval at the log-likelihood threshold
    ZERO     recombinationally inseparable from a framework locus
    REGION   fits in more than one interval at the specified threshold
    OFF-END  mapped off one or both ends of the chromosome
    ERROR    if ’error detection’ is on, candidate errors were discovered
    PROBLEM  three-point analysis excluded the marker from all intervals
```

The ’list results’ and ’show placements’ commands will redisplay previously computed placement data. The ’together’ command may be used to resolve multiple loci which place near each other, and can automatically insert placed loci into the framework order. Placement data are saved between invocations of MAPMAKER.

TOGETHER Command

Summary: Places Loci Together into the Framework

The ’together’ command takes all loci in the current sequence which have been mapped relative to a framework, and tries to resolve them relative to each other -- when multiple loci are plaaced into the same interval, all possible orders of those loci are tried. When the number of possible orders grows too large, a simple "greedy" heuristic is used to incrementally extend the framework. Of course, it is always best to verify orders selected by this method using commands such as ’ripple’, ’compare’, and ’try’. Note that the current framework is not changed, however a variable named "together" is set (uniquely for each chromosome, of course) to the final accepted order. You could then update the framework by typing something like:

```
    sequence chrom1
    together
    framework chrom1
```

LIST LOCI Command (abbreviation ’ll’)

Summary: List Various Facts about Some Loci
Argument: <loci to list>

The ’list loci’ command displays, for each locus specified in the argument list, (i) its number and name; (ii) the sort of typing data available for it and the number of informative individuals (in an F2 intercross, codominant loci are indicated as "codom", dominant and recessive loci are indicated as "+/-", and loci with a mixture are indicated as "mixed"); (iii) the assigned linkage group (via the ’group’ command) and the assigned chromosome (via the ’assign’ or other commands); (iv) the assigned haplotype group (via the "join haplotypes" command); (v) the assigned class (via the "class" command); and (vi) the assigned age status (old or new, set via the "age" command). If no arguments are given, then the information is displayed for loci in the current sequence.

LIST STATUS Command (abbreviation ’ls’)

Summary: List Mapping Status for Some Loci
Argument: <loci to list>

The ’list status’ command lists, for each locus in the current sequence: (i) its number and name, and (ii) its assigned chromosome, method of assignment, and LOD-score. For loci which are placed on the chromosome, MAPMAKER also displays (iii) the quality of its placement on the chromosome, including the support likelihood, neighboring locus, and net error LOD score.

LIST CHROMOSOMES Command (abbreviation ’lc’)

```
Summary:     List the Number of Markers Mapped to Each Chromosome
No Arguments
```

The ’list chromosomes’ command lists, for each chromosome, the number of anchor loci, the number of assigned/attached loci (includes borderline loci), the number of loci which have been placed, the number of loci which placed uniquely (either UNIQUE or ZERO status) and the number of framework loci. Totals for the data set are also displayed.

LIST ASSIGNMENTS Command (abbreviation ’la’)

```
Summary:     List the Markers Assigned to Each Chromosome
No Arguments
```

The ’list assignments’ command lists, for each chromosome, the loci which have been assigned to that chromosome (by any means, including the ’assign’, ’attach’, and ’anchor’ commands). The format is similar to that used by the ’group’ command.

DRAW CHROMOSOME Command

```
Summary:     Draw Frameworks and Placements in PostScript
Arguments:   <chromosome name> <file name> <scale>
Defaults:    the currently selected chromosome
```

The current chromosome is drawn as a PostScript graphic file. These files may be viewed or printed if you have the appropriate software and/or printer. If a chromosome name is specified as an argument, that chromosome is drawn, otherwise the currently selected chromosome is drawn. If a file name is given, the graphics are placed there, otherwise the name of the chromosome (with the extension ".ps") is used.

A chromosome is drawn on one page, with the framework markers and distances in bold type. Placed markers are placed next to the upper marker flanking the interval they place in, and are printed along with their distance from that marker. (The obvious exception being markers which place off the top end, for which the distance is reversed.) Markers which place into unique intervals are drawn in regular type, others are drawn in italic type (in their most likely position.)

If a third argument is given after the file name, this will be the scale (in dots per centimorgan) used to draw the map. Most printers will display 72 dots per inch. If this argument is omitted, the map will be drawn to a scale that covers the full length of the page.

DRAW ALL CHROMOSOMES Command

```
Summary:     Draw All Frameworks and Placements in PostScript
Argument:    <file name>
Default:     the name of the data file
```

All defined chromosomes with set frameworks are drawn as a PostScript graphic file. These files may be viewed or printed if you have the appropriate software and/or printer. If a file name is given, the graphics are placed there, otherwise the name of the data set (with the extension ".ps") is used.

Each chromosome is drawn on one page, with the framework markers and distances in bold type. Placed markers are placed next to the upper marker flanking the interval they place in, and are printed along with their distance from that marker. (The obvious exception being markers which place off the top end, for which the distance is reversed.) Markers which place into unique intervals are drawn in regular type, others are drawn in italic type (in their most likely position.)

The scale that these chromosomes is chosen such that the longest chromosome fills the entire length of a page. This scale is displayed and may be given as an argument to the "draw map" and "draw chromosome" commands if you desire all your maps to be of the same scale.

AGE Command

Summary:	Declare Markers as ’Old’ or ’New’
Argument:	<old or new>

The age command sets the age for loci listed in the current sequence. This value is displayed by the ’list loci’ command, and the named sequences ’old’ and ’new’ are set accordingly.

CLASS Command

Summary:	Declare the Class for Markers
Argument:	<class name>

The ’class’ command sets the class for loci listed in the current sequence. This value is displayed by the ’list loci’ command, and named sequences with the same name as the class name are set accordingly. Classes are exclusive: each marker can be in one and only one class. Classes must be defined by the "make class" command.

MAKE CLASS Command

Summary:	Define New Class(es) of Markers
Argument:	<class name(s)>

The ’make class’ command defines new classes to which markers may be subsequently assigned by the "class" command. Class names must start with a letter and may be only 8 characters long or less. Only 10 classes may be defined. Initially, only the class named "no_class" is defined. Presently, there is no way to undo the "make class" command, although you may safely ignore classes to which no markers are assigned.

(7) SYSTEMATIC ERROR DETECTION MECHANISM

MAPMAKER’s error detection mechanism provides a powerful mechanism for coping with data sets which may contain a small number of isolated genotyping errors. In practice, it usually allows you to both (i) identify a small set of candidate errors which, upon rechecking in the lab, turns out to contain the vast majority of the existing errors; and (ii) to find plausible map orders and distances in the face of typing error. The typing error mechanism is described in detail Lincoln and Lander, Genomics 1992 (see the references in the Manual or in the READ.ME file).

We briefly summarize its operation here. To each locus, we assign an ’a priori’ error probability, essentially the percentage chance of mistyping which relates an observed genotype (as entered into MAPMAKER) to the true genotype (essentially as a penetrance function). When using ANY MAPMAKER multipoint analysis feature, the penetrance function is taken into account in the usual way when computing recombination fractions and log-likelihoods. In addition, after the analysis has been done, ’a posteriori’ likelihoods of error are calculated for each data point (i.e., each marker in each individual), and these are reported as "error-LODs". Error LODs indicate the strength of evidence for each data point, given the rest of the data set. Note that that calculation is done by simultaneous search: all loci are allowed to be in error while each LOD is calculated. Error LODs may range from -10.0 (or lower) to +10.0 (or higher), with higher numbers indicating greater chance of mistyping on a log-scale. Error LOD thresholds between 0.0 and +2.0 detect the majority of typing errors in typical data sets, as described in the Genomics paper.

The commands which control error detection include:

```
    ’error detection’ option
    ’error probability’ command
    ’error thresholds’ parameter
```

All multipoint analysis commands use error detection, including ’map’, ’compare’, ’try’, ’orders’, and ’genotypes’. Three-point analysis may also use error detection, as described below for the ’triple error detection’ option.

ERROR DETECTION Option

Summary:	Turn the Typing Error Detection Mechanism On/Off
Argument:	<on or off>
Default:	off

When ’error detection’ is on, all multipoint analysis commands allow a small possibility of typing error in each locus for each individual, as determined by the ’error probability’ value for that locus. (Loci for which the ’error probability’ is zero can not, by definition, have typing errors). Commands such as ’compare’ and ’try’ search over map orders allowing errors (thus affecting the log-likelihoods reported), although the candidate errors themselves are not reported. The ’map’, ’genotypes’, and ’place’ commands will report the candidate errors with their associated LOD-error values.

Note that the setting of the ’error detection’ option does NOT control the use of error detection in three-point analysis. See ’triple error detection’ discussed above.

ERROR PROBABILITY Command

```
Summary:     Apriori Probability of Genotyping Error
Argument:    <percentage chance of error>
Default:     none
```

The ’error probability’ command sets the ’a priori’ probability of error for loci listed in the current sequence (separate ’a priori’ error rates are maintained for each locus).

The ’a priori’ chance of error is used in ALL multipoint map construction procedures (NOT two-point procedures), when the ’allow errors’ option is on. The ’error probabilty’ command cannot be used to list these values -- the ’list loci’, ’map’, and ’genotypes’ commands display these probabilities.

ERROR THRESHOLDS Parameter

```
Summary:     LOD-Error Thresholds for Candidate Errors
Arguments:   <base threshold> <single error threshold> <net error threshold>
Defaults:    1.0 2.0 3.0
```

When ’error detection’ is on, LOD-error scores are calculated for each locus in each individual. These values can range from -10 or lower to +10 or higher -- the higher the LOD-error, the greater the ’a posteriori’ chance of error, on a log base 10 scale. The "base threshold" determines the minimum LOD-error which will be reported by the ’map’ and ’genotypes’ commands. The "single error threshold" and "net error threshold" are used by the ’order’ and ’place’ commands: if any LOD-error greater then the "single error threshold" is introduced by placing a marker into a framework, the marker will be considered in error and may not be placed. In addition, if the sum over individuals of the LOD-error values which are greater than the "base threshold" is greater than the "net error threshold", then the marker may not be placed.

GENOTYPES Command

```
Summary:     Display a Map at the Individual Crossover Level
No Arguments
```

A map is calculated for the order specified by the current sequence (exactly as with the ’map’ command) and a detailed picture of the individual’s genotypes and crossover positions is displayed. Symbols used include "X" to denote a crossover, "O" to denote two crossovers in one interval, "|" surrounding a typing to indicate a candidate error (if ’error detection’ is on), and "?" to denote an obligate recombinant which can not be placed. The total number of obligate recombinants for each individual is displayed, another useful indication of potential typing error.

(8) THREE-POINT ANALYSIS MECHANISM

Three-point analysis provides a convenient way to speed up many analytical commands which search over multiple map orders. The trade-off is that three-point analysis is not as complete as full multipoint analysis, although by using extremely conservative thresholds, you can minimize the effect this might have on your results. Briefly, three-point analysis works as follows:

1. Initially, you precompute three-point likelihoods for any linkage group using the 'three-point' command. You also need to set 'use three point' to 'on' and should set the 'triple exclusion thresholds' appropriately.

2. You use multipoint analysis as usual. For example, type:

```
    sequence {1 2 3 4 5}
    compare
```

3. As MAPMAKER searches over orders, each order is tested against the three-point data as described below. Orders found to be incompatible with the three-point results are excluded from further analysis (effectively, they are assigned an extremely low multipoint log-likelihood). Orders found to be compatible with the three-point data are subjected to full multipoint analysis as usual.

More specifically, when the 'three point' command is executed, multipoint analysis is applied to each set of three linked markers (a triple, in which we will call the markers A, B, and C). MAPMAKER computes the multipoint maps and corresponding log-likelihoods of the three orders:

```
    A-B-C     A-C-B     B-A-C
```

The log-likelihoods of these orders (relative to the best, as with the 'compare' command) are remembered.

The three-point test then works as follows: given an order of N markers (for N equal to 3 or more), MAPMAKER examines the configurations of each subset of three markers in the order. For any such subset, if three-point data have been precomputed for this triple, and if the order they are in is not the "best" or within some likelihood threshold of the best, then the entire order is considered "excluded".

It is well known that three-point analysis can produce incorrect results (e.g., excluding correct orders or accepting NO orders at all) particularly with tightly linked markers and more particularly when there are a small number of genotyping errors present. We thus recommend that the three-point analysis features be used with strict log-likelihood thresholds (at least 3.0 -- we use 4.0 or higher on our own data sets). We also recommend using three-point analysis with the error detection feature on. In practice, this helps prevent MAPMAKER from excluding orders which may in fact be correct.

The commands using this technique include:

```
    compare             try
    order               build
    place               together
```

Commands, options, and parameters used to control these features include:

```
    three point          triple exclusion criteria
    use three point      triple linkage criteria
                         triple error detection
```

Three-point log-likelihoods are saved between invocations of MAPMAKER in the ".3pt" file.

USE THREE POINT Option

Summary:	Whether Three-Point Analysis Will Be Used
Argument:	<on or off>
Default:	with no arguments, displays the current value

When ’use three point’ is ’on’, three-point analysis will be used by many order finding commands to restrict the set of locus orders which must be examined. When off, all order comparisons are done using full multipoint analysis.

In any case, three-point tests will only be performed for those loci for which three-point data have been precomputed using the ’three point’ command.

THREE POINT Command

Summary:	Precompute Three-Point Log-Likelihoods
Argument:	<fast three-point step size>
Default:	with no arguments, three point will calculate all maps normally

The "three point" command precomputes all three-point likelihoods for all linked triples of the loci listed in the current sequence (this command uses the same criteria for linkage as described for the "group" command). While these likelihoods are printed as they are computed, providing some information about map order, there are far more efficient ways to analyze these data. Repeated uses of the "three point" command add to MAPMAKER’s internal database of three-point data -- only the "forget three point" command causes three-point likelihoods to be forgotten.

If a parameter is given to the "three point" command, a faster approximation is used in the calculation of three-point likelihoods. Instead of computing each map sequentially, tables of likelihoods and recombinations are precomputed for recombination distances ranging from 0% to 50% in steps specified by the given argument. We’ve found that a step size of 1% (.01) recombination give usefully accurate likelihoods.

This "fast three point" method can be quite useful when you have a data set of several hundreds of markers or more (yielding perhaps tens of thousands of three-point maps to be computed and which can, depending on your hardware take hours to compute). During these calculations of three-point maps, much work is repeated by the program and it is these steps that are precomputed and used to approximate the actual likelihoods, yielding up to a factor of 20 speed-up for large three-point requests. The storage of the pre-calculated information, while enabling a marked speed improvement, uses a great deal of space in memory (on the order of 10-20 MB) so should only be used on those machines with sufficient memory to handle such requests.

TRIPLE LINKAGE CRITERIA Parameter

Summary:	Linkage Criteria for ’Three Point’ Triples
Arguments:	<min LOD score> <max distance> <number of links, 2 or 3>
Defaults:	5.0 30cM 3

These criteria determine which sets of three loci (triples) will be examined using the 'three point' command.

When precomputing three-point likelihoods, MAPMAKER first divides the markers in the sequence into linkage groups groups using simple pairwise analysis at the 'default linkage criteria' (as with the 'group' command). Next, subsets of three loci are found within each linkage group where the three loci alone form a linkage sub-group at the (presumably more stringent) 'triple linkage criteria'. With these criteria, if the "number of linkages" is 2, then the criteria are applied completely transitively, as with the 'group' command. If the "number of linkages" is 3, then each locus in the triple must be linked to BOTH of the other two loci individually at the specified criteria.

TRIPLE EXCLUSION CRITERIA Parameter

Summary:	Log-Likelihood Thresholds for Excluding a Triple
Argument:	<log-likelihood threshold, a positive real number>
Default:	4.0

When building maps using the 'try', 'place', 'orders', 'build', and 'extend framework' commands, any map order containing three loci in a sub-order excluded at the specified log-likelihood threshold will be considered unallowed. This test is only performed if the 'three point' command has previously been used to compute the necessary data (otherwise, only full multipoint analysis is used). If the 'triple exclusion threshold' is 0, then triples are not excluded (again, only full multipoint analysis is performed).

TRIPLE ERROR DETECTION Option

Summary:	Error Detection Option for Three-Point Analysis
Argument:	<on, off, or fixed error rate, in percent>
Default:	with no arguments, prints the current value

When precomputing three-point likelihoods, MAPMAKER may or may not use its 'error detection' mechanism, described below. If 'triple error detection' is on, then following executions of the 'three point' command will use that mechanism (assuming the per-marker error probabilities, as set by the 'error probability' command). If 'triple error detection' is set to a real number, then that number is used as the error rate in three-point analysis for ALL markers regardless of the per-locus error probability.

In many cases with data sets containing errors, using 'triple error detection' helps prevent three-point analysis from excluding correct orders because of false double crossovers. However, error detection also reduces the power of three-point analysis to exclude improbable orders.

Note that the use of error-detection in three-point analysis is completely independent of the setting of the multipoint 'error detection' option.

FORGET THREE POINT Command

Summary: Forget Precomputed Three-Point Likelihoods
No Arguments

All currently stored three point likelihoods are forgotten permanently. Useful when those likelihoods need to be recalculated, or if they are unneeded and saving/loading the resulting large data files is becoming slow.

(9) JOINING HAPLOTYPES IN LARGE DATA SETS

In data sets with large numbers of loci and relatively few informative meioses per marker, MAPMAKER analyses, while automatic, can be time consuming. As one possible optimization, MAPMAKER can collect together markers which are recombinationally inseparable, and treat them as a single locus with the concensus genotype data. Moreover, this allows markers with complementary missing data points or +/- scorings (in intercross data) to fill in for each other. This function is accomplished using the "join haplotypes" command.

The validity of this method rests on the observation that, under almost all conditions (including markers with missing data and +/- typings in intercrosses) markers unseparated by pairwise analysis always map together by multipoint analysis, as you would expect. However, this is not absolutely guarenteed by the standard genetic model under all circumstances, and it is important that you verify all results obtained using "join haplotypes" by considering the markers separately. In large simulations, we only observe differences with particularly messy data.

When haplotypes are joined, one marker in the inseparable "haplotype group" becomes the name of the group, and the fact that it is a group is indicated by a ’+’ after the name). For example, the group consiting of unseparated markers M710, B422, and Q166, might be named B422+. Unfortunately, because MAPMAKER tries to keep many results in its working data set, you cannot quickly flip back and forth between using haplotyped groups and not: in the above example, the markers M710, B422, and Q166 essentially become invisible, replaced by the marker B422+, until you either turn "join haplotypes" off or "restore haplotypes" for that haplotype-group. When either making or unmaking haplotype-groups, you may have to manually recompute various pieces of stored information (two-point and three-point data, sometimes assignments, placements, frameworks, etc). MAPMAKER warns you as to which data it had to invalidate so you can update it -- PAY ATTENTION TO THIS!

The commands used include:

```
join haplotypes   - to collect haplotypes together and turn this feature on
                    or to restore all haplotypes and turn this feature off
list haplotypes   - to list what markers are included in any haplotype group
restore haplotypes- to restore original haplotype data for only a few markers
```

JOIN HAPLOTYPES Command

```
Summary:     Collect Unseparated Markers into Haplotype Groups
Arguments:   either ’on’ <minimum LOD> <maximum distance>  or  ’off’
Defaults:    with no arguments, specifies the current value
```

To find haplotype groups of markers in the current sequence, MAPMAKER first divides those markers in the sequence into linkage groups using the specified LOD and distance criteria (as with the "group" command). Within those linkage groups, MAPMAKER then searches for recombinationally unseparated subgroups (thus using more stringent linkage criteria helps this operation run faster on large data sets). MAPMAKER displays a list of the haplotype groups it found, along with their assigned names.

As described above under this topic, when haplotypes are joined the original markers essentially become inaccessable by most commands. Only the haplotype group itself may be used (the haplotype group is essentially substituted for any reference to any of the loci in the group).

Typing "join haplotypes off" essentially removes all haplotype groups, returning MAPMAKER to its original state. Some results may need to be recomputed and settings may need to be changed: pay attention to the messages displayed by MAPMAKER. If "join haplotypes on" is entered without specification of LOD and distance criteria for linkage, the default linkage criteria in the MAPMAKER session are used.

LIST HAPLOTYPES Command (abbreviation 'lh')

Summary:	List Haplotype Groups for Certain Loci
Argument:	<list of loci>

Lists the haplotype-groups to which the specified loci belong. If no loci are specified, loci in the current sequence are used.

RESTORE HAPLOTYPES Command

Summary:	Un-Join Particular Haplotype Groups
Argument:	<list of loci>

Removes all haplotype-groups containing ANY markers in the specified list. Some results may need to be recomputed and settings may need to be changed: pay attention to the messages displayed by MAPMAKER.

(10) MAPMAKER PARAMETERS AND OPTIONS

Here, we describe other MAPMAKER options and parameters not described elsewhere.

PRINT NAMES Option

Summary:	Display Locus Names Instead of Numbers
Argument:	<on or off>
Default:	with no argument, displays current setting

The "print names" option, when "on", instructs MAPMAKER to display the names of loci instead of their assigned numbers. Most of MAPMAKER’s commands are able to alter their output based on the setting of this option. "Print names" is "off" when MAPMAKER begins. Note that when asking for loci as command arguments or as input to a prompt (including within sequences) MAPMAKER will usually accept either locus names or numbers.

PRINT MAPS Option

Summary:	Print All Maps for Placed Markers
Argument:	<on or off>
Default:	with no argument, displays current setting

The "print maps" option, when "on", modifies the behavior of MAPMAKER’s ’order’, ’build’, ’place’, and ’together’ commands. When ’print maps’ is ’off’, a cartoon is drawn showing the intervals into which certain markers can be placed. However, no further information is given, such as distances, candidate errors, etc. When ’print maps’ is ’on’, MAPMAKER also displays the multipoint map for the most likely position of each marker. Because ’print maps’ generates a good deal of output, it is ’off’ by default.

TOLERANCE Parameter

Summary:	Convergence Tolerance
Argument:	<value>
Default:	with no argument, displays current setting

Whenever MAPMAKER calculates a map, it uses an iterative technique which tests for convergece by measuring the change in log-likelihood between the newly estimated map and the previous iteration’s map. If this change is less than the value of the "tolerance" parameter, MAPMAKER determines that it has converged. When MAPMAKER begins the "tolerance" is set to 0.001.

AUTO SAVE DATA Option

Summary:	Automatically Save Data
Argument:	<on or off>
Default:	with no argument, displays current setting

Version II of MAPMAKER saves the status of many of its current settings, as well as a great many computed results (including all two and three point maps calculated) into your data files when you quit the program. This behavior is controlled by the "auto save" option, and is intended to easily let you quit and restart MAPMAKER, resuming your analysis right where you left off. By default, this option is set to "on", although you may turn it off by typing:

```
    1> auto save data off
```

Items which are saved include:

- most option and parameter values
- the command and sequence histories
- the list of saved sequence names (created by "let" and other commands)
- all two and three point map calculations
- the saved Genome Analysis information, described above

Because of the quantity of data written out, the saving process can take quite some time on slower computers, and thus you may wish to disable it after loading.

UNITS Parameter

```
Summary:     Units of Map Distance, Either cM or RF
Argument:    <type of units, cm or rf>
Default:     with no argument, displays current value
```

The "units" parameter is used to tell MAPMAKER which type of units should be used when displaying map distances.
Two choices are available:

```
    Recombination Fractions (values 0.00 to 0.50)
and Centimorgan Distances   (values 0.50 to 999.0)
```

For example, to use centimorgan distances, you would type:

```
    4> unit cm
    The ’units’ are now set to (Haldane) centimorgans.
```

Most of MAPMAKER’s commands are able to alter their output based on the setting of this parameter. The function for converting recombination fractions to centimorgan distances is selected by "centimorgan function" parameter.

When MAPMAKER begins, map distances are displayed in centimorgans. Note that when entering a map distance as a command argument or as input to a prompt, MAPMAKER usually assumes that any map distance values greater than or equal to 0.50 are centimorgan distances, while values less than 0.50 are recombination fractions.

CENTIMORGAN FUNCTION Parameter

```
Summary:     Default Map Function, Either Haldane or Kosambi
Argument:    <name of function>
Default:     with no argument, displays choices and current
```

The "centimorgan function" parameter selects the map function that MAPMAKER should use when displaying centimorgan distances. Two commonly used functions are available: the so-called Haldane and Kosambi functions.

For example, to use the Kosambi function, you would type

```
    7> cent kos
    centimorgan function: Kosambi
```

By default, the Haldane function is selected.

MORE MODE Option

Summary: Enable Pauses after Each Screen of Output
Argument: <on or off>
Default: with no arguments, displays current setting

The "more mode" option, when "on", causes MAPMAKER to pause in between screenfuls of text, prompting you to type a key before it continues displaying output. Even when set to "on", this option applies only to certain commands which perform no computationally difficult analyses but often generate large amounts of output (e.g., 'help' and 'list loci'). In addition, "more mode" is enabled only if you are using MAPMAKER interactively. "More mode" is "on" by default (unless the 'nomore' option was selected when MAPMAKER was started.

(11) MISCELLANEOUS COMMANDS

Here we describe other MAPMAKER commands not described elsewhere.

PREVIOUS COMMANDS Command (abbreviation ’p’)

Summary: Review and Re-enter Previous Commands
No Arguments

Typing "previous commands" instructs MAPMAKER to display a list of previous commands entered in the current session of MAPMAKER. To reenter a command without explicitly retyping it, simply type the number of the command you wish to enter. For example:

```
    4> previous commands
    Previous commands:
      1  load sample
      2  sequence 1 {2 3 4}
      3  seq 4 5 6 7

    5> 2
    => seq 1 {2 3 4}
    sequence= 1 {2 3 4}
```

REVIEW OUTPUT Command

Summary: Review Previous MAPMAKER Output
No Arguments

This command causes the last 125 lines of MAPMAKER output to be redisplayed on the screen, with MAPMAKER pausing in between each screenful. This is useful for reviewing the results of analyses which have since scrolled off the top of the screen (if you are using a terminal or terminal-emulator which does not provide a "scrollback" feature). Luckily, most do: ’review output’ is usually only needed on DOS systems.

CHANGE DIRECTORY Command (abbreviation ’cd’)

Summary: Change Default Directory Used by MAPMAKER
Argument: <directory name>
Default: with no arguments, displays the current default directoy

The ’cd’ command works essentially the same way it does under DOS or Unix. By default, all files are read or written from this directory unless specified otherwise. However, under DOS, it is currently NOT possible to change the current default drive (only the directory). We recommend that you only use files on your primary hard disk (usually C:), and that you are sure this disk is selected (displayed in your prompt) before you run MAPMAKER.

RUN Command

Summary: Accept Commands from an Input File
Argument: <file name>

The "run" command instructs MAPMAKER to take a series of commands from a text file. Such a file may be created using any ordinary text editor, and should contain lines of commands and other input to MAPMAKER just as they would be typed into MAPMAKER interactively.

For example, a "run" file to place a locus in a particular known order might read

```
    load chrom7
    photo place3.out
    sequence 2 5 4 7 9 6
    try 3
    all
    quit
```

and could be run with the command

```
    1> run place3.in
```

Unless the command file contains a "quit" command, MAPMAKER returns to reading user input after the commands in the file have been executed. By default, MAPMAKER assumes that the file name ends with the ".in" extension, although you may explicitly specify a different one. File names including directories may be used, for example:

```
    1> run /users/joe/work/place3.in       (UNIX)
or  1> run \scripts\place3.in              (DOS)
```

TIME Command

```
Summary:     Print Current Time
No Arguments
```

The "time" command reports the current date and time, which may be useful for timing MAPMAKER functions. For example:

```
    3> time
    The current time is:  Fri Oct 2 18:47:08 1987
```

COMMENT Command

```
Summary:     Enter a Comment into the Transcript
Argument:    <comment>
```

The "comment" command allows you to type comments into MAPMAKER which will be recorded in the output transcript file (if a ’photo’ file is open). "Comment" allows both single and multiple line comments in the following manner:

For a single line comment, type the comment as an argument:

```
    3> comment Now we will try to place marker 16

    4>
```

For a multiple line comment, give no arguments:

```
    4> comment
    Enter your comment. End it with a period (’.’) on a line by itself.

    Given that markers 8 and 9 are so close, we will place them in
    megalocus brackets so that this test will take less time.
    .

    5>
```

REMARK Command

```
Summary:     Same as 'Comment'
Argument:    <comment>
```

Type 'help comment' for details.

SYSTEM Command

```
Summary:     Run System Command(s) Without Quitting MAPMAKER
Argument:    <command name>
Default:     with no arguments, starts a system command interpreter
```

The "system" command is used to temporarily interrupt MAPMAKER and start up a new command interpreter from the operating system. Commands which are normally typed to the operating system may then be issued. This now works on most Unix and VAX/VMS systems. You can usually return to MAPMAKER by typing "exit" (on Unix or DOS), or by hitting Control-D (Unix only). For example, on a DOS system:

```
    3> system

    C:\MYDATA> dir

    stuff

    C:\MYDATA> exit

    Back in MAPMAKER

    4>
```

If you wish to execute just one operating system command, that command may be given as an argument to "system". For instance you could do the same thing as above by typing:

```
    3> system dir

    stuff

    OK

    4>
```

Note that this feature only works for simple commands, and depends heavily on your precise system configuration.

Alternatively, because MAPMAKER works under most windowing systems, you can simply open a new window to run other programs or commands while you run MAPMAKER. Performance and system resources (e.g. free memory) may be issues, depending on your system setup.

As yet another alternative, on some versions of Unix (including A/UX and SunOS) MAPMAKER may be stopped and restarted using job control features (e.g., control-Z, and the "fg" command, respectively).

IMPORT DATA Command

```
Summary:     Load Data from a Database
Arguments:   <system dependent>
```

Loads data from a database. This program needs to be customized by your particular site. By default, it does nothing.

EXPORT DATA Command

Summary: Save Data to a Database
Arguments: <system dependent>

Sends mapping results to a database. This program needs to be customized by your particular site. By default, it does nothing.

WIZARD MODE Command

Summary: Danger Will Robinson, Danger!
Argument: <on or off>

The "wizard mode" enables MAPMAKER features which are only useful for debugging or which have been delibrately disabled. Steer clear.

MAPMAKER/EXP 3.0B COMMAND REFERENCE:

MAPMAKER/EXP 3.0B QUICK REFERENCE:

(1) GENERAL INFORMATION ON MAPMAKER VERSION 3.0

About MAPMAKER	License and Contact Information for MAPMAKER
Release Notes*	Information on the Release Notes
Starting MAPMAKER*	How to Start MAPMAKER and Available Options
Tutorial*	Information on the MAPMAKER Tutorial
entering commands*	How to Type Commands Into MAPMAKER
abbreviating commands*	How to Abbreviate Commands You Type
keyboard editing*	Editing Commands and Sequences Using the Keyboard
PostScript Output*	What to Do With MAPMAKER PostScript Graphic Output

(2) BASIC MAPMAKER COMMANDS

help	Read On-Line Help Information
photo	Begin Saving MAPMAKER Output to a Text File
prepare data	Prepares a New Data Set for Analysis
load data	Load Data Set for Analysis
save data	Save Status, Mapping Results, etc.
quit	Quit from a MAPMAKER Session

(3) SEQUENCE COMMAND AND RELATED FEATURES

sequence	Select the Loci and Order(s) You Wish to Analyze
history	List Previous Sequences
expand sequence	Set the Sequence, Expanding Names
insert	Insert a Marker into the Sequence
append	Append Marker(s) to the End of the Sequence
delete	Deletes a Marker or Markers from the Sequence
edit sequence	Edit a Sequence Using the Keyboard
let	Name a Sequence
names	List the Named Sequences
forget named sequence	Erase a Named Sequence
translate	Show the Names and Numbers of Loci in the Sequence

(4) TWO-POINT (PAIRWISE) ANALYSIS COMMANDS

two point	Compute Pairwise Distances and LOD Scores
lod table	Print All Two-Point Data for the Current Sequence
big lods	List Linked Pairs of Markers in Sequence
near	List Markers in Sequence Near Other Marker(s)
links	Find Any Markers Near Given Marker(s)
pairwise	Print Two-Point Data Between Sequence and Other Loci
group	Separate Markers in Sequence into Linkage Groups
default linkage criteria	LOD and Distance Thresholds for Two-Point Linkage
suggest subset	Find Highly Informative Well Spaced Markers
informativeness criteria	Criteria for Finding Highly-Informative Markers

(5) MULTIPOINT ANALYSIS COMMANDS

map	Compute and Print a Maximum Likelihood Map
draw map	Compute and Draw a Maximum Likelihood Map
compare	Compare Likelihoods of Many Map Orders
try	Insert Markers into an Order and Compare Likelihoods
ripple	Test a Map Order by Permuting Local Orders
order	Automatically Build Map Orders from Scratch
build	Sequentially Add New Markers into a Known Map Order
multipoint criteria	Mapping Criteria for ’Order’, ’Build’, etc.

(6) GENOME ANALYSIS FEATURES

make chromosome	Specify the Name(s) or Chromosome(s)
assign	Assign Markers to a Chromosome by Linkage
attach	Assign Markers to a Chromosome, Regardless
unassign	Unassign Markers from All Chromosomes
anchor	Specify the Anchor Loci for a Chromosome
framework	Set the Framework Map Order for a Chromosome
place	Place Markers Relative to a Chromosome Framework
together	Places Loci Together into the Framework
list loci	List Various Facts about Some Loci
list status	List Mapping Status for Some Loci

list chromosomes	List the Number of Markers Mapped to Each Chromosome
list assignments	List the Markers Assigned to Each Chromosome
draw chromosome	Draw Frameworks and Placements in PostScript
draw all chromosomes	Draw All Frameworks and Placements in PostScript
age	Declare Markers as ’Old’ or ’New’
class	Declare the Class for Markers
make class	Define New Class(es) of Markers

(7) SYSTEMATIC ERROR DETECTION MECHANISM

error detection	Turn the Typing Error Detection Mechanism On/Off
error probability	Apriori Probability of Genotyping Error
error thresholds	LOD-Error Thresholds for Candidate Errors
genotypes	Display a Map at the Individual Crossover Level

(8) THREE-POINT ANALYSIS MECHANISM

use three point	Whether Three-Point Analysis Will Be Used
three point	Precompute Three-Point Log-Likelihoods
triple linkage criteria	Linkage Criteria for ’Three Point’ Triples
triple exclusion criteria	Log-Likelihood Thresholds for Excluding a Triple
triple error detection	Error Detection Option for Three-Point Analysis
forget three point	Forget Precomputed Three-Point Likelihoods

(9) JOINING HAPLOTYPES IN LARGE DATA SETS

join haplotypes	Collect Unseparated Markers into Haplotype Groups
list haplotypes	List Haplotype Groups for Certain Loci
restore haplotypes	Un-Join Particular Haplotype Groups

(10) MAPMAKER PARAMETERS AND OPTIONS

print names	Display Locus Names Instead of Numbers
print maps	Print All Maps for Placed Markers
tolerance	Convergence Tolerance
auto save data	Automatically Save Data
units	Units of Map Distance, Either cM or RF
centimorgan function	Default Map Function, Either Haldane or Kosambi
more mode	Enable Pauses after Each Screen of Output

(11) MISCELLANEOUS COMMANDS

previous commands	Review and Re-enter Previous Commands
review output	Review Previous MAPMAKER Output
change directory	Change Default Directory Used by MAPMAKER
run	Accept Commands from an Input File
time	Print Current Time
comment	Enter a Comment into the Transcript
remark	Same as ’Comment’
system	Run System Command(s) Without Quitting MAPMAKER
import data	Load Data from a Database
export data	Save Data to a Database
wizard mode	Danger Will Robinson, Danger!

* = reference information only - not a command